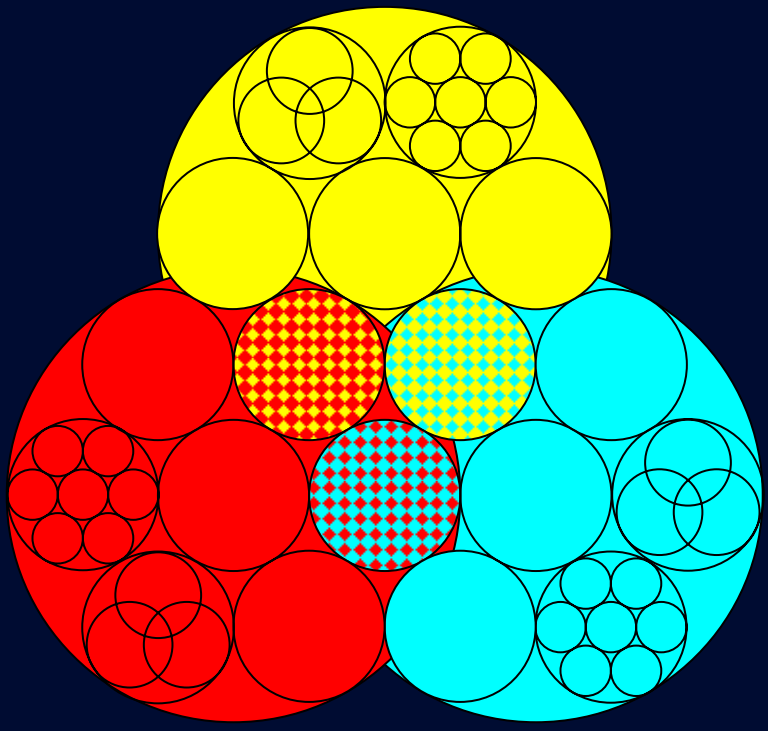
Multiprocess Multilevel Modeling

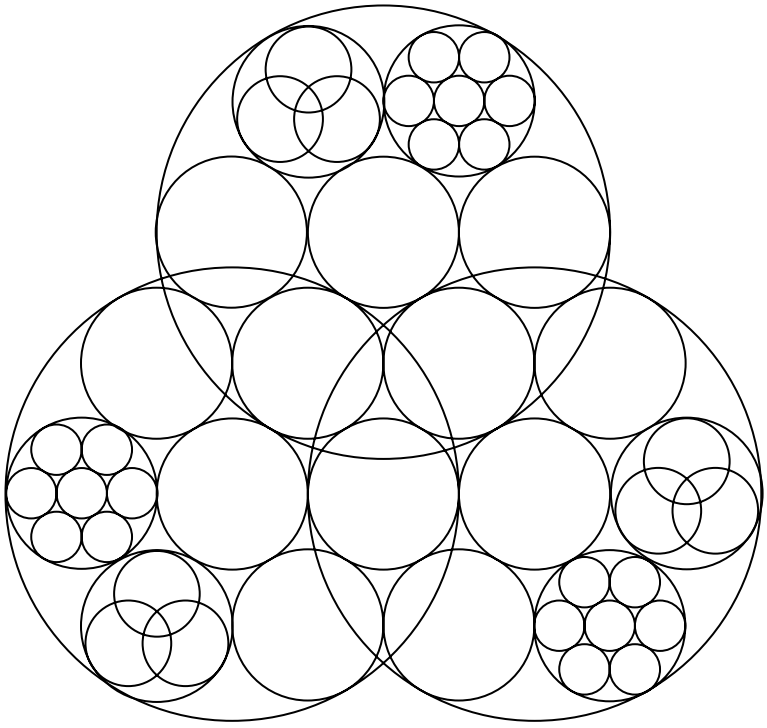


aML Version 2

User's Guide and Reference Manual

Lee A. Lillard
Constantijn W.A. Panis

Multilevel Multiprocess Modeling



aML Version 2

USER'S GUIDE
and
REFERENCE MANUAL

Lee A. Lillard
Constantijn W.A. Panis

EconWare, Los Angeles, California
http://www.applied-ml.com

Version 2.0

The suggested citation for this software is:

Lillard, Lee A. and Constantijn W.A. Panis. 2003. *aML Multilevel Multiprocess Statistical Software, Version 2.0.* EconWare, Los Angeles, California.

New in Version 2

aML Version 2 incorporates many new features, efficiency and robustness improvements, and better documentation. Among the most notable changes are the following:

- Multinomial probit models (Sections 2.11 and 13.15).
- Multinomial logit models (Sections 2.10 and 13.14).
- Poisson models (Sections 2.6 and 13.11).
- Tobit models (Sections 2.9 and 13.13).
- Newly parameterized negative binomial models, while preserving backward compatibility with Version 1 (Sections 2.7 and 13.12).
- Grid searches: specify the plausible lower and upper bounds of one or more parameters, plus the number of grid points, and aML will evaluate all grid points before selecting the best one and proceeding to estimate the model (Section 6.2).
- Improved search path by accurate evaluation of the matrix of second derivatives in cases where the BHHH approximation is poor (Section 13.1.4). This is particularly useful in problems with small sample sizes.
- Model statements may subtract as well as add building blocks (Section 13.3.2). This is useful for models with errors in variables and many more applications. For example:

```
model = regset BetaX - par Lambda;
```

- Model statements may include simple variables, without coefficient (page 327). This is particularly useful for estimating random coefficients models. For example:

```
model = varname * res(draw=..., ref=...) + ...;
```

- Dramatically more efficient algorithms for models with very many outcomes per observation.
- Numerous additions to the manual with better explanations of complex concepts such as residual draws, preparation of data with multiple levels, etc.
- Numerous additional informative error messages and warnings.

In addition, many smaller changes were made to make the program more efficient with computation and memory resources, less sensitive to numerical overflows and underflows, more user-friendly, et cetera.

Preface

When aML was first released in the year 2000, we noted that rapidly increasing availability of longitudinal survey data had spurred a lively interest in statistical methods that account for correlated outcomes. The trend has continued forcefully, with many exciting opportunities opening up for in-depth analyses that shed new light on the determinants of human behavior. By now, researchers have a choice of several excellent multilevel software packages. Even our favorite general-purpose data management and statistical analysis packages can now estimate two-level extensions of several popular model types. To date, only a handful of specialized packages support models with three or more levels. aML is one of them.

What sets aML apart are the many types of models that it can handle and its capability to estimate several types of models jointly. aML originally grew out of the substantive research interests of Lee Lillard and myself in the late 1980s and early 1990s. Lee was interested in the effect of children on the stability of marriages and was concerned that instable couples are more likely to postpone childbearing. I was interested in the effects of medical care on child health and was concerned that the frail are more likely to seek care. We worked together on many more models involving two or more domains of life: marriage, divorce, childbearing, health, disability, mortality, wages, household income, education, etc.

Lee unexpectedly passed away in December of 2000, to the tremendous loss of his family, friends, colleagues, and peers. On a professional level, his ability to clearly see the underlying statistical structure of human behavior and his ability to extract information from data was amazing. Fortunately, his legacy lives on, in part through aML.

The core algorithms of what would become aML were originally developed as part of our economic research at RAND. Most research was funded by the National Institute on Aging (NIA) and the National Institute on Child Health and Human Development (NICHD). We are grateful for subsequent support from an NICHD Small Business Innovative Research grant to develop a user-friendly interface, unify and generalize the treatment of statistical equation modules, and implement a large number of informative warnings and error messages. Since then, aML funds its own development.

aML Version 2 owes much to aML users for their critical and constructive comments, including (but not limited to) Fiona Steele, Jan Hoem, Steve Kramer, and Øystein Kravdal. Many thanks also go to Christina Ljungqvist Panis for tolerating many late nights of my absence.

Constantijn (Stan) Panis
Los Angeles, California

Contents

REFERENCE MANUAL 217

About This Manual

This book contains user documentation of aML, a statistical software package for the estimation of multilevel and multiprocess models. It consists of three distinct parts.

Part 1, *Getting Started*, provides a very brief overview of the way in which aML should be used and explains the role of each of the application files that are bundled with the aML package.

Part 2, *User's Guide*, provides an introduction to aML's multilevel multiprocess modeling capabilities. It specifies detailed sample programs for all types of supported models and discusses options and features. All sample files are also included electronically in the installation files. It starts out with very simple models, which may also be estimated using standard statistical packages, and gradually builds up to more complex multilevel and multiprocess models.

Part 3, *Reference Manual*, defines the exact data format and syntax rules and provides a full list of options. The implementation of models is defined exactly, so that interpretation of parameter values will be unambiguous.

Page

As a new user, you should follow the *Getting Started* instructions and, at a minimum, read Section 2.1 of the *User's Guide*. Depending on the nature of models in which you are interested, you may then skip to other sections of the *User's Guide*. The *Reference Manual* will benefit more experienced users.

Notation

Throughout this book, we distinguish normal text from commands and statements that users need to type into the computer by adopting different type fonts. Keywords, statements, and other strings that need to be typed literally or need to appear literally in control files are typeset in `typewriter` font. Optional statements are enclosed in [square brackets]. Statements or strings from which the user must select one, and which are not optional, are enclosed in {curly parentheses}; the alternatives are separated by the pipe ('|') character. Many statements and keywords may be abbreviated; the shortest permissible abbreviation is underlined. Names or numbers that are user-defined are *italicized*. Integers are denoted by an *"n"*, real values by an *"x"*. A list of zero or more integers is denoted by *"n...n"*; a list of zero or more real numbers by *"x...x"*; a list of zero or more variable names by *"varlist"*. Lists of integers, real numbers, and variable names must be separated by one or more blanks or commas. For example:

```
define spline [splinename];  [reference = n...n;]
   [intercept;]
   [effect = {right|full};]
   nodes = x...x;
```

In the definition of a spline, its (user-defined) name is optional. Also optional are reference numbers, which consist of zero or more integers. The word "reference" may be abbreviated to "ref" (and also to "refer", "referen", et cetera). The "intercept" specification is optional. The "effect" statement is also optional; it must be followed by either "right" or "full". The "nodes" statement is not optional; nodes are specified as zero or more real numbers. The word "nodes" may be abbreviated to "node".

We tend to indent statements that are part of the main statement, but appear on subsequent lines. This only serves to improve readability and is not part of the syntax. You may freely insert spaces and wrap statements over multiple lines. Each line may be up to 80 characters wide, including spaces. Note that every statement terminates with a semicolon.

Keywords that are part of the aML syntax may be written in lowercase, uppercase, or mixed case: `define`, `DEFINE`, `Define`, et cetera. User-defined names, such as variable names and other names (such as *splinename*, above) are case-sensitive. Variable names are limited to eight characters, other names to 12 characters.

User-specified text that is part of a statement is written in *italicized* font. Entire statements that the user may need to insert are enclosed in <arrow-shaped parentheses>. For example, this book contains many sample control files. In the interest of conciseness, we often write:

```
<...>
```

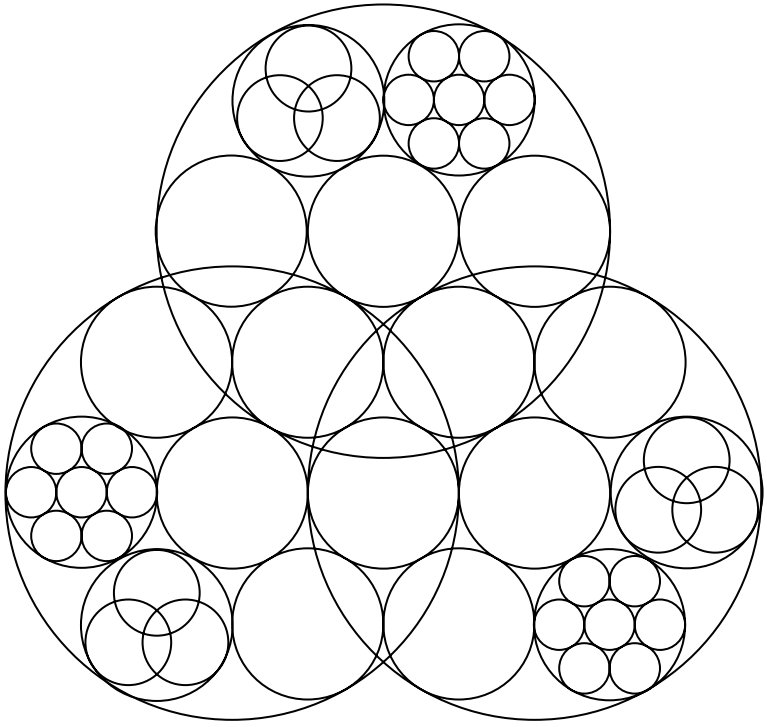
where omitted statements may need to be inserted.

Send Us Your Comments

Your input is critically important to improve aML and its user-friendliness. We would therefore like to hear from you. If you detect a bug, please contact us immediately. If something in the manual or an error message was unclear, please let us know, even if after a while you figured it out. If you would like additional features, make sure they move up on our list of priorities for further development.

The best way to contact us is by sending e-mail to <support@applied-ml.com>. Alternatively, post your question or message on the listserver for aML users which may be accessed from http://www.applied-ml.com. We look forward to hearing from you.

Multilevel Multiprocess Modeling



aML

GETTING STARTED

What is aML?

aML is a statistical software package for estimating multilevel, multiprocess models. It handles a wide variety of models: continuous outcome models, probits, logits, hazards, and count models. Many generalizations are supported, such as random coefficients models, ordered and sequential probit models, Tobit models, selection models, et cetera. All may have arbitrarily many levels and all may be mixed and matched as desired. Included with aML is a data preprocessing program and several auxiliary utilities to make life easier.

... And what is it not?

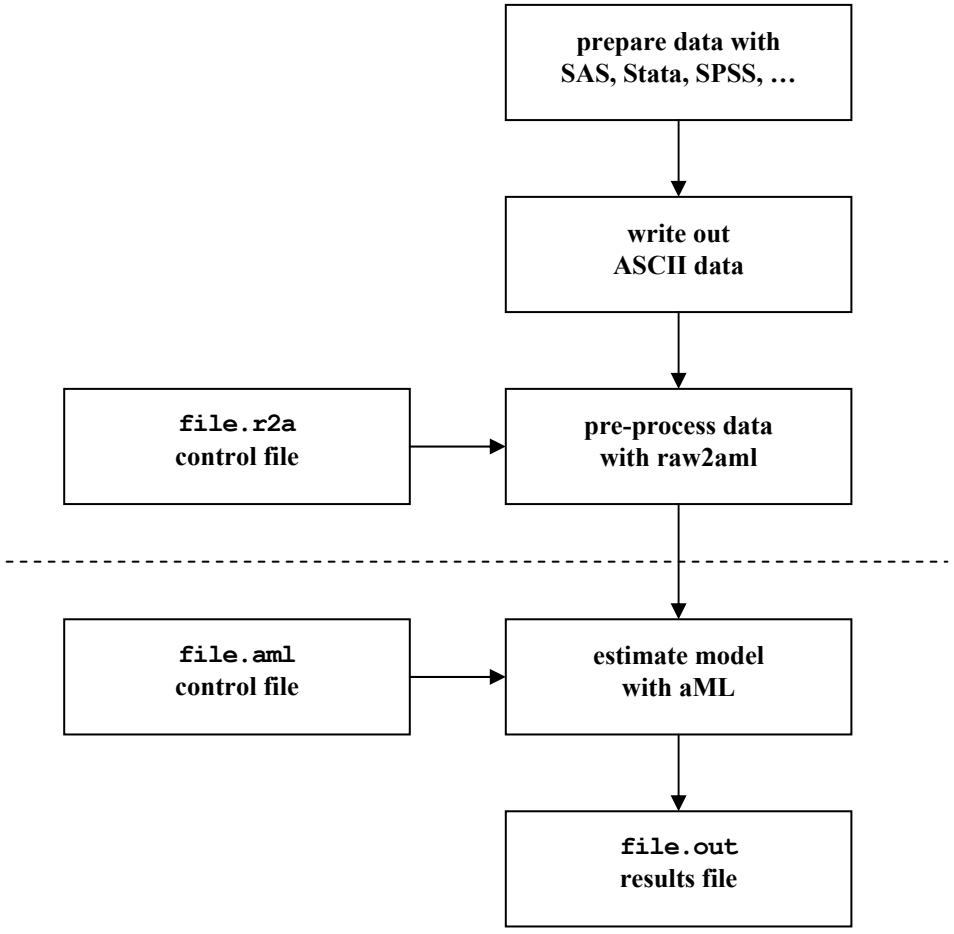
aML comes in PC, UNIX, and Linux flavors. They are almost identical. All are command-line driven programs, i.e., aML does not have a pretty user interface with windows, menu bars, et cetera. The PC version is a 32-bit application and requires Windows 95 or later (98, XP, ME, NT, 2000). You need to open a Command Prompt window, better known as a DOS window, and run aML from the command line. (You will appreciate some non-default settings; see Recommended work environment on page 3.) Models are specified in a control file which you create using your favorite editor; the output appears both on the screen and in an output file.

The briefest overview

The User's Guide will navigate you through sample problems of increasing levels of complexity. The figure below presents a schematic overview of the steps involved.

First, you need to prepare the data. This involves sample selection, resolution of inconsistencies and/or missing values, transformation of variables, et cetera. Any data management package may be used, including but not limited to SAS, Stata, and SPSS. These packages store data in an internal binary format which differs from the format aML requires. The second step is therefore to write out the data in ASCII (raw, text) format and convert them into aML's binary format using raw2aml. Raw2aml is an application that is bundled with aML. It requires a control file with information on the names of the ASCII input file(s), the level hierarchy of variables, and more. You create this control file using any text editor. We recommend extension "`.r2a`" for this control file. Raw2aml produces a data file in aML format. By default, this file has extension "`.dat`". This completes the data preparation stage.

Model estimation takes place in the second stage (under the dotted line). You need to specify your model in a control file. This control file, preferably with extension "`.aml`", may be created using any text editor. aML parses the control file, reads the data, estimates the model equation(s), and writes the results both to the screen and an output file (by default with extension "`.out`"). You may review and copy the results from any text editor. (In addition, auxiliary utility `mktab`, bundled with aML, is helpful in creating nicely formatted tables.)



Overview of Data Preparation and Model Estimation

aML programs and files

aML includes the following executable files.

`aml.exe`	The main program. Reads an aML control file and an aML-formatted data set. Estimates models and produces output files.
`raw2aml.exe`	A data preprocessor. Reads a raw2aml control file and one or more ASCII data files. Produces an aML-formatted data set and a documentation file.
`update.exe`	Utility to update the starting values of model parameters with the converged values in the corresponding output file.
`mktab.exe`	Utility to create a nicely formatted table from estimates in one or more output files.
`amltest.exe`	Utility to perform a likelihood ratio test using two output files with nested models.
`points.exe`	Utility to write out Gauss-Hermite Quadrature support points and weights, as used in numerical integration.

The table lists names of Windows programs; UNIX and Linux names are identical but without the "`.exe`" extension.

aML also comes with sample data, control files, and output files which are discussed in the User's Guide. Sample files discussed in Chapter 2 are located in "`Samples\Chapter2`"; et cetera. (UNIX and Linux pathnames contain forward slashes, e.g., "`Samples/Chapter2`".)

aML distinguishes several file types: raw2aml control files, aML-formatted data files, data documentation files, aML control files, and aML output files. Each type has a recommended extension, as described in Section 2.1.8.

Recommended work environment

aML is a command line driven program which takes a control file as input and generates an output file with results. Both the control and output file are text files, readable by the human eye. Your work environment should be optimized to conveniently create and edit control and output files, and to conveniently submit commands from a command line. You will want to be able to edit a control file and/or review an output file while a job is running in the background.

Under UNIX and Linux, a convenient work environment consists of multiple command line windows, such as X-windows. Use your favorite text file editor (vi, emacs, pico, et cetera) to edit control files and review output files in one window, and run aML jobs in another.[1]

On a PC running Windows, a convenient work environment consists of multiple Command Prompt windows, better known as DOS windows, plus a good text editor. Do not double-click on aml.exe to run aML; it will run but without arguments, write out an error message, and stop execution after a fraction of a second. Also, do not boot the machine in DOS mode. From within Windows, you may open a DOS window by pressing Start—Programs—Accessories and selecting Command Prompt. (Alternatively, press Start—Run and type "`cmd`".) By default, this opens a DOS window with 24 lines and 80 columns. We recommend that you increase the number of lines: hold the cursor over the top border; right-click the mouse; select Properties; select the Layout tab; increase the heights of the window size to, say, 60 lines. If you like, select the Font tab and select a font that suits you, such as 8x12 pixels. Make sure the DOS window comes up as a window, not in full screen mode. This is controlled under the Options (Windows NT/2000) or Screen (Windows 95/98) tab. You may toggle between window and full screen mode by pressing <Alt+Enter>. By default, the DOS window starts in a directory like "`C:\Documents and Settings\username\Desktop`". We recommend changing this initial directory to a directory in which you tend to store aML control files, say, "`C:\AML`". This may be set by first creating a shortcut to `cmd.exe`. (The location of the Command line program, `cmd.exe`, is displayed in the top border of its Properties window. Under Windows 2000 it may be, for example, "`C:\WINNT\System32\cmd.exe`".) Select the Shortcut tab of the Properties window of the shortcut and type the desired initial directory in the "Start in" box.

To edit control files and review output files, any text editor is suitable. Windows operating systems come bundled with Notepad, which is adequate. We recommend a more powerful editor like UltraEdit or Programmer's File Editor (PFE). PFE is freeware that may be downloaded from various archive sites, such as http://dl.winsite.com/files/931/ar1/win95/misc/pfe101i.zip. It is great but no longer supported. UltraEdit is inexpensive shareware; see http://www.ultraedit.com. (We are in no way affiliated with the producers of these editors.) Word processor packages like Word and Wordperfect are not suitable because, by default, they include formatting characters in the document, which should be plain text.

Under Windows, you may register file types and specify an action for Windows to take when you double-click on a file name of a registered type. For example, you could register aML control files and specify that aML should be run when you double-click on a file with extension "`.aml`". Windows would then open a small DOS window to run aML, and the window would close as soon as aML finishes. We recommend against registering aML files. First, since Windows closes the DOS window as soon as aML finishes, you would not have much benefit from the screen output.

[1] If you prefer Windows-based text editors, you may map UNIX disk drives as-if they are local to the PC and edit files that way. aML is indifferent with respect to control characters that denote the end of a line. (Under UNIX, a line is terminated by a linefeed character; DOS and Windows use both a carriage return and a linefeed. aML accepts both conventions.)

Second, you would still need a DOS window to run `update`, `mktab`, or other utilities. And third, by default, Windows Explorer does not show filename extensions for files that are registered, which makes it difficult to distinguish between the various types of files.

License issues

aML may only be used by licensed users. The first time that you run aML, it asks for your license information: an 8-character serial number, an 8-character authorization code, your name, and (optionally) your organizations' name. The serial number is printed on the medium on which you ordered aML. The serial number and authorization code are printed on a separate, single sheet of white paper with License Information. Do not lose that sheet, as you need the authorization code to re-install aML in the event of a disk crash.

To provide license information, type "`aml -l`" from the DOS or UNIX/Linux command prompt. (Argument `-l` contains the lowercase letter L, not the number 1.) As prompted, type in your serial number, authorization code, name, and (optionally) your organization's name. aML stores this information in a license file, "`aml.lic`", located in the same directory as the aML executable file. It is read every time you execute aML. To verify or update your license information, just type "`aml -l`" again.

We encourage you to visit our website at http://www.applied-ml.com. It contains helpful hints, latest updates, additional sample programs, and answers to frequently asked questions. We would appreciate it if you would register your name and e-mail address with us, so that we can notify you in the unlikely event of a bug report (http://www.applied-ml.com/register). We will not share your e-mail address or identity with other companies. Also, we urge you to join the online aML listserver, in which users help each other with modeling and technical issues. It is a free service.

Getting help

Where can you turn for help if something behaves unexpectedly? First, of course, this manual should answer most questions. The User's Guide explains most features, generally in increasing order of complexity. It contains sample data, control files, and output files, which are included in electronic form on the aML installation CD. The Reference Manual defines all commands exactly and unambiguously, including all supported options. Note the glossary starting on page 437 and the index starting on page 441.

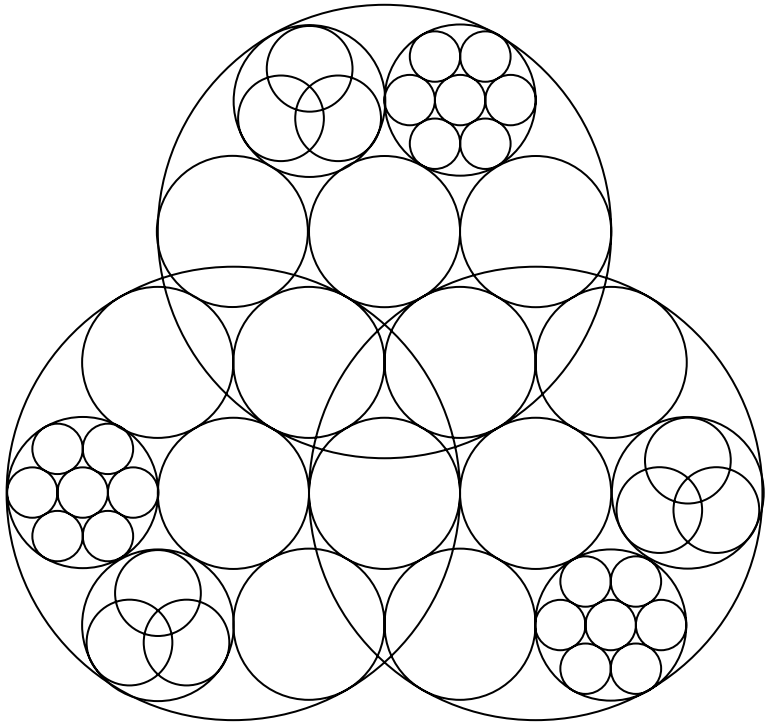
Second, you may find your question answered among Frequently Asked Questions, accessible at http://www.applied-ml.com/faq.

Third, fellow aML users in the aML users forum may shed light on the issue. The users forum communicates through an electronic mailing listserver. You may join the group by following the simple instructions on http://www.applied-ml.com/listserver. The users forum subscription is absolutely free and open to anyone. We encourage you to participate in this

exchange as much as possible, and actively monitor and contribute ourselves. The users forum is also the place to submit questions on statistical modeling and approach which are not specifically related to the aML software.

Fourth, you may contact us directly for technical support by sending e-mail to <support@applied-ml.com>. Be sure to include the serial number of your product and a telephone number where we may reach you. Also, if relevant, attach control and/or output files that illustrate the issue.

Multilevel Multiprocess Modeling



aML

USER'S GUIDE

1. Roadmap

The User's Guide is intended for both beginning and more experienced aML users. It starts by explaining the basic concepts and philosophy, and introduces many of aML's more advanced features toward the end. It contains the following chapters.

This chapter explains the entire process of model estimation, from data preparation to interpretation of the results. It only covers single-level, single-process models of the basic types: probit, logit, continuous, hazard, binomial, Poisson, negative binomial, ordered probit, ordered logit, tobit, multinomial logit, and multinomial probit. Beginners should read Section 2.1 to get familiar with aML, and may then skip to the section corresponding to the type of model of interest.

This chapter explains how to prepare multilevel and multiprocess data. A thorough understanding of the way data are structured is essential to the successful use of aML's full capabilities.

This chapter introduces multilevel and multiprocess models. It explains how to specify heterogeneity and how to make sure that residuals are correlated across equations.

This chapter illustrates some of the more exciting models that aML is capable of estimating: extensions of probit and ordered probit models, multilevel Heckman selection models, random effects models, models with heteroskedasticity, errors in variables, hazard models with multiple duration dependencies, et cetera. Each is an illustration of how basic features which may be combined to specify advanced models.

This chapter provides hints on the specification of good starting values. The more complex the model, with multiple levels and multiple simultaneous processes, the more important it is to specify good starting values.

2. Single-Level Single-Process Models

This Chapter provides sample programs for all types of models that are handled by aML: (ordered) probit/logit, continuous, hazard, and (negative) binomial. Only the basic model formulations are described, with one data level and one process (type of model) at a time. Chapter 4 discusses models with multiple levels and/or multiple processes. We strongly recommend that you, at a minimum, read Section 2.1. It describes the entire sequence from data preparation to interpretation of the results of a simple probit model. In addition, you should read the section(s) related to the type(s) of model you wish to estimate:

- Section 2.2 for logistic regression (logit) models;
- Section 2.3 for continuous models;
- Section 2.4 for hazard models;
- Section 2.5 for binomial models;
- Section 2.6 for Poisson models;
- Section 2.7 for negative binomial models;
- Section 2.8 for ordered probit and logit models;
- Section 2.9 for tobit models;
- Section 2.10 for multinomial logit models; and
- Section 2.11 for multinomial probit models..

The procedure for other types of models is much the same as that for the probit model, and to avoid duplication, numerous references are made to Section 2.1.

All data and control files used in this chapter may be found in the "`Samples\Chapter2`" directory.

2.1. Probit Model

This section explains how to estimate a simple probit model. Even if you are not interested in probit models, we urge you to read on, as most of this section applies equally to other types of models.

aML offers only limited data manipulation capabilities. Most data preparation is therefore performed using a third-party software package, and aML serves to estimate model parameters only. The main steps are:

- Prepare your data using a third-party software package (SAS, Stata, SPSS, or other). Data preparation involves sample selection, outcome measurement, and selection of potential explanatory variables. Many software packages drop all observations with any missing model variable from model estimation; aML does not accept missing values at all.[2]
- Write out your data into standard ASCII (raw, text) format. In SAS this involves the "`put`" command; in Stata the "`outfile`" command.
- Convert the raw data into aML format using executable program raw2aml (included with the aML package).
- Specify and estimate your model using executable program aML.

We now illustrate each of these four steps, starting with data preparation. We assume that the sample has been selected and that missing values have been resolved.

2.1.1. Data Preparation

The steps to estimate a probit model are best illustrated with an example. Suppose we have data on the educational attainment of 471 individuals. We would like to determine what the effects are of family background and other covariates on the probability of high school graduation. The (SAS, Stata, SPSS) data contain the following variables:

`id`	respondent identification number
`educ`	respondent education: 1 = high school drop-out 2 = high school graduate 3 = college graduate
`female`	indicator variable for female (0 = male; 1 = female)
`birth18`	indicator variable for whether the respondent gave birth at or before age 18
`dadeduc`	education of respondent's father (same codes as educ)

[2] This does not mean that you cannot use aML if your data contain missing values. Instead, all missing values must thus be resolved, for example by creating indicator variables that flag missing values and setting missing values equal to the mean over nonmissing values. See Section 3.5 for more on missing data.

momeduc education of respondent's mother (same codes as educ)
poorkid indicator variable for whether the respondent grew up in poverty

Our sample data, as most data collected through a survey, contain a unique identifier (ID) for each person. If such a variable does not exist, you must create one.

aML supports multilevel multiprocess modeling with potentially many replications of outcomes and explanatory variables. It is therefore necessary to tell aML which outcomes and explanatory covariates belong together, i.e., which are part of a particular observation. This is done by assigning an ID to all records containing outcomes and explanatory covariates. All records with the same ID are assumed to belong to the same observation. IDs must be strictly positive integers.

We wish to analyze high school graduations using a probit model. The outcome should be a binary (0/1) variable, but the data contain educational attainment as a categorical variable. We therefore create a new variable, say, HSgrad, which is equal to 1 if educ equals 2 or 3, and 0 otherwise.

The explanatory variables include respondent characteristics (sex and, if female, whether she gave birth to her first child at or before age 18) and family background characteristics (parental education and whether the respondent lived in poverty as a child). Parental education is measured in the same way as the respondent's education. We create indicator variables for whether the father did not complete high school (dadltHS) or completed college (dadcoll), and similar variables for maternal education (momltHS, momcoll). The data are now ready for analysis.

2.1.2. Write Out the Data in ASCII Format

We now write out the data in ASCII (raw, text, human-readable) format. In SAS, this may be done as follows:

```
data _null_;
   set dataname;
   file 'education.raw';
   put id educ HSgrad female birth18 dadeduc momeduc
       dadltHS dadcoll momltHS momcoll poorkid;
```

In Stata, the same may be achieved using the outfile statement: [3]

[3] The "#delimit" statement tells Stata to keep parsing the command until a semicolon is encountered. It allows commands to occupy multiple lines. By default, Stata separates data fields by multiple blank spaces, resulting in ASCII data files that are much larger than they need to be. The "comma" option eliminates those blank spaces and separates the numbers by single commas. By default, lines are wrapped at 80 columns; the "wide" option keeps all numbers on one line. This tends to be helpful in case something goes wrong and you need to visually inspect the ASCII file.

```
#delimit ;
outfile id educ HSgrad female birth18 dadeduc momeduc
   dadltHS dadcoll momltHS momcoll poorkid
   using education.raw, comma wide;
```

The order in which variables are written out is important. The first variable must always be the ID. The remainder of the variables in this simple example may be in any order, but there are rules for more complicated multilevel and/or multiprocess data, as explained in Chapter 3. To illustrate, the first five rows of the (SAS-generated) file are:

```
6 2 2 1 1 0 0 1 0 1 0 0
10 2 2 2 1 0 0 0 0 1 1 0
13 2 2 2 1 0 0 0 0 1 0 0
17 1 2 2 0 0 0 0 0 1 1 1
18 3 3 2 1 0 1 0 0 0 0 0
```

This file, "`education.raw`", and all other files that are named throughout this User's Guide are located in the "`Samples`" directory with which your copy of aML was distributed. Files related to the current section are in subdirectory "`Samples\Section2`".

2.1.3. Conversion into aML Format Using Raw2aml

The raw ASCII data now need to be converted into an aML-suitable format. This is done by using raw2aml, an important program that is included in the aML package. The inputs to raw2aml are the ASCII data file and a control file which specifies the input data and variable names. Multilevel data and data pertaining to multiple processes require additional information in the raw2aml control file, as explained in Chapter 3. We recommend extension "`.r2a`" for the raw2aml control file. The control file may be created using any text editor (page 3). The sample raw2aml control file (`education.r2a`) is:

```
1  ascii data file = education.raw;
2
3  var = educ HSgrad female birth18 dadeduc momeduc
4        dadltHS dadcoll momltHS momcoll poorkid;
```

Note that we added line numbers to facilitate the discussion; they are not part of the control file. The raw2aml control file consists of a number of statements. All statements must be terminated by a semicolon.

Blank lines may be inserted freely throughout the control file. The control file is parsed in free format, i.e., all statements may span multiple lines and multiple statements may appear on one line. The maximum line length, however, is restricted to 80 columns. C-style comments may be inserted freely, as explained below. We now discuss the two statements.

```
ascii data file = education.raw;
```

This statement specifies the raw data input file. It is assumed to be in the current working directory; specify a (disk drive and) path name if needed. If the data file name contains spaces or other special characters, it must be enclosed by single or double quotes.

```
var = educ dadeduc momeduc HSgrad dadltHS dadcoll momltHS
     momcoll poorkid female birth18;
```

This statement specifies the variable names. Variable names may contain up to eight characters and are case sensitive. Variable names may contain alphanumeric characters and underscores ('_'), except that the first character may not be an underscore. (Variable names starting with an underscore are reserved for use by aML.)

You may assign any names to the variables. To minimize confusion, it is a good idea to use the same names as in your original (SAS, Stata, SPSS) data set, but you do not need to.

Raw2aml and aML are not case-sensitive when interpreting keywords. All names that are specified by the user, however, are case-sensitive. In other words, "`var = `" and "`VAR = `" are equivalent, but "`VAR = EDUC DADEDUC MOMEDUC`" makes the variable names uppercase. When using those variables later, in estimating models, they need to be specified with the same case. On UNIX and Linux platforms, file names are also case-sensitive; on PCs they are not.

Tip: Getting help

Raw2aml and aML do not come with online documentation. The best way to get quick help on the syntax of any statement is to make some obvious error in the control file. Raw2aml and aML will try to provide an informative error message, often with a statement of the expected syntax.

Raw2aml and aML must be executed from a DOS or UNIX/Linux window (page 3). From the command prompt, type the following to convert the raw data (`education.raw`) into an aML-compatible data file (`education.dat`) using control file `education.r2a`:

```
raw2aml education
```

Raw2aml expects the name of a control file as an argument. It assumes extension "`.r2a`", so the above command has the same effect as "`raw2aml education.r2a`".

By default, the output data set in aML-format has the same name as the control file, but with extension "`.dat`" rather than "`.r2a`". You may override this default using command line option "`-o`" (page 221) or "`option output data file`" in the control file (page 228). In the example, raw2aml creates "`education.dat`". In addition, raw2aml generates a second output file, "`education.sum`", with summary information on the data file. That summary information

is also written out to standard output (the window or screen). In the example, summary information file "`education.sum`" contains the following:

```
 1  Documentation for 'education.dat'
 2  Created on Mon Dec  9 13:20:36 2002 with raw2aml version 2.00.
 3  Ascii data set: 'education.raw'
 4
 5  Number of observations:     471
 6
 7  ------------------------------------------------------------
 8
 9  LEVEL 1 VARIABLES:
10  Variable    N       Mean     Std Dev        Min        Max
11  _id       471   576.2038   329.2789        6.0     1149.0
12  educ      471   2.106157   .6567038        1.0        3.0
13  HSgrad    471   .8322718   0.374022        0.0        1.0
14  age       471   44.04034   14.60066       18.0       69.0
15  female    471   .5095541   .5004402        0.0        1.0
16  birth18   471   .1146497   .3189375        0.0        1.0
17  dadeduc   471   1.804671   0.746666        1.0        3.0
18  momeduc   471   1.660297   .7080804        1.0        3.0
19  dadltHS   471   .3949045   .4893499        0.0        1.0
20  dadcoll   471   .1995754   .4001061        0.0        1.0
21  momltHS   471   0.477707   .5000339        0.0        1.0
22  momcoll   471   .1380042   .3452712        0.0        1.0
23  poorkid   471   .2208068   .4152315        0.0        1.0
24  income    471   1379.299    2437.96       10.0    25950.0
25
26  ------------------------------------------------------------
27
28  NOTE: there is variation in all data variables.
```

It is very important to carefully study this documentation file and check that it conforms to what is expected. For instance, check that the mean, the minimum, the maximum of the variables are correct, check that the number of observations is correct, and pay attention to any notes and warnings.

Note that all variables are "`level 1 variables`"; data with multiple levels are discussed in Chapter 3. Also note that variable "`_id`" was added to the list. It was created by raw2aml and is always equal to the observation ID. You may use it in model specifications just like any other variable.

2.1.4. Model Specification and Estimation

The propensity (index) function of a standard probit model may be denoted by:

$$y_i^* = \beta' x_i + u_i ,$$

where we normalize $u_i \sim N(0,1)$. Subscript *i* denotes the observation number. Throughout this manual, we tend to suppress the observation subscript and only indicate replications within

observations. Re-write the model therefore as $y^* = \beta' x + u$. Outcome y depends on the value of y^* relative to an implicitly defined zero-threshold:

$$y = \begin{cases} 0 & \text{if } y^* < 0 \\ 1 & \text{if } y^* \geq 0 \end{cases}$$

aML is capable of estimating far more general probit models. Residuals do not need to have unit variance, the threshold may be an estimable parameter, there may be multiple residuals, et cetera. The example only serves to illustrate the steps you need to take to estimate any type of model.

The log-likelihood of one observation is given by:

$$\ln L = \begin{cases} 1 - \Phi(\beta' x / \sigma_u) & \text{if } y = 0; \\ \Phi(\beta' x / \sigma_u) & \text{if } y = 1. \end{cases}$$

where $\Phi(\cdot)$ denotes the cumulative normal probability function.

The model specification is communicated to aML through a control file, much like control files used by raw2aml. We recommend extension `.aml` for aML control files. File `educ1.aml` specifies the model and initializes the parameters:

```
 1  /* Specify the data set as converted by raw2aml */
 2  dsn = education.dat;
 3
 4  /* Specify regressors */
 5  define regressor set BetaX;
 6     var = 1 female birth18 dadltHS dadcoll momltHS momcoll poorkid;
 7
 8  /* Specify the model:  outcome and right-hand-side */
 9  probit model;
10     outcome = HSgrad;
11     model = regset BetaX;
12
13  starting values;
14
15  Constant    T    0
16  dadltHS     T    0
17  dadcoll     T    0
18  momltHS     T    0
19  momcoll     T    0
20  poorkid     T    0
21  female      T    0
22  birth18     T    0
23  ;
```

The control file consists of three parts. The first specifies the name of the data set and a number of options, if desired. No such options are present in this example. Note the text surrounded by `/* ... */`; these are comments, not statements, and will be ignored (see Section 16.1 for details). The second part defines "building blocks" of models and specifies the model(s).

Building blocks represent anything on the right-hand-side of a model equation: regressors, residuals, et cetera. In this example, only a "regressor set" is defined and one probit model specified. The third part initializes parameter values for the initial round of optimization.

Each statement (terminated by a semicolon, as before) is now discussed in turn.

```
dsn = education.dat;
```

The "dsn" (data set name) statement specifies the data file to be used. Specifying extension ".dat" is optional.

```
define regressor set BetaX;
   var = 1 female birth18 dadltHS dadcoll momltHS momcoll poorkid;
```

This statement defines a "regressor set." Unlike most other software packages, aML requires that regressors, residuals, and other explanatory elements (collectively referred to as building blocks) are defined up-front. Among such building blocks are regressor sets, best interpreted as linear combinations of explanatory variables, e.g., $\beta' X$. Here we define a regressor set and assign it the name "BetaX". Names of building blocks may be up to twelve characters in length. They are user-defined and thus case-sensitive. Our regressor set contains a "1" and seven variables. Mathematically, this regressor set represents:

$$\begin{aligned}\text{BetaX} = \; & \beta_0 \cdot 1 + \beta_1 \cdot \text{female} + \beta_2 \cdot \text{birth18} + \beta_3 \cdot \text{dadltHS} + \\ & \beta_4 \cdot \text{dadcoll} + \beta_5 \cdot \text{momltHS} + \beta_6 \cdot \text{momcoll} + \beta_7 \cdot \text{poorkid}\end{aligned}$$

The "1" is always equal to one and therefore generates an intercept, β_0. The other seven variables generate one parameter each, β_1 through β_7, which we want to estimate.

Unlike most other statistical packages, aML does not assume an intercept in any model. You must specify it explicitly.

```
probit model;
```

This statement indicates that we wish to estimate a probit model. The model consists of an outcome and the right-hand-side of the model equation with explanatory covariates and residuals.

```
outcome = HSgrad;
```

This statement specifies that HSgrad is the outcome variable. aML will check that this variable is binary (0 or 1); other values result in a fatal error (with an informative message).

```
model = regset BetaX;
```

The model statement specifies the right-hand-side (explanatory part) of the model to be estimated. In this example, only covariates in regressor set `BetaX` enter the model. Instead of specifying

```
model = regressor set BetaX;
```

we used abbreviation `regset` for regressor set; both forms are accepted. (aML accepts abbreviations for most key words; see the Reference Manual for allowable syntax.) Note that we defined the regressor set as `BetaX` and referred to it in the same way. Its name is user-defined and thus case-sensitive. Model specifications with `betax`, `BETAX`, et cetera, result in a fatal error.

Our model is very simple and only contains one building block. Strictly speaking, of course, the model also contains a residual; we return to this issue below. Many more model statements below show how to specify more complicated models.

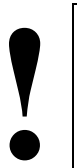
```
starting values;

Constant    T    0
female      T    0
birth18     T    0
dadltHS     T    0
dadcoll     T    0
momltHS     T    0
momcoll     T    0
poorkid     T    0
;
```

aML uses full information maximum likelihood with an iterative search algorithm to find parameter estimates. It requires that the user specifies starting values for (almost) all parameters, i.e., values which are used in the first iteration. Selection of good starting values is somewhat of an art; see Chapter 6 for its fundamentals.

Good starting values are essential for successful model estimation. The importance of understanding and specifying sensible starting values can hardly be overstated.

Our model contains eight parameters (β_0 through β_7) which are associated with eight variables or expressions in regressor set `BetaX`. The user may select any name for these eight parameters. We chose "`Constant`" and names that correspond to variable names, but we could have picked "`beta0`", "`beta1`", et cetera, or any other names. Any character except blank spaces and tabs is allowed, including, for example, "`dad<HS`". Parameter names may be up to eight characters in length.



There is an important conceptual distinction between names in the starting values statement and those in the variable list of a regressor set definition. The variable list in a regressor set contains names of data variables (x). The names in the starting values statement are names of model parameters (β, but also σ_u et cetera).

Not all model parameters need always be estimated. In fact, sometimes a model would not be identified without restrictions on parameter values. Also, it is often a good idea to estimate models in two stages: first we optimize over the intercept only, and subsequently all parameters are estimated. You may specify which parameters are estimated and which are fixed at their starting value. In our example, each starting value name is followed by a '`T`' (short for 'true'), telling aML to find the optimal value for this parameter. Alternatively, an '`F`' (short for 'false') would fix the parameter to its starting value. Think of these `T`s and `F`s as referring to the presumption that a parameter is estimated—true or false?

The model is estimated by running aML from a DOS or UNIX/Linux prompt:

```
aml educ1
```

aML expects the name of a control file as an argument. There is no need to specify default extension "`.aml`". By default, aML creates an output file with the same name as the control file, but with extension "`.out`" rather than "`.aml`". You may override this default by using command line option "`-o`" (see page 264). The output is also written to standard output (the window or screen). The output file, "`educ1.out`", contains the following:

```
 1                      +------------------------------+
 2                      |  aML version:    2.00        |
 3                      |  Serial number:  D4711010    |
 4                      |  Licensed to:    Stan Panis  |
 5                      |                  EconWare    |
 6                      |  Standard license, 10 users  |
 7                      +------------------------------+
 8
 9  Start of program:  Mon Dec  9 13:44:40 2002
10  Control file:      educ1.aml
11  Input data file:   education.dat
12       Created on:   Mon Dec  9 13:20:36 2002
13       Created by:   raw2aml version 2.00
14
15  Converge if wgn < .1
16
17  Note: the number of observations is 471; they generated 471 outcomes.
18
19  All observations are equally weighted.
20
21  The following regressor sets have been defined:
22
23  define regressor set BetaX;
24     var = 1 female birth18 dadltHS dadcoll momltHS momcoll poorkid;
```

```
25
26      ---------+----------------  Summary statistics  ----------------
27      name     |      #         Mean     Std Dev         Min          Max
28      ---------+-----------------------------------------------------------
29      Constant |    471          1.0         0.0         1.0          1.0
30      female   |    471    .5095541    .5004402         0.0          1.0
31      birth18  |    471    .1146497    .3189375         0.0          1.0
32      dadltHS  |    471    .3949045    .4893499         0.0          1.0
33      dadcoll  |    471    .1995754    .4001061         0.0          1.0
34      momltHS  |    471    0.477707    .5000339         0.0          1.0
35      momcoll  |    471    .1380042    .3452712         0.0          1.0
36      poorkid  |    471    .2208068    .4152315         0.0          1.0
37
38
39    The following models have been specified:
40
41    probit model;
42       outcome = HSgrad;
43       model = regset BetaX +
44          res(draw=_iid, ref=N(0,1))
45          ;
46
47       Summary statistics of the outcome and selected variables:
48
49           outcome |       Freq.    Percent
50       ------------+------------------------
51                 0 |         79       16.77
52                 1 |        392       83.23
53       ------------+------------------------
54             Total |        471      100.00
55
56    ============================================================================
57
58    Number of parameters in model:        8
59    Number of parameters estimated:       8
60
61    Starting values:
62         Name      Est?     Value
63      1  Constant   T      0.000000
64      2  female     T      0.000000
65      3  birth18    T      0.000000
66      4  dadltHS    T      0.000000
67      5  dadcoll    T      0.000000
68      6  momltHS    T      0.000000
69      7  momcoll    T      0.000000
70      8  poorkid    T      0.000000
71
72    ============================================================================
73    =                       RESULTS OF OPTIMIZATION                            =
74    ============================================================================
75
76
77    ITERATION 1            LOG-LIKELIHOOD:  -326.472322
78
79    PARAMETER        VALUE     GRADIENT   SEARCH DIR
```

```
Constant            0.0     249.7379     1.326318
female              0.0     111.7038     .0017991
birth18             0.0    -1.595769    -.9876018
dadltHS             0.0     55.85192    -.4823325
dadcoll             0.0     68.61807     .0673997
momltHS             0.0     93.35249    -.1984345
momcoll             0.0     50.26673     .0573285
poorkid             0.0     30.31961    -.5308673

SMALLEST EIGENVALUES:
     21.17582     24.1404    30.56209    47.47524    57.86358

REL PARAM CHG:  Infinity      WGTD GRAD NORM:  16.70198
     GRAD NORM:  307.9576       REL LN-L IMPR:       N/A

SEARCHING TO IMPROVE FUNCTION VALUE (OLD LN-L =     -326.472322):
                    STEPSIZE:  1       NEW LN-L =     -166.830419
                    STEPSIZE:  2       NEW LN-L =     -157.982869

--------------------------------------------------------------------------------

ITERATION 2             LOG-LIKELIHOOD:  -157.982869
                        ABSOLUTE IMPROVEMENT:  168.489453

PARAMETER         VALUE     GRADIENT   SEARCH DIR
Constant       2.652637    -44.14336    -.3379006
female         .0035983    -16.90111     .0274338
birth18       -1.975204     .1351667     .2784378
dadltHS       -.9646651    -32.05151     .0189694
dadcoll        .1347995    -.7763404     .0886933
momltHS       -.3968691    -32.07908     .0153044
momcoll        0.114657     .9670044     .5300022
poorkid       -1.061735    -15.84378    -.0378441

SMALLEST EIGENVALUES:
     3.893571    6.715957    9.993674    19.70316    36.01714

REL PARAM CHG:  7.624181      WGTD GRAD NORM:  3.799256
     GRAD NORM:   67.4035       REL LN-L IMPR:  .5160911

SEARCHING TO IMPROVE FUNCTION VALUE (OLD LN-L =     -157.982869):
                    STEPSIZE:  1       NEW LN-L =     -148.492579
                    STEPSIZE:  2       NEW LN-L =     -149.811800

--------------------------------------------------------------------------------

ITERATION 3             LOG-LIKELIHOOD:  -148.492579
                        ABSOLUTE IMPROVEMENT:  9.4902905E+00

PARAMETER         VALUE     GRADIENT   SEARCH DIR
Constant       2.314736    -15.07696    -.1672268
female         0.031032    -5.533966     .0059886
birth18       -1.696766    -.3710752     .1228115
dadltHS       -.9456957    -10.26931     .0292697
dadcoll        .2234928    -.0709573     .0546473
```

```
momltHS      -.3815647    -11.51978    -.0041862
momcoll       .6446592    -.5013509    -.0005105
poorkid      -1.099579    -3.450812     .0700738

SMALLEST EIGENVALUES:
       3.5673    6.367475    9.205457    17.41933    30.66419

REL PARAM CHG:  0.244515      WGTD GRAD NORM:  1.394551
    GRAD NORM:  22.54785       REL LN-L IMPR:  .0600716

SEARCHING TO IMPROVE FUNCTION VALUE (OLD LN-L =     -148.492579):
                  STEPSIZE:  1       NEW LN-L =     -147.398819
                  STEPSIZE:  2       NEW LN-L =     -148.129171

--------------------------------------------------------------------------

ITERATION 4            LOG-LIKELIHOOD:  -147.398819
                       ABSOLUTE IMPROVEMENT:  1.0937598E+00

PARAMETER        VALUE     GRADIENT   SEARCH DIR
Constant      2.147509    -1.460431     .0215061
female         .0370206     .0149266     .0225124
birth18      -1.573954    -.0082149    -.0242349
dadltHS      -0.916426    -1.303831    -.0106711
dadcoll       .2781401      0.43008     .0253557
momltHS      -.3857509    -2.299784    -0.044566
momcoll       .6441487    -.1524645    -.0457431
poorkid      -1.029505    -.3623104    -0.007604

SMALLEST EIGENVALUES:
       3.3725    6.191245    9.076885    16.63573     29.0509

REL PARAM CHG:  .6081051      WGTD GRAD NORM:  .3258322
    GRAD NORM:  3.075974       REL LN-L IMPR:  .0073658

SEARCHING TO IMPROVE FUNCTION VALUE (OLD LN-L =     -147.398819):
                  STEPSIZE:  1       NEW LN-L =     -147.345206
                  STEPSIZE:  2       NEW LN-L =     -147.395925

--------------------------------------------------------------------------

ITERATION 5            LOG-LIKELIHOOD:  -147.345206
                       ABSOLUTE IMPROVEMENT:  5.3612662E-02

PARAMETER        VALUE     GRADIENT   SEARCH DIR
Constant      2.169015    -.4236185    -.0190039
female        0.059533    -.1142998    -0.006433
birth18      -1.598189     .1375638     .0190104
dadltHS      -.9270972    -.1274416     .0109889
dadcoll       .3034959     0.045473     .0125469
momltHS      -.4303169    -.2230889     0.006256
momcoll       .5984055    -.0795033    -0.009026
poorkid      -1.037109    -.0239648     .0112697

SMALLEST EIGENVALUES:
```

```
190       3.338989    6.200334    9.224347    16.66032    28.93176
191
192   REL PARAM CHG:  .1080579      WGTD GRAD NORM:  0.098096
193       GRAD NORM:  .5351764       REL LN-L IMPR:  .0003637
194
195
196   ================================================================================
197   =                  ESTIMATION CONVERGED SUCCESSFULLY                           =
198   =                        RESULTS OF ESTIMATION                                 =
199   ================================================================================
200
201   Convergence based on:
202      Weighted gradient norm:         .098096 < .1
203      Relative function improvement:  .0003637
204      Gradient norm:                  .5351764
205      Relative parameter change:      .1080579
206
207   ================================================================================
208
209   Log Likelihood:  -147.3452
210                                                BHHH-based, non-corrected
211     Parameter    Free?     Estimate          Std Err        T-statistic
212
213     1  Constant    T     2.1690153696     .26482410439        8.1904
214     2  female      T      .05953298334    .20359465728        0.2924
215     3  birth18     T    -1.5981892361     .26934305816       -5.9337
216     4  dadltHS     T    -.92709717231     .20713749611       -4.4758
217     5  dadcoll     T     .30349586089     .37146558685        0.8170
218     6  momltHS     T    -.43031691961     .19506406359       -2.2060
219     7  momcoll     T     .59840551662     .51691293565        1.1577
220     8  poorkid     T    -1.0371088315     .19918910979       -5.2067
221
222   ================================================================================
223
224   Elapsed clock time is 0 seconds.
```

Line numbers are not part of the output file, but were added to facilitate discussion. We now briefly discuss the output.

Lines 1-7 indicate the version of aML which estimated the model and writes out license information.

Lines 9-13 document the time of execution, the name of the control file, and the name, creation date, and raw2aml version of the data set.

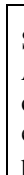
Line 15 states the convergence criterion. By default, the iterative search terminates when the weighted gradient norm (wgn) is less than 0.1.[4] There are several options to select alternative

[4] The underlying idea of a weighted gradient norm is to stop searching when the parameter changes are sufficiently small. The weight applied to each parameter is inversely related to the precision with which that parameter is estimated. The Gauss-Newton search direction is given by (Judge et al., 1985):

convergence criteria (such as unweighted gradient norm, relative function improvement, and relative parameter change) and alternative convergence thresholds. See "`option converge`" on page 276.

Lines 17-19 report on the number of observations, the number of outcomes, and estimation weights, if any. By default, all observations are weighted equally. aML also supports weighted optimization. See "`option weight`" and "`option normalized weight`" on page 274.

Lines 21-36 restate the definitions of building blocks. In this example, regressor set `BetaX` is the only building block. The output file feeds back the names of corresponding parameter names (the names the user assigned in the list of starting values), and provides summary statistics on the regressors.



Starting values must be assigned in the same order as building blocks were defined. Adding and deleting building blocks from control files must thus be accompanied by corresponding changes to the list of starting values. This creates potential for user error. The output file feeds back the parameter names so that the user may check that parameters refer to the intended building blocks.

Lines 39-45 restate the model specification. Note that there is one element which we did not specify in the control file, a residual (abbreviated to "`res`"):

```
res(draw=_iid, ref=N(0,1))
```

Below we will discuss the syntax for specifying residuals. Suffice it here to say that line 44 specifies a residual which is drawn independently, and that the residual is distributed normally with zero mean and unit standard deviation. This residual, capturing u above, is part of any probit model. Its standard deviation is not identified in this simple model. aML therefore does not require an explicit specification of this residual; it assumes a standard normally distributed residual by default. (You may, of course, specify it explicitly; see Section 13.8) To remind you that it is really there, aML explicitly writes that default residual in the output file.

$$\delta = -\left(\frac{\partial^2 \ln L}{\partial\theta\partial\theta'}\right)^{-1}\left(\frac{\partial \ln L}{\partial\theta}\right)$$

We weight the length of the search direction vector by the inverse of the covariance matrix of parameter estimates, $\Sigma_{\hat{\theta}\hat{\theta}}$, which may be approximated by minus the inverse of the matrix of second derivatives:

$$\begin{aligned}\text{wgn} = \sqrt{\delta\Sigma_{\hat{\theta}\hat{\theta}}^{-1}\delta} &= \sqrt{\left(\frac{\partial \ln L}{\partial\theta}\right)'\left(\frac{\partial^2 \ln L}{\partial\theta\partial\theta'}\right)^{-1}\left(\left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'}\right)^{-1}\right)^{-1}\left(\frac{\partial^2 \ln L}{\partial\theta\partial\theta'}\right)^{-1}\left(\frac{\partial \ln L}{\partial\theta}\right)} \\ &= \sqrt{\left(\frac{\partial \ln L}{\partial\theta}\right)'\left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'}\right)^{-1}\left(\frac{\partial \ln L}{\partial\theta}\right)}.\end{aligned}$$

Lines 49-54 provides a tabulation of the model's dependent variable, for your information.

Lines 58-59 state that there are eight parameters in this model, and that all eight are estimated. Not all parameters need to be estimated, as explained above. Some parameters may be fixed to their initial value (in the starting values statement), so that optimization is restricted.

Lines 61-70 restate the starting values of this run.

The remainder of the output file provides information on the iterative search procedure and the parameter estimates.

Lines 77-98 show information related to the initial iteration. At the initial parameter values that the user supplied, the log-likelihood is -326.472322 (line 77). Lines 80-87 show the current iteration's parameter values, derivatives of the log-likelihood with respect to the parameters, and the search direction. The search algorithm is Gauss-Newton (Judge et al., 1985), i.e., the search direction is given by minus the product of the inverse matrix of second derivatives of the log-likelihood (Hessian matrix) and the vector of its first derivatives (gradient).

Line 90 shows (up to five) eigenvalues of the Hessian matrix. (Strictly speaking, the eigenvalues of the Hessian matrix are all negative, so that aML reports their opposites.) If any of these eigenvalues were zero or close to zero, the model would be underidentified and you should probably reformulate or re-evaluate the model in some way. (This would be the case, for example, if a threshold was being estimated in addition to an intercept, or a standard deviation of the residual, σ_u.) No such problem exists here.

Lines 92-93 show the current iteration's values of metrics which may be used to decide on convergence. Only the weighted gradient norm is relevant in this run. In the first iteration it is 16.70198, which exceeds the (default) threshold of 0.1. The program will thus continue searching. Convergence criteria may alternatively be based on the relative parameter change, gradient norm, and the relative function improvement. See "`option converge`" on page 276 for more details and definitions.

Lines 95-97 report the log-likelihood as the program takes increasingly large steps in the search direction. At stepsize 1, the log-likelihood improves from about –326 to –167, et cetera. (At stepsize λ, parameters are equal to their values at the beginning of the current iteration plus λ times the search direction. Parameter `Constant`, for example, equals about $0.0 + 1.326 = 1.326$ at stepsize 1 and $0.0 + 2 * 1.326 = 2.652$ at stepsize 2.) At stepsize 1, the log-likelihood is better than the initial value. aML continues to increase the stepsize until the log-likelihood worsens. The log-likelihood at stepsize 2 is better than at stepsize 1, so aML could try stepsize 4. However, by default in most models, aML stops at stepsize 2. (You may change that; see "`option step range`" on page 275.) For the next iteration, parameter values are set to their "best" value, i.e., initial values plus twice the search direction in this example.

Lines 101-122 provide information on the second iteration. The best log-likelihood found in the first iteration was -157.982869 at stepsize 2 (line 97). That log-likelihood is found back on line 101; line 102 shows the absolute improvement in the log-likelihood function. Stepsize 2

implies that the current values of parameters are equal to the initial values plus twice the search direction, as found on lines 105-112.

The weighted gradient norm (line 117) is 3.799256; better than before, but still not low enough to trigger convergence.

As shown on lines 120-122, the best log-likelihood was found at stepsize 1. Parameter values carried over into the third iteration are thus equal to iteration 2's values plus the search direction.

Lines 126-193 show the results for iterations 3, 4, and 5. After evaluating the log-likelihood (and derivatives) at iteration 5's parameter values, the program found a weighted gradient norm of 0.098096. This is below the threshold value 0.1, so convergence is achieved and no attempt is made to improve the parameter values any further.

Lines 202-205 show the values of the four potential convergence metrics with an indication of the criterion that triggered convergence (line 195).

Lines 202-213 provide the log-likelihood upon convergence with parameter estimates, standard errors of parameter estimates, and t-statistics. Line 210 notes that the standard errors and t-statistics are based on the BHHH procedure (Berndt, Hall, Hall, and Hausman, 1974) and "non-corrected."[5] For small samples, you may want to compute standard errors more accurately using "`option numerical standard errors`" (Section 13.1.5). In addition, you may compute robust (Huber-corrected) standard errors using "`option huber`" (Section 13.1.6).

Line 224, finally, reports the duration of the estimation. This example ran in "0 seconds" on a PC with a 1.7GHz Xeon processor. On UNIX machines, both elapsed clock time and actual Central Processing Unit (CPU) time are reported.

As you run `educ1` yourself, you will notice that the screen output is less detailed than that written to the output file. The level of detail to both screen and output file is under control of the user through "`option screen info level`" and "`option file info level`", respectively (page 270). Level 0 implies that nothing is written out; level 5 provides maximum detail. By default, level 3 is used for screen output and level 5 for file output. You may be

[5] The standard errors that are reported here are subject to two approximations. First, the variance-covariance matrix of parameter estimates is approximately equal to minus the inverse of the Hessian matrix (matrix of second derivatives). Second, by default aML approximates the Hessian matrix as minus the sum over individual observations of the outerproduct of first derivatives (Berndt, Hall, Hall, and Hausman, 1974):

$$\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \approx -\sum_{i=1}^{n} \frac{\partial \ln L_i}{\partial\theta} \frac{\partial \ln L_i}{\partial\theta'},$$

where $\ln L_i$ is the log-likelihood for the i-th observation, and n is the number of observations. The latter approximation is used throughout the optimization search to determine the search direction. (Instead, you may force aML to compute the Hessian matrix numerically, that is, as the numerical derivative of analytically computed first derivatives. See "`option numerical search`" on page 270.) Both approximations hold asymptotically, i.e., for very large samples.

tempted to reduce the output levels and just focus on the final results. We urge you to resist this temptation and instead learn about your model from intermediate results. For more detailed explanations of the informational content of eigenvalues, weighted gradient norms, stepsizes, et cetera, see Section 14.4.

2.1.5. Using Expressions Instead of Variables

In most places of the aML control file where you specify a variable name, aML accepts expressions as well. We already encountered one example in the definition of regressor set `BetaX`: instead of a variable name, we specified a simple "1". Many more expressions are allowed. For example, our original data set contained respondent and parental education variables coded as 1 (less than high school), 2 (high school graduate), and 3 (college graduate). We created indicator variables `HSgrad`, `dadltHS`, `dadcoll`, `momltHS`, and `momcoll` for use in the model. However, we could have defined the regressor set as follows:

```
define regressor set BetaX;
   var = 1 female birth18 (dadeduc==1) (dadeduc==3)
         (momeduc==1) (momeduc==3) poorkid;
```

In other words, variables are replaced by expressions. In this case, these expressions are conditions which evaluate to zero (if false) or one (if true). A condition that tests for equality requires double equality signs (==), similar to the convention in C, Stata, and other computer languages. We added parentheses around each expression to improve readability; they may be omitted.

Expressions may also be used in the outcome specification. For example, our outcome could have been specified as

```
outcome = (educ>=2);
```

The ability to define indicator and other variables in the model specification permits greater flexibility in model exploration. For example, you may experiment with alternative dummy variable definitions, and with interactions of variables. In addition, since only the underlying categorical variables need to be included in the data, the data set size may be kept relatively small.

The table below lists the operators may be used in expressions.

`exp(x)`	exponent
`int(x)`	integer portion (truncation)
`log(x)`	natural logarithm
`abs(x)`	absolute value
`sqrt(x)`	square root
`spline(x, nodes)`	piecewise-linear spline transformation (see below)
`min(x,y[,...])`	minimum
`max(x,y[,...])`	maximum

`x^y`	x to the power y
`x*y`	x times y
`x/y`	x divided by y
`x+y`	x plus y
`x-y`	x minus y
`x==y`	x equals y (evaluates to 0 if false, to 1 if true)
`x<y`	x is less than y (evaluates to 0 if false, to 1 if true)
`x<=y`	x is less than or equal to y (evaluates to 0 if false, to 1 if true)
`x>y`	x is greater than y (evaluates to 0 if false, to 1 if true)
`x>=y`	x is greater or equal to than y (evaluates to 0 if false, to 1 if true)
`x!=y`	x is not equal to y (evaluates to 0 if false, to 1 if true)
`x and y`	Boolean and (evaluates to 1 if and only if both x and y equal 1)
`x or y`	Boolean or (evaluates to 1 if x, y, or both equal 1)
`not x`	Boolean not (evaluates to 1 if x equals 0)

The expressions may also be combined as desired, and standard mathematical preference rules apply.

aML supports the piecewise-linear spline transformation. It transforms a variable into a vector of new variables. Each new variable represents the original variable on a specific segment of its range, so that the estimated effect of a variable is no longer linear, but piecewise-linear. Suppose we would like to estimate the effect of log-income (`lninc`) on some outcome of interest. We would like to explore whether the effect of income varies over its range. For example, we would like to allow for a different coefficient (slope) on log-income for each of its four quartiles. Suppose the 25-th, 50-th, and 75-th quartiles of log-income are equal to 6, 7.5, and 9, respectively. The regressor set may contain the following expression:

```
spline(lninc, 6 7.5 9)
```

This spline, with three nodes (also known as bend points or knots) translates into four new variables. The first captures the effect of log-income between $-\infty$ and 6; the second between 6 and 7.5; the third between 7.5 and 9; and the fourth between 9 and ∞. Formally, the spline transformation is:

$$\text{spline}(x,\ \nu_1\ \nu_2 \ldots \nu_n) = \begin{pmatrix} \min[x, \nu_1] \\ \max[0, \min[x-\nu_1, \nu_2-\nu_1]] \\ \vdots \\ \max[0, x-\nu_n] \end{pmatrix},$$

where $\nu_1\ \nu_2\ \ldots \nu_n$, denote the nodes. In aML's implementation, spline coefficients may thus be directly interpreted as slope coefficients, not as marginal slopes. See Panis (1994) for additional detail on piecewise-linear splines.

Expressions may also be nested. For example, we could write

```
spline(log(income), 6 7.5 9)
```

where "income" is a data variable, or even

```
spline(log(income+sqrt(1+income^2)), 6 7.5 9)
```

(This transformation, the inverse hyperbolic sine, is similar to the logarithm but allows for negative input values.) Expressions may involve any number of variables.

2.1.6. Starting Values

For lack of any better ideas, we set all starting values to zero in our example. The program converged after just five iterations, but there are many models where convergence would have been much more difficult to achieve. As a general rule, we suggest that you first let the intercept settle in before freeing up all parameters. Taking the above example, first replace the starting values by:

```
Constant    T    0
female      F    0  /* note: not estimated */
birth18     F    0  /* note: not estimated */
dadltHS     F    0  /* note: not estimated */
dadcoll     F    0  /* note: not estimated */
momltHS     F    0  /* note: not estimated */
momcoll     F    0  /* note: not estimated */
poorkid     F    0  /* note: not estimated */
```

This model converges to Constant=0.963. (We could have found this intercept value without estimation. Data summary file "education.sum" tells us that the mean value of HSgrad (mean probability, fraction successes) is 0.832. The corresponding propensity value is $\Phi^{-1}(0.832) = 0.962$, i.e., equal to the estimated constant, up to rounding error. Also see Chapter 6.) The converged value may then be used as a starting value in a run with all parameters free.

Alternatively, the two rounds may be combined in one control file:

```
Constant    TT    0
female      FT    0
birth18     FT    0
dadltHS     FT    0
dadcoll     FT    0
momltHS     FT    0
momcoll     FT    0
poorkid     FT    0
```

aML will estimate the model in multiple (here, two) rounds, fixing and freeing up parameters as specified by the Ts and Fs. These Ts and Fs must not be separated by spaces or other

characters. Combining multiple rounds of estimation in one control file is particularly useful for very large models that run overnight.

Tip: Smoother Search Paths

In cases where good starting values are not available, we suggest a first round in which only the intercept is estimated, followed by a second round where all parameters are optimized.

2.1.7. The Update Command

It is often useful to update starting values in a control file with their converged values. Auxiliary executable program "`update`" provides the easiest and quickest way. After estimating a model, say, in "`educ1.aml`", invoke update from a DOS or UNIX/Linux prompt:

```
update educ1
```

where extension "`.aml`" is by default assumed. Update reads output file "`educ1.out`", figures out the converged parameter values, and overwrites the starting values statement in "`educ1.aml`".

Alternatively, "`option starting value format`" (page 273) instructs aML to write out converged parameter values in the same format that the control file uses for starting values. You may then copy that portion of the output file and paste it back into the control file.

2.1.8. The Mktab Command

The mktab ("make tabulation") program is another executable utility that is bundled with the aML package. It reads one or more aML output files and writes parameter estimates in tabular format to standard output. We could tabulate the results in "`educ1.out`" by typing the following from a DOS or UNIX command prompt:

```
mktab educ1
```

As before, default extensions (`.out`) may be omitted.

Mktab is especially useful to compare parameter estimates in multiple output files. For example, to compare the probit results in "educ1.out" with the logit results in "educ2.out" (discussed in Section 2.2), type:

```
mktab educ1 educ2
```

The output of this command is:

```
            educ1          educ2

Constant     2.1690 ***     3.7370 ***
            (0.2648)       (0.4999)
female       0.0595         0.0648
            (0.2036)       (0.3536)
birth18     -1.5982 ***    -2.7110 ***
            (0.2693)       (0.4704)
dadltHS     -0.9271 ***    -1.6298 ***
            (0.2071)       (0.3815)
dadcoll      0.3035         0.5296
            (0.3715)       (0.6727)
momltHS     -0.4303 **     -0.7111 **
            (0.1951)       (0.3396)
momcoll      0.5984         1.3260
            (0.5169)       (1.1104)
poorkid     -1.0371 ***    -1.7676 ***
            (0.1992)       (0.3448)

ln-L         -147.35        -148.73

NOTE: Asymptotic standard errors in parentheses;
      Significance: '*'=10%;  '**'=5%;  '***'=1%.
```

Mktab features some useful command-line options, including options to include t-statistics or p-values rather than standard errors in parentheses, to change the number of digits after the decimal point, to specify stricter significance levels to the significance asterisks, to stack estimates from multiple output files into one column (rather than present them column-by-column), and to write the table out in comma-delimited format that is easily imported into spreadsheets and word processors. See Section 15.2 for details.

2.1.9. aML File Types and File Names

In the discussion above, we encountered several file types: data files, control files, and output files. aML does not require that file names have specific extensions, but we recommend that you use the default extensions to facilitate keeping track of the various files. The table below shows aML's file types and recommended file name extensions.

Extension	File type
.raw	ASCII input file. Created by a third-party data management package (SAS, Stata, SPSS) and processed by raw2aml.
.r2a	Raw2aml control file. Contains information on the number of levels in the data, variables at each levels, and more. Read by raw2aml.
.dat	Data file in aML-readable format. Produced by raw2aml, read by aML.
.sum	Documentation on the "**.**dat" data file: summary statistics on the maximum numbers of subbranches at each level, on all variables, and more. Produced by raw2aml; read by the user.
.aml	aML control file. Specifies the model to be estimated, the data on which the estimation is based, and more. Read by aML.
.out	aML output file. Provides results of estimation. Produced by aML; read by the user.

This concludes our example of a very simple probit model. Chapters 4 and 5 provide more examples in which many more probit features are illustrated, including multilevel probit models with heterogeneity, probit models with nonzero threshold and/or a residual that is not standard normally distributed, probit selection models, and ordered probit models. The remainder of the current chapter illustrates simple logit (Section 2.2), continuous (Section 2.3), hazard (Section 2.4), binomial (Section 2.5), Poisson (Section 2.6), negative binomial (Section 2.7), ordered probit and logit (Section 2.8), tobit (Section 2.9), multinomial logit (Section 2.10), and multinomial probit (Section 2.11) models.

2.2. Logit Model

This section illustrates the steps that are required to estimate a simple logit (logistic regression) model. The main steps to estimate logit models are similar or identical to those for probit models, as explained in Section 2.1. We therefore only highlight differences between estimation of probit and logit models. If you have not read Section 2.1 yet, please do so now.

The logistic regression (logit) model is very similar to the probit model. A latent propensity or index function is again given by

$$y^* = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + u$$

and outcome *y* depends on the value of y^*:

$$y = \begin{cases} 0 & \text{if } y^* < 0; \\ 1 & \text{if } y^* \geq 0, \end{cases}$$

where we suppressed the observation subscript. The difference with the probit model is that we assume $u \sim N(0,1)$ for probit models, whereas in logit models, *u* is assumed to follow the logistic distribution. The two distributions are very similar, but the logistic distribution has a larger variance and fatter tails.

Estimating a logit model is very similar to the above probit model. Consider sample control file `educ2.aml`:

```
 1  dsn = education.dat;
 2
 3  define regressor set BetaX;
 4     var = 1 female birth18 dadltHS dadcoll momltHS momcoll poorkid;
 5
 6  logit model;
 7     outcome = HSgrad;
 8     model = regset BetaX;
 9
10  starting values;
11
12  Constant    T    0
13  female      T    0
14  birth18     T    0
15  dadltHS     T    0
16  dadcoll     T    0
17  momltHS     T    0
18  momcoll     T    0
19  poorkid     T    0
20  ;
```

The only difference with `educ1.aml` is that the words "`logit model`" appear in lieu of "`probit model`". The output file is also much like "`educ1.out`". One difference is in the restatement of the model statement:

```
41  logit model;
42     outcome = HSgrad;
43     model = regset BetaX
44        ;
```

Unlike in the probit model, no implicit residual is made explicit. Logit models must always have a standard independent logistic residual; non-standard extensions are not supported.

The search procedure is similar to the probit model. Convergence is achieved after five iterations; see output file "`educ2.out`" and the mktab output shown above in Section 2.1.8. Note that the logit estimates tend to be larger in absolute value than probit estimates (due to the larger variance of its residual distribution). Parameter interpretation is roughly the same, though.

2.3. Continuous Model

This section illustrates the steps that are required to estimate a simple continuous (normal density) model. The main steps to estimate continuous models are similar or identical to those for probit models, as explained in Section 2.1. We therefore only highlight differences between estimation of probit and continuous models. If you have not read Section 2.1 yet, please do so now.

The procedure is best illustrated with an example. We use the data on educational attainment that were described in Section 2.1 and estimate a model of log-income as a function of education, sex, and age. (It is highly dubious to use these same data for analyses of educational attainment and income, but we happily ignore this issue and focus on the mechanical steps only.)

Formally, the continuous (normal density) model is given by:

$$y = \beta' x + u \,,$$

where y is the observed outcome and $u \sim N\left(0, \sigma_u^2\right)$. We suppressed the observation subscript. The log-likelihood function for one observation is:

$$\ln L = -\tfrac{1}{2}\ln\left(2\pi\sigma_u^2\right) - \tfrac{1}{2}\left(\frac{y - \beta' x}{\sigma_u}\right)^2 .$$

Simple models like these are best estimated using ordinary least squares in any statistical package; the effort of learning aML only pays off in multilevel cases, where the residual structure is more complicated, and in multiprocess cases, where continuous equations need to be estimated jointly with other continuous or qualitative equations. For expositional purposes, we briefly discuss the simplest case to get the basics out of the way. Multilevel and multiprocess extensions are discussed in subsequent sections.

Our continuous model covers very many types of models. All linear models may be estimated, including random coefficients models with any number of levels. We use the word "continuous" rather than "linear" to emphasize that the outcome is a continuous variable, as opposed to discrete or qualitative. Indeed, aML's continuous models are not restricted to linear models. For example, the model equation may contain such expressions as $\left(\beta_1' X_1\right)\left(\beta_2' X_2\right)$ and "structural" specifications such as $\lambda\left(\beta' X\right)$. There is one restriction on continuous models: (reduced form, non-integrated) residuals must be distributed normally, so that the likelihood function follows the normal density functional form. ARMA and cumulative AR(1) residuals are supported, as are finite mixture residuals (Heckman and Singer, 1984).

Model Specification and Estimation

Control file inc1.aml defines the building blocks and specifies the continuous model:

```
 1  dsn = education.dat;
 2
 3  define regressor set BetaX;
 4     var = 1 female (educ==1) (educ==3) age age*age;
 5
 6  define normal distribution; dim=1;
 7     name=u;
 8
 9  continuous model;
10     outcome = log(income);
11     model = regset BetaX +
12        res(draw=1, ref=u);
13
14  starting values;
15
16  Constant    T    6.344076  /* mean of log(income) in raw data */
17  female      T    0
18  dropout     T    0
19  college     T    0
20  age         T    0
21  age2        T    0
22  sigmau      T    1.374338  /* standard deviation of log(income) */
23  ;
```

Line 1 specifies the input data set, as created by raw2aml and discussed in Section 2.1.

Lines 3-4 define a regressor set which contains the explanatory covariates. Note that we use variable transformations. Education categories 1 and 3 denote high school drop-outs and college graduates, respectively. Age enters both linearly and in quadratic form. (We wrote `age*age` but could have used the equivalent `age^2`.)

Lines 6-7 defines a building block that we did not encounter before:

```
define normal distribution;  dim=1;
   name=u;
```

These statements define a normal distribution with dimension equal to one, i.e., a univariate normal distribution. The name of the associated residual is "`u`". Names of residuals may consist of up to twelve characters. This definition introduces one new parameter, namely the standard deviation of the residual. Distributions may have any number of dimensions; a residual name must be supplied for each dimension. See Section 13.2.6.

Lines 9–12 specify the model:

```
continuous model;
   outcome = log(income);
   model = regset BetaX +
      res(draw=1, ref=u);
```

We chose to analyze the natural logarithm of income, rather than income itself. The outcome variable is therefore "`log(income)`". Note that any expression involving any number of variables may be used to specified the outcome. The right-hand-side of the model equation is specified in the "`model=`" statement. We specify regressors through regressor set `BetaX`, similar to the specification of probit and logit models. The following is new:

```
res(draw=1, ref=u)
```

Note that "`res`" is short for "`residual`"; either may be used. The model contains a residual on the right-hand-side. Every continuous model must have at least one residual in the model specification. The residual specification contains two important pieces of information: a draw and a name.

Let's start with the name: "`ref=u`". In the example, only one distribution with one residual was defined. However, complicated multilevel, multiprocess models may contain several distributions and residuals. The "`ref=u`" (short for "`reference=u`") tells aML which residual enters the equation. (It refers to a residual, not a distribution. An *n*-variate distribution has *n* residuals. Distributions do not have names, residuals do.)

The draw specification requires that the user has a solid understanding of the model. In the current simple model, with just one outcome per observation, the residual structure is straightforward. The important issue is that the residuals are assumed to be independent across observations. That assumption is always made; if residuals are correlated across outcomes, then those outcomes should be made part of the same observation (same ID), and a more complex model arises (see Section 4.1). The draw statement specifies which residuals are correlated within each observation. Residuals that are correlated stem from the same "draw" from a distribution; residuals that are uncorrelated have different draws. In the current simple model, there is only one outcome, one residual per observation, and the draw issue does not arise. We specified "`draw=1`" to indicate that the residual comes from the first draw; we could have specified "`draw=43`" to get the forty-third draw, with identical results. (The actual draw number is completely irrelevant; aML only cares about whether draws are equal to each other.) As stated above, the draw specification specifies which residuals are correlated within each observation. Across observations, draws are always independent. It is therefore not necessary to specify different draws for different observations.

The draw specification requires a variable or expression that evaluates to a strictly positive integer. We just used "`draw=1`" here, but we could have used a variable expression, such as "`draw=educ`" or "`draw=_id`", where "`_id`" is an implicit variable equal to the observation's ID. In the current example, with one outcome and one residual per observation, all these expressions

are equivalent. See Section 4.1 for more complicated draw specifications in multilevel and multiprocess models.[6]

Lines 14-23 specify the parameter values that aML will use in its first iteration of the search process:

```
starting values;

Constant    T    6.344076  /* mean of log(income) in data */
female      T    0
dropout     T    0
college     T    0
age         T    0
age2        T    0
sigmau      T    1.374338  /* standard deviation of log(income) */
;
```

The first six parameters correspond to the regressors in regressor set "BetaX". The univariate normal distribution is fully defined by a mean and a standard deviation. Normal distributions in aML always have zero means, so that only a standard deviation is estimated.[7] Note that the distribution was defined after the regressor set. Starting values are specified in the order in which building blocks were defined.

Also note that the example was pretty smart about specifying a starting value for the intercept. When we prepared the data, we noticed that the mean of log-income was 6.344076 (not shown), so that is a good guess for the intercept of a model in which all other regressors are initialized at zero. Similarly, the standard deviation of log-income was 1.374338, so we initialize the standard

[6] For the curious reader, here is a little preview. Suppose we have a data set with multiple test scores per student. We hypothesize that there are unobservable characteristics pertaining to a student which affect all his test scores, in addition to unobservables that are more test-specific. A multilevel model may be appropriate:

$$y_{it} = \beta' X_{it} + \varepsilon_i + u_{it}$$

where y_{it} is the score of student i on a test taken at time t; ε_i captures student-specific unobserved characteristics; and u_{it} captures test-specific unobservables. All records of a single student are grouped into one observation. Residual ε_i is the same for all test scores of one student, i.e., only one draw of ε_i applies to all test scores, whereas new values for u_{it} are "drawn" for each test score. Suppose the data contain variables `student` (with the student's ID) and `testnum` (for test number). The model specification may then be:

```
model = regset BetaX +
        res(draw=student, ref=eps) +
        res(draw=testnum, ref=u);
```

[7] We could have parameterized distributions in terms of variances and covariances. A future version may include this as an option; the current version parameterizes normal distributions in terms of standard deviations and correlations.

deviation of the residual to that value. It is extremely important to select good starting values, particularly for complicated models (see Chapter 6).

File "`inc1.out`" contains the output. Its structure is very similar to output files from other model types. Note that aML uses maximum likelihood to estimate the model coefficients, even though ordinary least squares would be far more efficient. aML always uses full information maximum likelihood; its strength is in complicated multilevel and multiprocess models, not in simple continuous outcome models such as the one illustrated in this chapter.

2.4. Hazard Model

This section illustrates the steps that are required to estimate a simple hazard model. The main steps to estimate hazard models are similar or identical to those for probit models, as explained in Section 2.1. We therefore only highlight differences between estimation of probit and hazard models. If you have not read Section 2.1 yet, please do so now.

Hazard models, also known as failure time or intensity models, are used when the outcome of interest is a duration until the occurrence of some event: a recovery from an illness, a death, a birth, a marriage, a divorce, a machine failure, a change of jobs, et cetera (Kalbfleisch and Prentice, 1980). The hazard at time *t* is the probability density of the event's occurrence at time *t*, conditional on the fact that the event did not take place before time *t*. The period between the moment at which the event became at risk of occurring and the actual occurrence is known as a spell or episode. We often deal with survey data in which the event of interest has not taken place yet: the patient has not yet recovered, the couple is still married, et cetera. Such spells are known as censored or open spells. The outcome of a hazard process is thus a combination of

- an indicator variable for whether the spell is censored or not; and
- a duration between the moment at which the event became at risk and either the timing of the event (uncensored spell) or the censoring date (censored spell).

There are many types of hazard models; aML handles the most common type, proportional hazard models, where the effect of covariates on the hazard of occurrence is multiplicative. (It actually supports some non-proportional hazard models as well; see Section 13.9) Within that class, aML offers many features which make it extraordinarily flexible. The general formulation of its simplest implementation is:

$$\ln h(t) = \gamma T(t) + \beta' X(t),$$

where $\ln h_i(t)$ is the log-hazard of occurrence at time *t*, $\gamma T(t)$ captures the baseline hazard duration dependence, and $\beta' X(t)$ represent (potentially time-varying) covariates which shift the baseline hazard. The baseline hazard duration dependence, $\gamma T(t)$, is always a piecewise-linear spline (also known as generalized Gompertz or piecewise-linear Gompertz) in aML. Several popular duration patterns may be approximated by the piecewise-linear spline, as we show below.[8]

As an illustration, we study the timing of marital divorce. The unit of observation is a couple; they become at risk of divorcing on the wedding date. We start with a data set in which all variables of interest have been merged and missing values resolved. We first precisely define the

[8] aML always requires specification of a baseline duration dependency pattern. Cox regression models are thus not supported. In practice, this is rarely a limitation, since aML's piecewise-linear duration dependencies adjust readily to any pattern in the data.

outcome; then write out the data in ASCII format; then preprocess them using raw2aml; and finally specify and estimate the hazard model.

2.4.1. Data Preparation

Our sample data come from a panel survey, i.e., a survey in which individuals are interviewed several times, in our case over a period spanning several decades. We first illustrate a model in which all explanatory covariates remain constant for the duration of the spell, i.e., without time-varying covariates, and return to the case with time-varying covariates below. Our data contain the following variables.

`weight`	sampling weight
`wedding`	wedding date
`lowerdiv`	if divorced: lower bound of divorce date; if not divorced, missing.
`upperdiv`	if divorced: upper bound of divorce date; if not divorced, missing.
`survey`	last survey date
`hiseduc`	husband's education (in years of schooling)
`hereduc`	wife's education (in years of schooling)
`heblack`	indicator for whether the husband is African American
`sheblack`	indicator for whether the wife is African American
`agediff`	age difference between husband and wife

Note that educational attainment variables hiseduc and hereduc are in years of schooling, unlike the definitions used in the sample data of Sections 2.1, 2.2, and 2.3.

As stated above, the outcome consists of two parts: an indicator for whether the couple divorced and the duration of the marriage. The event indicator is named the "censor" variable. If the couple did not divorce, the spell is "censored;" if the couple did divorce, the spell is "noncensored." Censored marriages may occur because the couple was still married as of the last survey date, because the couple was still married when the panel ended, when they dropped out of the panel survey, or because one of the spouses died. In our example, we create a new variable, say, "`censor`", which is equal to zero if "`lowerdiv`" and "`upperdiv`" are non-missing, and one otherwise. Nonmissing values of "`lowerdiv`" and "`upperdiv`" are namely indicative of a dissolved marriage, whereas missing values indicate that the marriage was still intact as of the last survey date.[9]

The duration of the marriage is straightforward if the couple is still married as of the last survey date. The censored duration is then equal to the period between the wedding and survey dates. If the couple divorced, however, we are faced with some inevitable uncertainty. Survey data rarely specify the exact date of a divorce. A respondent may report, for example, that he or she divorced in March 1995. This implies that the divorce took place sometime between March 1

[9] To keep the example simple, we ignore spells that are censored due to widowhood.

and March 31. In other words, there is no single moment, but always a "window" within which the event occurred. The width of the window may be one month, one year, or anything else, depending on the precision with which the event date was reported. Even if the exact date is known there is a window, namely of 24 hours. aML always requires the user to specify the lower and upper bounds of event windows. In the example, the lower bound is the period between "`wedding`" and "`lowerdiv`"; the upper bound is the period between "`wedding`" and "`upperdiv`".[10]

The time unit (year, month, day, hour, second) may be chosen by the user to best suit the hazard process under analysis. To prevent numerical problems during the estimation, we recommend that the average duration is in the order of between 1 and 100. When studying divorce data, an appropriate unit of time is a year, since the majority of marriages last between 1 and 60 years, with the average somewhere in between. We could measure the durations in months without major problems, but expect numerical underflows and/or overflows when measuring the durations of marriages in seconds or millenia. For most demographic and medical processes, the year is an appropriate time unit. For other processes, such as space physics or nuclear physics, other units may be better. The choice of time unit does not affect the results. You must be consistent and measure all time variables in the same time unit. So far, we only saw two time variables (the two duration variables), but below we illustrate many more, such as nodes in piecewise-linear duration dependency splines and time marks for time-varying covariates. All those nodes and variables must be measured in the same time unit.

We are now ready to define the outcomes of this hazard process. Suppose all dates are measured in days since some arbitrary date, as they typically are in standard commercial data management packages. Note that there are, on average, about 365.25 days in a year. We use "`censor`" for the censor variable and "`lower`" and "`upper`" for the lower and upper bound of the event window. In SAS, the code would be:

```
if (lowerdiv ~= .) then do;
   censor = 0;
   lower = (lowerdiv-wedding)/365.25;
   upper = (upperdiv-wedding)/365.25;
end; else do;
   censor = 1;
   lower = (survey-wedding)/365.25;
   upper = lower;
end;
```

[10] You may wonder about the precision of the wedding date. It, too, may only be known up to a window. However, hazard models always require that the moment at which the event became at risk of occurrence is specified exactly. In other words, you must create "`wedding`" as an exact date. Very little can be done about this measurement problem. By contrast, aML fully accounts for the range of dates for the event date (up to accuracy of the information provided by the respondent.)

Note that in censored cases, "lower" and "upper" are both set to the duration between the wedding and the last survey date. If a spell is censored, the duration variables must be equal to each other. Uncertainty in the event date is accounted for by the two duration variables. Uncertainty in the begin date of the spell, or in the end date of a censored spell (e.g., the survey date) is not accounted for. Prepare the data using the best guess of the exact date.

Outcomes of a hazard process

The outcome of a hazard process consist of three variables. If the event happened, the censor variable is zero and the two duration variables are equal to the durations from the beginning of the spell to the lower and upper bound of the event window. If the event did not happen, the censor variable is one and the two duration variables are both equal to the duration from the beginning to the end of the spell.

2.4.2. Conversion into aML Format Using raw2aml

Conversion of the data into a format that aML understands proceeds in the same way as illustrated in sections above. We first write out the data in ASCII format and then convert them using raw2aml. Note that the "wedding", "lowerdiv", "upperdiv", and "survey" dates are no longer relevant. ASCII file "Samples\Chapter2\divorce1.raw" contains the following variables: id weight censor lower upper hiseduc hereduc heblack sheblack agediff. The first five records are:

```
9 23 1 10.546 10.546 12 12 0 0 1.013
11 23 1 34.943 34.943 3 3 0 0 0.687
13 23 0 2.793 2.875 8 8 0 0 2.352
15 23 0 15.012 20.052 7 7 0 0 2.352
33 17 1 1.418 1.418 12 10 0 0 0.830
```

The raw2aml control file, "divorce1.r2a", contains the following:

```
1  ascii data file = divorce1.raw;
2
3  var = weight censor lower upper hiseduc hereduc heblack sheblack agediff;
```

As always, it is important to check the output summary file, "divorce1.sum:":

```
 1  Documentation for 'divorce1.dat'
 2  Created on Sun Feb  6 11:25:55 2000 with raw2aml version 1.00.
 3  Ascii data set: 'divorce1.raw'
 4
 5  Number of observations:    3371
 6
 7  ------------------------------------------------------------
 8
 9  LEVEL 1 VARIABLES:
10  Variable     N       Mean    Std Dev        Min        Max
```

```
 11  _id        3371   7761.647   4959.523         9.0    17302.0
 12  weight     3371   15.88876   10.23573         1.0       32.0
 13  censor     3371   .6938594   .4609572         0.0        1.0
 14  lower      3371   17.99482   15.02624        0.06     73.068
 15  upper      3371   18.83722   15.04849       0.079     73.068
 16  hiseduc    3371   11.54109   3.012785         1.0       21.0
 17  hereduc    3371   11.51824   2.858982         1.0       21.0
 18  heblack    3371   .2210027   .4149838         0.0        1.0
 19  sheblack   3371   .2438446   .4294637         0.0        1.0
 20  agediff    3371   2.293223   4.923548    -39.663     38.081
 21
 22  ------------------------------------------------------------
 23
 24  NOTE: there is variation in all data variables.
```

Convince yourself that the number of observations is correct. If it is too low, chances are that you listed too many variables in the raw2aml control file.

The mean sample weight, `weight`, is 15.88876. If we were to weight each observation by this weight, the false impression would arise that the sample is over 15 times as large as it actually is, and all t-statistics would be inflated. We will deal with this issue below.

2.4.3. Model Specification and Estimation

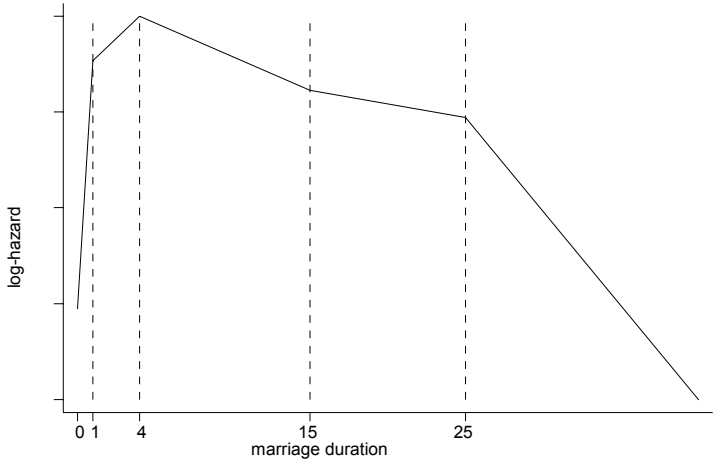
A simple divorce hazard equation is specified in "`div1.aml`":

```
 1  option normweight = weight;
 2
 3  dsn = divorce1.dat;
 4
 5  define spline DurMar; nodes = 1 4 15 25;
 6
 7  define regressor set Getdiv;
 8     var = 1 heblack (heblack!=sheblack)
 9           (hiseduc<12) (hiseduc>=16)
10           (agediff>10) (agediff<-10);
11
12  hazard model;
13     censor=censor; duration=lower upper;
14     model = durspline(origin=0, ref=DurMar) +
15             regset Getdiv;
16
17  starting values;
18
19  dur0-1       TT   -.041
20  dur1-4       TT   -.041
21  dur4-15      TT   -.041
22  dur15-25     TT   -.041
23  dur25+       TT   -.041
24  Constant     TT   -5.64
25  heblack      FT    0
26  mixrace      FT    0
```

```
 27  dropout     FT    0
 28  college     FT    0
 29  heolder     FT    0
 30  sheolder    FT    0
 31  ;
```

Line 1 shows the use of "`option normweight`", short for normalized weight. Our sample data contain an oversample of African Americans, so we wish to maximize the log-likelihood such that individual observations are weighted by variable "`weight`". This could be achieved by "`option weight = weight`". However, as we saw above in the data summary file, "`divorce1.sum`", the average weight more than one, so that weighted optimization would result in inflated t-statistics and a false sense of parameter estimate accuracy. We therefore normalize the weights, i.e., we rescale the weights such that the average weight is one, and the weights sum up to the number of observations. The normweight option makes aML rescale the weights.

Line 5 introduces a new building block, the "spline." A spline is a piecewise-linear transformation of a variable. It has two major applications in aML. First, a spline transformation offers a convenient way to relax the assumption that the effect of a covariate is linear. Instead, the effect may be linear over a certain range, and again linear but with a different coefficient (slope) over the next range, et cetera (also see page 28). Second, a spline may be used to parameterize the shape of the baseline log-hazard function. To illustrate this, consider the figure below.



Piecewise-Linear Baseline Log-Hazard of Divorce

The figure illustrates the piecewise-linear spline, in this case with nodes at 1, 4, 15, and 25 years. (Nodes are sometimes called knots or bend points.) The pattern is taken directly from estimates of the baseline log-hazard of divorce, as shown below. It illustrates that the hazard of divorce increases very rapidly during the first year of marriage. It then continues to increase, though not as fast, for the next three years. Between the fourth and fifteenth wedding anniversary,

the risk decreases; it declines more mildly for the next ten years, and more steeply after the silver anniversary.

aML requires the baseline log-hazard pattern to be piecewise-linear. In practice, this is hardly a constraint. The user may specify any number of nodes at any desired location. As a result, the baseline pattern is capable of approximating any pattern in the data.

It is not always easy to decide on the "right" number of nodes and their locations. Inevitably, it will require some experimentation. We recommend the following strategy.

Tip: How to find adequate baseline log-hazard nodes

First specify and estimate a stripped-down hazard model with an intercept and a linear (Gompertz) log-hazard, i.e., a spline without nodes (`nodes=;`). This results in estimates of an intercept and a slope. Then specify four or five nodes, spread out roughly evenly over the relevant spells' range. Initialize the intercept at the first-stage estimated intercept, and all slopes at the first-stage slope estimate. Inspect the resulting pattern and remove nodes of which the surrounding slopes are roughly equal. Typically, two or three nodes are sufficient to adequately capture the baseline duration pattern. Experiment by shifting individual nodes to find a pattern that captures the essence of the pattern in the data.

For the interested reader, we formally present the baseline duration pattern and the likelihood function. The baseline log-hazard pattern is based on the following transformation of the spell duration, *t*:

$$T(t) = \begin{pmatrix} \min[t, \nu_1] \\ \max[0, \min[t-\nu_1, \nu_2-\nu_1]] \\ \max[0, \min[t-\nu_2, \nu_3-\nu_2]] \\ \max[0, \min[t-\nu_3, \nu_4-\nu_3]] \\ \max[0, t-\nu_4] \end{pmatrix},$$

where $\nu_1 = 1$, $\nu_2 = 4$, $\nu_3 = 15$, and $\nu_4 = 25$ denote the nodes. Covariates enter the log-hazard equation additively, so they proportionally shift the baseline hazard duration pattern. For now, consider only covariates that are constant over the duration of the spell. The effects of such covariates are captured by $\beta' X$. Recall that log-hazard is defined as:

$$\ln h(t) = \gamma T(t) + \beta' X .$$

The survivor function (i.e., the probability that the event has not happened yet at time *t*) is by definition:

$$S(t) = \exp\left\{-\int_0^t h(\tau)d\tau\right\} = \exp\left\{-\int_0^t e^{\gamma T(\tau)}d\tau\right\}^{\exp\{\beta' X\}},$$

and the likelihood is:

$$L = \begin{cases} S(t) & \text{if the spell is censored at } t; \\ S(t^l) - S(t^u) & \text{if the event occurred between } t^l \text{ and } t^u. \end{cases}$$

Line 5 defined the spline:

```
define spline DurMar; nodes = 1 4 15 25;
```

We assigned the name "DurMar" to this spline. As with all building blocks, names of splines may be up to twelve characters long. Our spline has four nodes, at 1, 4, 15, and 25 time units. These time units correspond to the time unit that we chose in the data preparation, i.e., the nodes are measured in years since the beginning of the spell. Given four nodes, there are five slopes: from the beginning of the spell until 1 year, between 1 and 4 years, between 4 and 15 years, between 15 and 25 years, and beyond 25 years. A spline with n nodes generates n+1 slope parameters.

```
define regressor set Getdiv;
   var = 1 heblack (heblack!=sheblack)
         (hiseduc<12) (hiseduc>=16)
         (agediff>10) (agediff<-10);
```

Lines 7-10 defined a regressor set. It is similar to those we encountered in probit, logit, and continuous models. The first expression, "1", serves as intercept. The third expression, `(heblack!=sheblack)` is an indicator variable for mixed-race couples. The third and fourth are indicator variables for whether the husband is a high school drop-out or college graduate. The `agediff` transformations flag couples in which the husband is substantially older than the wife and vice versa. (In the example, all covariates happen to be either zero or one. This is not necessary.)

```
hazard model;
   censor=censor; duration=lower upper;
   model = durspline(origin=0, ref=DurMar) +
           regset Getdiv;
```

Lines 12-15 specify the model. The censor statement expects a variable or expression which is one if the spell is censored (the event did not happen) and zero otherwise. In our example, variable "`censor`" was created for this purpose. The duration statement expects two variables or expressions denoting the lower and upper bounds of the window within which the event happened. For censored spells, there is no event window and the duration variables must both be equal to the length of the censored spell.

The model statement specifies the right-hand-side of the log-hazard equation. The first component is "`durspline(origin=0, ref=DurMar)`", to be read as "duration spline `DurMar` which originates at the beginning of the spell." The origin specification indicates at what moment, relative to the beginning of the spell, the duration pattern starts. It thereby allows for duration

patterns which start at moments other than the beginning of the spell. This is particularly useful in cases where the hazard depends on multiple durations, multiple "clocks;" see Section 5.9 on overlapping splines. The "`ref`" specification indicates which of the splines that have been defined in the control file should be used. The model also includes regressor set "`Getdiv`".

```
starting values;

dur0-1      TT   -.041
dur1-4      TT   -.041
dur4-15     TT   -.041
dur15-25    TT   -.041
dur25+      TT   -.041
Constant    TT   -5.64
heblack     FT    0
mixrace     FT    0
dropout     FT    0
college     FT    0
heolder     FT    0
sheolder    FT    0
;
```

Lines 17-31 specify the parameters' starting values. Since the spline was defined first, its corresponding five slope parameters are initialized first. We give them easy-to-interpret names. The specification of good starting values is extremely important in hazard models. We built this model up in steps. As suggested in the Tip box on page 46, we first specified and estimated a simple Gompertz hazard model, without covariates. A good starting value for the intercept is given by (see Section 6.6):

$$\gamma_0 = -\ln\left(\frac{1}{N_{nc}}\sum_{i=1}^{N} t_i\right),$$

where N_{nc} is the number of non-censored spells in the data, N is the total number of spells, and t_i is the length of spell i. For censored spells, t_i is well-defined. For non-censored spells, however, we argued above that there always is a window within which the event happened, not an exact moment. Unfortunately, the intercept formula which incorporates such windows is very much more complicated than the one presented here. We therefore propose that you cheat, and set t_i equal to the midpoint of the event window for noncensored spells. In our data, 2,339 marriages survived to an average duration of 21.79 years and 1,032 marriage ended in a divorce after an average of 10.78 years. We therefore initialized the intercept at $-\ln\left(\frac{1}{1032}(2339*21.79+1032*10.78)\right) = -4.097$, and the Gompertz slope at zero. We estimated that model (not shown) and found an intercept of -5.64 and a slope equal to -.041. Control file "`div1.aml`" starts at this stage. We add nodes at 1, 4, 15, and 25 years and initialize the intercept

and all slopes to the Gompertz estimates. In the first round, we estimate the baseline pattern, and in the second round we add regressors.

aML produced output file `div1.out`, which we partially replicate:

```
     <license and control information>

 17  Note: the number of observations is 3371; they generated 3371 outcomes.
 18
 19  Observations are weighted by normalized variable 'weight'; the sum of
 20  weights is 53561.0, the number of observations that are used in the
 21  estimation 3371, so weights are scaled by .062938.
 22
 23  The following regressor sets have been defined:
 24
 25  define regressor set Getdiv;
 26     var = 1 heblack (heblack!=sheblack) (hiseduc<12) (hiseduc>=16)
 27           (agediff>10) (agediff<-10);
 28
 29     ---------+-----------------  Summary statistics  ----------------
 30     name     |      #        Mean     Std Dev         Min         Max
 31     ---------+-------------------------------------------------------
 32     Constant |   3371         1.0         0.0         1.0         1.0
 33     heblack  |   3371    .2210027    .4149838         0.0         1.0
 34     mixrace  |   3371    .1901513    .3924786         0.0         1.0
 35     dropout  |   3371    .3820825    .4859686         0.0         1.0
 36     college  |   3371    .1269653    .3329835         0.0         1.0
 37     heolder  |   3371    .0341145    .1815502         0.0         1.0
 38     sheolder |   3371    .0154257    .1232568         0.0         1.0
 39
 40  The following splines have been defined:
 41
 42  define spline DurMar;
 43     nodes = 1 4 15 25;  /* slope coefficients: dur0-1 - dur25+ */
 44
 45     ---------+-----------------  Summary statistics  ----------------
 46              |      #        Mean     Std Dev         Min         Max
 47     ---------+-------------------------------------------------------
 48     origin   |   3371         0.0         0.0         0.0         0.0
 49
 50
 51  The following models have been specified:
 52
 53  hazard model;
 54     censor = censor;
 55     duration = lower upper;
 56     timemarks = ;
 57     model = durspline(origin=0, ref=DurMar) +
 58        regset Getdiv
 59        ;
 60
 61     Summary statistics of the outcome and selected variables:
 62
 63     Hazard spell durations (for noncensored spells, lower and upper
```

```
 64      duration variables and their difference; for censored spells,
 65      spell duration):
 66
 67          censor |      #        Mean     Std Dev         Min         Max
 68      -----------+--------------------------------------------------------
 69        / lower  |   1032    9.402495    7.756847        0.06      50.075
 70      0 - upper  |   1032    12.11546    10.11249       0.142      70.439
 71        \ window |   1032    2.712905    6.700789    .0110002      46.954
 72      1 - spell  |   2339    21.78585    15.87305       0.079      73.068

     et cetera...
```

Lines 19-21 document the normalized weight. The weights are scaled down such that the sum of weights is equal to the number of observations, i.e., the average weight is one.

Lines 23-38 restate the regressor set definition and provide (unweighted) summary statistics.

Lines 40-43 repeat the spline definition and states that you assigned the names `dur0-1` through `dur25+` to the slope parameters. This information serves as a check that you initialized all parameters in the correct order. Lines 45-48 provide summary statistics of the origin variables in spells to which the spline applies. In our example, the duration pattern always originates at the beginning of the spell, so that the origin is always zero.

Lines 53-59 repeat the hazard model statement. Note the "`timemarks = ;`" specification, which was not in our control file. The timemarks statement is optional. It is needed only when the model contains time-varying covariates; see below.

Lines 63-72 provide summary statistics on the duration of the hazard spells that contribute to the model. For noncensored spells, aML summarizes the lower bound, the upper bound, as well as the width of the event window. For censored spells, it summarizes the length of the spells. It also states how many spells are censored and how many noncensored. Please check that these statistics match those in your data, especially when working with complicated multilevel, multiprocess data.

The remainder of the output file contains iteration-by-iteration results of the optimization process. Their interpretation is discussed above in Section 2.1. Typing "`mktab div1`" from the command line presents the results in a user-friendly format:

```
dur0-1       1.7334 ***
            (0.5168)
dur1-4       0.1011 *
            (0.0524)
dur4-15     -0.0458 ***
            (0.0120)
dur15-25    -0.0197
            (0.0171)
dur25+      -0.1315 ***
            (0.0207)
Constant    -5.6546 ***
            (0.4684)
```

```
heblack      0.0840
            (0.1432)
mixrace      0.4323 ***
            (0.1525)
dropout     -0.2458 ***
            (0.0726)
college     -0.2946 ***
            (0.0993)
heolder     -0.4765 **
            (0.2191)
sheolder     0.3891
            (0.2764)

ln-L       -5916.01
```

The significance asterisks, '*', '**', and '***', indicate significance at 10, 5, and 1 percent, respectively. Note that the log-hazard increases sharply during the first year after the wedding,[11] continues to increase—though only significant at the 10 percent level and at a more moderate pace—until the fourth anniversary; decreases moderately until the fifteenth anniversary; decreases very slightly for the next ten years, and continues to decrease thereafter. This pattern is simulated in the figure on page 45. The estimated covariates indicate that race has little effect by itself, but mixed-race couples are more likely to divorce; high school drop-outs and college graduates tend to be in more stable marriages than high school graduates (the omitted category); and marriages in which the husband is substantially older than the wife are more stable.

2.4.4. Time-varying covariates

Now consider a hazard process in which one or more covariates change value during the spell, but are constant over intervals within the spell. These value changes must be discrete, i.e., the covariates must jump from one value to another at a point in time. Variables that change continuously over the duration of the spell (such as age and calendar time) should be captured using duration dependencies, i.e., splines. Baseline hazard patterns in aML may consist of as many duration dependencies as desired to capture variables that change continuously. Covariates that are constant within intervals of time but change between intervals are "time-varying" covariates.

Time-varying covariates are implemented in aML as an additional data level. For example, a marriage may be a level 1 unit, and an interval a level 2 unit. There may be any number of level 2

[11] The rate of increase is 1.7319 per year, i.e., the hazard after one year is $\exp(1.73) = 5.65$ times the hazard at the time of the wedding. For moderate slopes, such as -0.0464 per year between the fourth and the fifteenth anniversary, the coefficients may be interpreted as approximate percentage change, i.e., a decline of approximately 4.64 percent per year: $100 * (\exp(-0.0464) - 1) \approx -4.64$ percent.

units within a level 1 (and any number of level 3 units within a level 2 unit, et cetera). Level 1 variables may include the spell outcomes (one censor and two duration variables), as well as variables that are not time-varying, such as husband's and wife's education, race, age difference, et cetera. Level 2 variables may include variables that change during the spell, such as the number of children that the couple has.

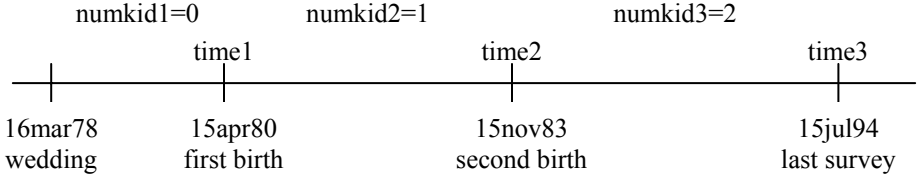
Concept: Multilevel Data

When data contain multiple units of interest that are nested, they naturally fall into a multilevel structure or hierarchy. For example, a family consists of several persons; every person has held zero or more jobs; and the data contain zero or more (annual) records of the wage rate on each job. aML numbers these levels "top-down," i.e., the largest unit (family) is level 1, the second (person) is level 2, the third level (job) is level 3, the fourth level (wage rate) is level 4, et cetera. Note that some other multilevel software packages number their levels in the reverse order!

To determine the effect of time-varying covariates on the timing of events, it is not only necessary to know what values the covariate takes on, we also need to know exactly when it changes to the next value. In other words, not only the time-varying covariates, but also the "time marks" which define intervals need to be known.

Consider again our divorce data, now including a time-varying variable, the number of children that is born during the couple's marriage. The time marks are equal to the children's birth dates, measured relative to the wedding date. We show two ways of writing out the data in ASCII format.

Suppose that the most fertile couple in our data gave birth 16 times, and that all children's birth dates have been converted into years since the wedding and stored as variables `time1-time16`. In addition, variables `numkid1-numkid16` contain the number of children that the couple has. Typically, `numkid1`=0, `numkid2`=1, et cetera, but twins would increase the `numkid` series by two. Most couples gave birth fewer than 16 times, so many of the `time1-time16` and `numkid1-numkid16` variables have missing values. Consider an example:



This couple married on March 16, 1978. Their two children were born on April 15, 1980 and November 15, 1983. They were last interviewed on July 15, 1994, at which time their marriage was still intact. During the first interval, their number of children was zero (`numkid1`=0); at `time1`=2.08 years after the wedding, the wife gave birth to their first child, so that `numkid2`=1.

At time2=5.67 years after the wedding, a second child was born, so numkid3=2. No more children were born until the end of the spell, time3=16.33 years after the wedding. Variables time4-time16 and numkid4-numkid16 contain missing values. (Missing and represented by a dot, ".", in SAS, Stata, SPSS, et cetera. aML does not accept missing values; we show below how to solve this.) We create variable numint=3 to indicate how many intervals there are for this couple.

We assign level 1 to each marriage and level 2 to each interval. With multilevel data, the data are always written out at level 2, i.e., there is one record per level 2 unit (interval). (Chapters 3 and 10 explain the full implications of this rule.) In a SAS data step, the data may be written out as follows:[12]

```
array time(*)   time1-time16;
array numkid(*) numkid1-numkid16;
do i=1 to numint;
   put id
       weight censor lower upper hiseduc hereduc
       heblack sheblack agediff
       time(i) numkid(i);
end;
```

The first lines of the resulting ASCII file (divorce2.raw in the sample data) are:

```
  9 23 1 10.546 10.546 12 12 0 0 1.013  3.734 0
  9 23 1 10.546 10.546 12 12 0 0 1.013 10.546 1
 11 23 1 34.943 34.943  3  3 0 0 0.687  0.767 0
 11 23 1 34.943 34.943  3  3 0 0 0.687 32.512 1
 11 23 1 34.943 34.943  3  3 0 0 0.687 34.943 2
 13 23 0  2.793  2.875  8  8 0 0 2.352  2.585 0
 13 23 0  2.793  2.875  8  8 0 0 2.352  2.875 1
 15 23 0 15.012 20.052  7  7 0 0 2.352 60.052 0
 33 17 1  1.418  1.418 12 10 0 0 0.830  1.418 0
  |     |    |     |                      |    |
id  censor lower  upper                time   numkids
```

These lines correspond to five observations (five marriages). The first variable on each line, the ID variable, indicates which records (intervals) belong together in the same spell. The last two variables are time and numkids. The first couple (id=9) thus had zero children from their wedding to 3.734 years after the wedding, and one child thereafter until 10.546 years after the

[12] In Stata, you would need to create one Stata-observation for every level 2 unit (interval):

```
reshape long time numkid, i(id) j(interval)
drop if time==.
outfile id weight censor ... agediff time numkid using filename
```

wedding, when they were last surveyed. (We know they did not divorce because their third variable, `censor`, is equal to 1.) The second couple (`id=11`) had one child 0.767 years after their wedding, and another after 32.512 years. The third couple (id=13) had a child 2.585 years after the wedding. Note that `censor=0`, `lower=2.793`, and `upper=2.875`, so they divorced sometime between 2.793 and 2.875 years after the wedding. The fourth couple (id=15) remained childless until they divorced. Only limited information was available on the date of divorce: it took place sometime between 15.012 and 20.052 years after the wedding. The fifth couple (`id=33`) remained childless until their last survey date, 1.418 years after the wedding.

Note that the first time mark is not zero. All spells start at time $t_o = 0$, so there is no need to explicitly write out that time mark. Each time variable thus marks the end of its interval.

Note that the highest time mark is always at least as large as the end of the spell (variable `upper`). If this were not the case, aML would not know what the value of the time-varying covariate is between the highest time mark and the end of the spell.

The ASCII data may be converted into aML format by the following raw2aml control file (`divorce2.r2a`):

```
1  input data file = divorce2.raw;
2
3  level 1 var = weight censor lower upper hiseduc hereduc heblack
4                sheblack agediff;
5  level 2 var = time numkid;
```

In raw2aml control files that we have seen so far, variable lists were specified by a "`var = ...`" statement. This is fine as long as there is only one level in the data. Here, however, we have two levels, and we need to indicate what level each variable is.

Note that `time1-time16` and `numkid1-numkid16` are (up to) 16 variables in the original data. In the aML-formatted data, however, they are just one variable each (`time` and `numkid`). Instead of one SAS observation with 16 variables, we created one aML observation with up to 16 level 2 units or "branches."

Before illustrating how to specify a hazard model with time-varying covariates in an aML control file, we present the likelihood function. Consider a log-hazard process with time-varying covariates *X(t)*:

$$\ln h(t) = \gamma T(t) + \beta' X(t),$$

where we suppress the subscripts for both the individual and the spell. Denote the baseline log-hazard as the duration dependency part, $\ln h_0(t) = \gamma T(t)$, and the corresponding baseline survivor function by $S_0(t)$. Denote the points in time that mark intervals by $(t_0, t_1, \ldots, t_I)$, so that $X(t)$ is constant and equal to $X(t_1)$ between t_0 and t_1, jumps to a $X(t_2)$ at t_1, remains constant between t_1 and t_2, jumps to $X(t_3)$, et cetera. All $(t_0, t_1, \ldots, t_I)$ are measured relative to the beginning of

the spell, i.e., $t_0 = 0$ and t_I is equal to the total duration of the spell (the upper bound for noncensored spells). The survivor function at various points in time is:

$$
\begin{aligned}
S(t_0) &= 1 \\
S(t_1) &= \left\{S_0(t_1)\right\}^{\exp\{\beta' X(t_1)\}} \\
S(t_2) &= S(t_1)\left\{\frac{S_0(t_2)}{S_0(t_1)}\right\}^{\exp\{\beta' X(t_2)\}} \\
&\vdots \\
S(t_I) &= \prod_{i=1}^{I}\left\{\frac{S_0(t_i)}{S_0(t_{i-1})}\right\}^{\exp\{\beta' X(t_i)\}}
\end{aligned}
$$

The likelihood is as before:

$$
L = \begin{cases} S(t) & \text{if the spell is censored at } t; \\ S(t^l) - S(t^u) & \text{if the event occurred between } t^l \text{ and } t^u. \end{cases}
$$

We now specify the model, including the time-varying covariate (`div2.aml`):

```
 1  option normweight = weight;
 2
 3  dsn = divorce2.dat;
 4
 5  define spline DurMar; nodes = 1 4 15 25;
 6
 7  define regressor set Getdiv;
 8     var = 1 heblack (hiseduc<12) (hiseduc>=16)
 9      (agediff>10) (agediff<-10) (heblack!=sheblack)
10      numkids;
11
12  hazard model;
13     censor=censor; duration=lower upper; timemarks=time;
14     model = durspline(origin=0, ref=DurMar) +
15        regset Getdiv;
16
17  starting values;
18
19  dur0-1      T    1.7333887636
20  dur1-4      T     .1011364705
21  dur4-15     T    -.0457769887
22  dur15-25    T    -.0197150656
23  dur25+      T    -.1314955828
24  Constant    T   -5.6545800935
25  heblack     T     .0840387419
26  mixrace     T     .4323170948
27  dropout     T    -.2457713662
28  college     T     -.294608909
29  heolder     T    -.4764981159
30  sheolder    T     .3891406844
```

```
31  numkids     T      0
32  ;
```

Note line 10, in which we add level 2 variable `numkids` to the list of regressors. Regressor sets may contain any level variable, and expressions involving variables at different levels are also allowed. (To be precise, all models require that variables in regressor sets must be at the same or higher, more aggregated level as the outcome variable. Hazard models form an exception in that time-varying variables are allowed and must be one level lower than the outcome. Variables at even lower levels would result in an error message, as it would not make sense to include such variables.)

The hazard model specification contains a new statement: "`timemarks=time`". If any regressor set that is used in a hazard model contains a time-varying variable, the timemarks statement is mandatory and indicates the points in time that separate intervals. The timemarks statement specifies the name of a variable that is one level below the level of the censor and duration variables. It may not be an expression. Here, variables `censor`, `lower`, and `upper` are level 1 variables, whereas `time` is a level 2 variable. The time marks variable measures time relative to the beginning of the spell. Its values must therefore be strictly positive and increasing from subbranch to subbranch. Each time variable marks the end of its interval, not the beginning. The value of the last time mark must be at least as large as the end of the spell, because otherwise aML would not know what the value of the time-varying covariate is between the highest time mark and the end of the spell.

The starting values of "`div2.aml`" are the converged values of "`div1.aml`". (Recall that the starting values in "`div1.aml`" are conveniently updated with the converged estimates in "`div1.out`" by typing "`update div1`"; see Section 2.1.7.) The newly added variable, `numkids`, is initialized to zero in the starting values statement. Running aML results in "`div2.out`". We compare the results of "`div1.out`" and "`div2.out`" by typing "`mktab div1 div2`" (see Section 2.1.8):

```
                 div1          div2

dur0-1       1.7334 ***    1.7519 ***
            (0.5168)      (0.5167)
dur1-4       0.1011 *      0.1280 **
            (0.0524)      (0.0527)
dur4-15     -0.0458 ***   -0.0351 ***
            (0.0120)      (0.0123)
dur15-25    -0.0197       -0.0195
            (0.0171)      (0.0171)
dur25+      -0.1315 ***   -0.1305 ***
            (0.0207)      (0.0208)
Constant    -5.6546 ***   -5.6475 ***
            (0.4684)      (0.4684)
heblack      0.0840        0.0450
```

```
                (0.1432)        (0.1444)
mixrace          0.4323 ***     -0.2778 ***
                (0.1525)        (0.0725)
dropout         -0.2458 ***     -0.2981 ***
                (0.0726)        (0.0994)
college         -0.2946 ***     -0.5005 **
                (0.0993)        (0.2187)
heolder         -0.4765 **       0.3822
                (0.2191)        (0.2732)
sheolder         0.3891          0.4972 ***
                (0.2764)        (0.1534)
numkids                         -0.1109 ***
                                (0.0266)

ln-L           -5916.01        -5905.82
```

Adding variable `numkids` changes the estimates of other covariates and the baseline pattern only little. The number of children born while the couple is married has a negative effect on the hazard of divorce, i.e., children appear to stabilize marriages.[13]

[13] We ignored the potential endogeneity of children in this discussion. If it is the case that couples with marital problems tend to postpone childbearing, the number of children is endogenous, and the above estimate overstates the actual effect of children on marital stability. One way to deal with the endogeneity issue is to model fertility and marital disruption jointly. This was the subject of the first journal article demonstrating simultaneous hazard models (Lillard, 1993).

2.5. Binomial Model

This section illustrates the steps that are required to estimate a simple binomial model. The main steps to estimate binomial models are similar or identical to those for probit models, as explained in Section 2.1. We therefore only highlight differences between estimation of probit and binomial models. If you have not read Section 2.1 yet, please do so now.

If outcome *Y* follows a binomial distribution, the probability distribution is:

$$\Pr(Y) = \binom{n}{x} p^x (1-p)^{n-x}$$

where *n* is the number of independent trials (also known as exposure), *p* is the probability in each trial that the outcome will be a success and *x* is the number of successes. Both *n* and *x* must be variables in the data; probability *p* may be estimated as a parameter, or it may be a function of regressor sets and residuals (heterogeneity components).

2.5.1. The Probability is a Constant

We begin with the simplest possible binomial model in which *p* is a constant to be estimated. Among the sample files is "`count.raw`" with raw data on 1,000 observations. It has just one level and contains variables `n`, `x1`, `x2`, `y1`, and `y2`. As controlled by "`count.r2a`", raw2aml converted the data into "`count.dat`". Exposure variable `n` ranges from zero to 14, as shown in data summary file "`count.sum`"; our outcome of interest is `y1` with a range from zero to 10. aML control file "`count1.aml`" contains the following.

```
 1  option title = "Binomial with constant probability";
 2  dsn = count.dat;
 3
 4  define parameter Prob; range = (0,1);
 5
 6  binomial model;
 7     outcome = y1;
 8     exposure = n;
 9     probability = linear(par Prob);
10
11  starting values;
12
13  p   T  .5
14  ;
```

Line 1 contains an optional title. It contains spaces and must therefore be delimited by single or double quotes.

```
define parameter Prob; range = (0,1);
```

Line 4 introduces a new type of building block, the parameter. A parameter is a scalar with a variety of purposes. It may be used to estimate a constant, as in this example, or as an intercept, or in interaction terms, et cetera. The name of a parameter may not exceed twelve characters. The parameter in this example contains the optional range specification. By default, parameters may take on any number on the real line. Optionally, their range may be restricted to strictly positive (0,Inf), strictly between 0 and 1 (0,1), or strictly less than one in absolute value (-1,1). Since we will use the parameter to represent a probability, we restrict the range to (0,1). Without this range restriction, aML's optimization search may venture into illegal values.

```
binomial model;
   outcome = y1;
   exposure = n;
   probability = linear(par Prob);
```

Lines 6–9 specify the binomial model. The count outcome must be specified first. Next is the exposure variable, which must be at the same level as the outcome. The probability statement determines the structure of the equation for the probability. In this case, the probability is simply a parameter, i.e., there is no transformation of any type. (See below for alternatives.) Instead of specifying that the probability is simply equal to parameter `Prob`, we need to state that it is linearly related (one-to-one) to parameter `Prob`. Indeed, "`par Prob`" (short for "`parameter Prob`") is a reference to previously defined parameter `Prob`; the probability may also be a linear combination of parameters, regressor sets, and/or residuals (heterogeneity components).

In this illustration, the entire model only contains one parameter, the probability. We initialize it halfway on the permissible range. File "`count1.out`" contains the output, repeated here in part.

```
 1  ================================================================================
 2  =                    Binomial with constant probability                        =
 3  ================================================================================

    et cetera...

25  The following parameters, vectors, and matrices have been defined:
26
27  define parameter Prob;  /* coefficient name: p */
28     range=(0,1);
29
30
31  The following models have been specified:
32
33  binomial model;
34     outcome = y1;
35     exposure = n;
36     prob = linear(par Prob);
37
38     Summary statistics of the outcome and selected variables:
```

```
     et cetera...

230  ================================================================================
231  =                  ESTIMATION CONVERGED SUCCESSFULLY                           =
232  =                        RESULTS OF ESTIMATION                                 =
233  ================================================================================
234
235  Convergence based on:
236     Weighted gradient norm:          .0718463 < .1
237     Relative function improvement:   7.22E-06
238     Gradient norm:                   13.64425
239     Relative parameter change:       .0041449
240
241  ================================================================================
242
243  Log Likelihood:  -1424.3868
244                                                BHHH-based, non-corrected
245    Parameter    Free?     Estimate          Std Err        T-statistic
246
247    1  p           T     .29362694189     .00526568481       55.7623
248
249  ================================================================================
250
251  Elapsed clock time is 3 seconds.
```

Note that the title is repeated in lines 1-3. Lines 25-28 restate the definition of the parameter. Its name is `Prob`; the associated parameter coefficient that is estimated was labeled "`p`" in the starting values. Lines 31-36 repeat the model specification, followed by tabulations of the outcome and the exposure variables (not shown). The model converged with a probability estimate of 0.294. That value could also have been found from the raw data: the ratio of the means of variables `y1` and `n`, as found in data summary file "`count.sum`", is 2.013/6.846=0.294. We could have saved ourselves the trouble of estimating this very simple model.

2.5.2. The Probability is a Function of Covariates

In a slightly more complicated case, the probability is assumed to be a function of covariates in the data. Since a probability must be between zero and one, we use a logistic or cumulative normal transformation to link covariates to the binomial probability:

$$p = \frac{1}{1+\exp(-\beta' X)}$$

$$\text{or} \quad p = \Phi(\beta' X).$$

Control file "`count2.aml`" contains an example in which the probability is assumed to be related, after a logistic transformation, to covariates `x1` and `x2`:

```
 1  dsn = count.dat;
 2
 3  define regressor set BetaX; var = 1 x1 x2;
 4
 5  binomial model;
 6     outcome = y2;
 7     exposure = n;
 8     probability = logistic(regset BetaX);
 9
10  starting values;
11
12  Constant    T    .5178432
13  x1          T     0
14  x2          T     0
15  ;
```

Line 8 specifies that the probability is a logistic transformation of regressor set BetaX. We could have specified "`probability = probit(regset BetaX)`" and used the cumulative normal transformation; the results would have been very similar.

We initialized the intercept at 0.5178432 (line 12). Summary file "`count.sum`" shows an average probability of outcome y2 of 4.29/6.846=0.627; the log-odds of that average is log(0.627/(1-0.627))=0.5178432, which is the optimal value for the intercept if other regressors are zero. In a simple model such as this one, the penalty for specifying poor starting values tends to be small. (If we had initialized all parameters to zero, it would have taken one additional iteration to converge, i.e., a fraction of a second.) However, in more complicated models, careful selection of good starting values is often critical.

2.6. Poisson Model

This section illustrates the steps that are required to estimate a simple Poisson model. The main steps to estimate Poisson models are similar or identical to those for probit models, as explained in Section 2.1. We therefore only highlight differences between estimation of probit and Poisson models. If you have not read Section 2.1 yet, please do so now.

Poisson regression models may be appropriate for the analysis of count data, that is, outcomes that may take on values 0, 1, 2, … Unlike binomial count models, there is no upper bound on the potential outcome. The model was first derived by Siméon-Denis Poisson in 1837 (Poisson, 1837).

The basic specification of the Poisson regression model is as follows. Consider an incidence rate λ_i for observation *i*. The probability distribution of outcome variable Y_i is given by:

$$\Pr(Y_i = y_i) = \frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!}.$$

In its most basic form, aML follows the usual parameterization of the incidence rate:[14]

$$\lambda_i = \exp(\beta' X_i).$$

The incidence rate is equal to the expected number of occurrences per observation. If observations vary in their exposure (e.g., number of periods over which events accumulate), the per-observation incidence rate λ_i is equal to the product of the exposure E_i and the per-period incidence rate:

$$\lambda_i = E_i \exp(\beta' X_i).$$

Model Specification and Estimation

Consider a model of shipping accidents (McCullagh and Nelder, 1983). We are interested in the accident rate of ships of various types and vintages over two operation periods. Sample data set `Chapter2\ships.dat` contains variables `accident` (number of shipping accidents over an observation period), `months` (number of service months), `lnmonths` (logarithm of `months`), `shiptype` (type of ship, coded 1-5), indicator variables for the year in which the ships were constructed (1960-64, 1965-69, 1970-74, 1975-79), and indicator variables for the operation period (1960-74, 1975-79). The number of observations in the data is 34.

[14] The multilevel extension allows for one or more (integrated) residuals, and of course all types of interactions are supported. See Section 13.11.

Naturally, the more ships are sailing, the higher the expected number of accidents. The exposure E_i is thus given by variable `months`. The model may be specified as follows (`Chapter2\ships1.aml`):

```
option title = "Poisson ship accidents example";
dsn = ships;

define regset BetaX;
   var = 1 (shiptype==2) (shiptype==3) (shiptype==4) (shiptype==5)
         year6569 year7074 year7579 oper7579;

poisson model;
   outcome = accident;
   exposure = months;
   incidence = exp(regset BetaX);

starting values;

Constant  T   -6.1300899  /* log(mean accident) - log(mean months)) */
Type2     T    0
Type3     T    0
Type4     T    0
Type5     T    0
Year6569  T    0
Year7074  T    0
Year7579  T    0
Oper7579  T    0
;
```

The Poisson model specification is very straightforward. The outcome is variable `accident`, exposure is variable `months`, and the per-period incidence rate is given by regressor set `BetaX`. The effective (per-observation) incidence rate λ_i is thus `months*exp(regset BetaX)`. The `exposure` statement is optional.[15]

Notice the starting value of the intercept, -6.1300899. Poisson models tend to be quite sensitive to good starting values. Around poor starting values, the likelihood surface is flat and the search may fail. We therefore recommend putting a little effort into finding a good starting value for the intercept. As explained in Section 6.8, the optimal intercept is equal to the difference

[15] Since $\lambda_i = E_i \exp(\beta' X_i) = \exp(\ln E_i + \beta' X_i)$, we could equivalently specify the model by including log-exposure as a variable (regressor with coefficient fixed to one) in the incidence equation:

```
poisson model;
   outcome = accident;
   incidence = exp(lnmonths + regset BetaX);
```

where `lnmonths` is a data variable that is equal to log(`months`). (Variable transformations are not permitted among building blocks.)

of the logarithms of the mean outcome and the mean exposure, $\log(\bar{Y}) - \log(\bar{E})$. These means may be found in the data summary file, `ships.sum`.

The parameter estimates are:

```
Constant          -6.4043 ***
                  (0.4297)
Type2             -0.5426
                  (0.3849)
Type3             -0.6912
                  (0.4236)
Type4             -0.0675
                  (0.3904)
Type5              0.3304
                  (0.3895)
Year6569           0.6951 ***
                  (0.2219)
Year7074           0.8115 ***
                  (0.2424)
Year7579           0.4319
                  (0.3013)
Oper7579           0.3876 **
                  (0.1578)

Ln-L             -68.42

NOTE:  Asymptotic standard errors in parentheses;
       Significance: '*'=10%;  '**'=5%;  '***'=1%.
```

It took 27 iterations to find these estimates. This strikes us as very many—the likelihood of the Poisson model is globally concave and the search should not take that many iterations. The problem is that the default search direction is quite poor. The search direction involves the matrix of second derivatives (Hessian matrix), which by default is approximated with the BHHH algorithm. This approximation can be poor in small samples, such as the sample under study (N=34). A solution is to specify "`option numerical search`" (page 270), as we have done in `Chapter2\ships2.aml`. This option computes the Hessian matrix and thus the search direction more accurately. Indeed, the program then needs only five iterations to converge. Naturally, the default (BHHH-based) and numerical search algorithms result in the same parameter estimates.[16] However, the reported standard deviations and significance levels differ. The calculation of standard deviations also involves the Hessian matrix and the default (BHHH-based) standard errors can therefore be inaccurate in small samples.

[16] Small deviations may result from an insufficiently strict convergence criterion. aML's default criterion is that the weighted gradient norm be less than 0.1. This criterion is designed to accurately pin down precisely estimated parameters but pays less attention to imprecisely estimated parameters. See "`option converge`" on page 276.

Searching on the basis of numerically computed Hessian matrices is computing-intensive, because each numerical Hessian requires *k* gradient evaluations, where *k* is the number of parameters. For some applications (with few observations but many outcomes per observation), this may be prohibitively time-consuming. In such cases, you may want to search with the default BHHH-based Hessian matrices, but request a one-time numerical evaluation of the Hessian upon convergence to accurately calculate standard errors. This is done with "`option numerical standard errors`" (page 271).

When the number of observations is small, the search direction may be poor and the reported standard errors of parameter estimates inaccurate. "`Option numerical search`" remedies the former; "`option numerical standard errors`" the latter.

The difference in standard errors can be substantial. For example, "`mktab ships1 ships2`" shows the following:

```
                ships1          ships2

Constant        -6.4043 ***     -6.4029 ***
                (0.4297)        (0.2175)
Type2           -0.5426         -0.5447 ***
                (0.3849)        (0.1776)
Type3           -0.6912         -0.6888 **
                (0.4236)        (0.3290)
Type4           -0.0675         -0.0743
                (0.3904)        (0.2906)
Type5            0.3304          0.3211
                (0.3895)        (0.2357)
Year6569         0.6951 ***      0.6958 ***
                (0.2219)        (0.1497)
Year7074         0.8115 ***      0.8175 ***
                (0.2424)        (0.1698)
Year7579         0.4319          0.4450 *
                (0.3013)        (0.2332)
Oper7579         0.3876 **       0.3839 ***
                (0.1578)        (0.1183)

ln-L            -68.42          -68.41

NOTE:  Asymptotic standard errors in parentheses;
       Significance: '*'=10%;  '**'=5%;  '***'=1%.

NOTE:  The standard errors are not all consistently computed:
```

```
Column 1 (ships1) is BHHH-based
Column 2 (ships2) is based on a numerical Hessian
```

In this case, all BHHH-based standard errors are larger than the more accurate numerical ones. This is a coincidence; BHHH-based approximate standard errors may be over- or underestimates.

2.7. Negative Binomial Model

Attention former users of aML Version 1:

There are many ways to parameterize negative binomial models. Versions 1 and 2 follow different parameterizations. See Section 13.12.1 for backward compatibility of Version 2.

This section illustrates the steps that are required to estimate a simple negative binomial model. The main steps to estimate negative binomial models are similar or identical to those for probit and Poisson models, as explained in Sections 2.1 and 2.6. We therefore only highlight differences between estimation of Poisson and negative binomial models. If you have not read Sections 2.1 and 2.6 yet, please do so now.

Much like the binomial and Poisson models, the negative binomial model applies to processes for which the outcomes are counts. As with the Poisson distribution, the negative binomial outcome is non-negative, integer-valued, and has no upper bound. The probability distribution is:

$$\Pr\left(Y=y\right)=\frac{\Gamma\left(y+\frac{1}{\alpha}\right)}{\Gamma\left(y+1\right)\Gamma\left(\frac{1}{\alpha}\right)}\theta^{\frac{1}{\alpha}}\left(1-\theta\right)^{y},$$

where $\Gamma(\cdot)$ denotes the Gamma function and

$$\theta=\frac{1}{1+E\alpha\exp\left(\beta'x\right)}.$$

E is the "exposure" and α is the "dispersion." The dispersion must be strictly positive and may be parameterized. The (observable part of the) incidence rate is $\exp\left(\beta'x\right)$. An outcome may be the result of exposure to multiple periods. The overall incidence rate is thus $E\exp\left(\beta'x\right)$. Heterogeneity is allowed in θ; see Section 13.12.

Model Specification and Estimation

Most applications in the literature estimate dispersion α as a parameter that is the same for all observations. To illustrate aML's broader capabilities, consider the following artificially generated data. Data set `Samples\Chapter2\negbin.dat` (created from `negbin.raw` and converted by raw2aml using control file `negbin.r2a`) contains outcome variable "`count`" and exposure variable "`exposure`". The incidence is a function of variables "`x1`" and "`x2`", the

dispersion is a function of variables "x3" and "x4". Let's first estimate the standard model in which the dispersion is constant across observations and the incidence rate is a function of covariates (negbin1.aml):

```
 1  dsn=negbin;
 2  option numerical search;
 3
 4  define regset BetaX;  var = 1 x1 x2;
 5  define parameter Alpha;  range=(0,Inf);
 6
 7  negative binomial model;
 8     outcome = count;
 9     exposure = exposure;
10     dispersion = par Alpha;
11     incidence = exp(regset BetaX);
12
13  starting values;
14
15  Constant    TT    0
16  x1          FT    0
17  x2          FT    0
18  Alpha       TT    1
19  ;
```

Line 2 shows a new statement:

```
option numerical search;
```

This statement affects the search direction. aML's (Gauss-Newton) likelihood maximization algorithm searches on the basis of the vector of first derivatives and the matrix of second derivatives of the log-likelihood with respect to estimated model parameters (see Section 13.1.4). By default, second derivatives are approximated using the BHHH algorithm (Berndt et al., 1974). This approximation is good for large samples, but may be poor in finite samples. We have found that it is particularly poor for negative binomial models. The "option numerical search" instructs aML to calculate second derivatives as numerical derivatives of analytically computed first derivatives. This calculation requires substantially more operations than the BHHH algorithm, but is typically still very fast and can lead to convergence in fewer iterations. See Section 13.1.4 for details.

The negative binomial model specification is straightforward. Line 8 specifies the outcome variable (or expression). Line 9 specifies the exposure variable (or expression), E. This statement is optional; by default, E=1.

Line 10 specifies the dispersion parameter, α. Line 5 defined it as being strictly positive to ensure that its value would not become zero or negative during the search. Equivalently, we could have defined a unrestricted parameter lnAlpha and specified:

```
dispersion = exp(par lnAlpha);
```

In other words, dispersion may be specified as either "`dispersion = ...`" or "`dispersion = exp(...)`". Since we are interested in α, not $\ln(\alpha)$, we opted for the former. An example of the latter follows below.

Line 11 specifies the incidence rate. It must be positive and must always be specified as an exponentiated function of building blocks. In other words, aML uses the log-link function.

Lines 15-18 specify starting values. We start the incidence parameters at zero and the dispersion parameter at one. We first let the incidence intercept and the dispersion parameter settle in before estimating other incidence coefficients.

Next, we allow the dispersion to be a function of regressors (`negbin2.aml`):

```
 1  dsn=negbin;
 2  option numsearch;
 3
 4  define regset BetaX;  var = 1 x1 x2;
 5  define regset AlphaX; var = 1 x3 x4;
 6
 7  negative binomial model;
 8     outcome = count;
 9     exposure = exposure;
10     dispersion = exp(regset AlphaX);
11     incidence = exp(regset BetaX);
12
13  starting values;
14
15  Constant   TT    0
16  x1         FT    0
17  x2         FT    0
18  Constant   TT    .87546874
19  x3         FT    0
20  x4         FT    0
21  ;
```

Line 10 specifies dispersion as

```
dispersion = exp(regset AlphaX);
```

This specification ensures that the dispersion is always positive. If you like, you could try:

```
dispersion = regset AlphaX;
```

and hope that the search does not venture into negative territory. (Try it: all goes well, albeit with one misstep that aML is able to ignore.) Of course, the two specifications are not identical. In the first case, dispersion terms enter multiplicatively, $\exp(\alpha' x)$, in the second additively, $\alpha' x$.

The model in which dispersion is a constant (`negbin1.aml`) estimated $\hat{\alpha} = 2.4$ (not shown). We therefore initialized the dispersion intercept at $\ln(2.4) = 0.875$ (line 18).

2.8. Ordered Probit and Logit Models

This section illustrates the steps that are required to estimate a simple ordered probit or ordered logit model. The main steps to estimate these types of models are similar or identical to those for probit or logit models, as explained in Sections 2.1 and 2.2. We therefore only highlight differences between estimation of probit/logit and ordered probit/logit models. If you have not read Sections 2.1 and 2.2 yet, please do so now.

Ordered probit models are probit models with multiple categorical outcomes that are ordered. They are sometimes known as normal interval models. Examples are include scores on such questions as "Would you say your health is poor, fair, good, very good, or excellent?" The five possible answers are naturally ordered. They are ordinal, not cardinal, i.e., nothing is implied by the numerical labels on categories, except their order. For example, it need not be the case that the difference between poor and fair is the same as between very good and excellent. The idea behind an ordered probit model is that some underlying propensity (here, a continuous latent health status measure) falls in certain categories. These categories are delimited by thresholds. For example, four thresholds result in five possible categories. If the propensity is distributed normally, the ordered probit is applicable; if the propensity follows the logistic distribution, the ordered logit is applicable.

Formally, an ordered probit is defined as follows. Consider outcome y which may take n+1 ordered outcomes 0 through n. The outcome is determined by propensity y^*:

$$y^* = \beta' X + u \text{, where } y = \begin{cases} 0 & \text{if} \quad y^* < \tau_1; \\ 1 & \text{if} \quad \tau_1 \le y^* < \tau_2; \\ 2 & \text{if} \quad \tau_2 \le y^* < \tau_3; \\ \vdots & \\ n & \text{if} \quad \tau_n \le y^*. \end{cases}$$

Note that the simple probit is a special case with $n = 1$ and $\tau_1 = 0$. The likelihood function is:

$$L = \begin{cases} \Phi\left(\left(\tau_1 - \beta' X\right)/\sigma_u\right) & \text{if } y = 0 \\ \Phi\left(\left(\tau_2 - \beta' X\right)/\sigma_u\right) - \Phi\left(\left(\tau_1 - \beta' X\right)/\sigma_u\right) & \text{if } y = 1 \\ \Phi\left(\left(\tau_3 - \beta' X\right)/\sigma_u\right) - \Phi\left(\left(\tau_2 - \beta' X\right)/\sigma_u\right) & \text{if } y = 2 \\ \vdots & \\ 1 - \Phi\left(\left(\tau_n - \beta' X\right)/\sigma_u\right) & \text{if } y = n \end{cases}$$

The ordered logit is defined analogously. Its residual u follows the logistic distribution, so its likelihood function involves not $\Phi\left(\left(\tau_i - \beta' X\right)/\sigma_u\right)$ but $F\left(\tau_i - \beta' X\right) = \left(1 + \exp\left(-\left(\tau_i - \beta' X\right)\right)\right)^{-1}$, $i = 1,\ldots,n$.

The data format and control file syntax of ordered probit and logit models depends on whether the thresholds are unknown (and to be estimated) or known (often in the form of data variables). Most commonly, the thresholds are unknown and need to be estimated. This case is discussed here. See Section 5.2 for ordered probit and logit models with known thresholds.

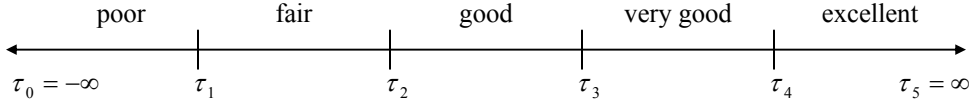
Model Specification and Estimation

Most software packages that support ordered probit models require that the outcome falls in one and only one category. For example, health status must be either poor, fair, good, very good, or excellent, and there is no room for responses that span multiple categories. aML supports the straightforward extension that allows responses to span more than one category. We first illustrate the case in which all outcomes fall in a single category. The example is an ordered probit model, but you may substitute the word "logit" for "probit" to specify ordered logit models. Their differences are the same as between simple probits and logits.

aML requires that all ordered outcomes are contiguous integer-valued numbers. This may seem self-evident, but the only conceptual requirement is that the outcomes are ordered. Health outcomes may thus be coded from 0 (poor) to 4 (excellent), or from 1 (poor) to 5 (excellent), or any other set of contiguous integers. The model statement differs slightly depending on the coding scheme you choose. Suppose you coded health status (`health`) as:

$$\texttt{health} = \begin{cases} 0 & \text{if health is poor;} \\ 1 & \text{if health is fair;} \\ 2 & \text{if health is good;} \\ 3 & \text{if health is very good;} \\ 4 & \text{if health is excellent.} \end{cases}$$

There are five categories, so we need four threshold parameters, τ_1 through τ_4, to map the underlying propensity (continuous health concept) into five categories:



We added $\tau_0 = -\infty$ and $\tau_5 = \infty$; these are not explicitly present, but greatly help in specifying the model correctly:

```
define vector Taus; dim=4;

ordered probit model;
   outcomes = health health+1;
   thresholds = Taus;
   model = <building blocks>;
```

Thresholds are to be estimated and must thus be defined as a vector building block. A vector is simply a set of parameters. Its dimension specification (`dim=4`) determines the number of parameters in the vector. By default, aML restricts vector elements to be strictly increasing, which is precisely what we want for ordered probit thresholds. For full details see Section 13.2.4.

The outcome of an ordered probit model is specified in the form of two variables or expressions. The reason for this seemingly redundant specification is in aML's ability to handle outcomes that span multiple categories; see below. The two variables or expressions indicate the subscript numbers of the thresholds between which the outcome lies. For example, as shown in the figure above, health is classified as fair (`health=1`) when the health propensity is between τ_1 and τ_2. the outcomes must thus evaluate to 1 and 2, which is achieved by "`outcomes = health health+1`". Note that $\tau_0 = -\infty$ and $\tau_5 = \infty$ are needed to deal with the extreme categories, poor (between $\tau_0 = -\infty$ and τ_1) and excellent (between τ_4 and $\tau_5 = \infty$).

If we had coded the health outcomes not from 0 to 4, but from 1 (poor) to 5 (excellent), the outcomes would need to be specified as "`outcomes = health-1 health`". You are, of course, free to create two data variables to indicate the outcome. The lowest value of the lower outcome must be zero and the highest value of the upper outcome must be equal to the number of categories.

If there are *n* categories in the data, you need a vector with dimension *n*-1 to represent the thresholds, τ_1 through τ_{n-1}. The outcome in an ordered probit or logit model is specified by two variables or expressions which evaluate to the subscript numbers of the thresholds between which the outcome lies. The *j*-th category lies between τ_{j-1} and τ_j; the lowest category lies between implicitly defined $\tau_0 = -\infty$ and τ_1; the highest category between τ_{n-1} and implicitly defined τ_n. In other words, the lowest value of the lower outcome must be zero and the highest value of the upper outcome must be equal to the number of categories.

A simple probit model requires two normalizations; a zero threshold and a standard normally distributed residual. Section 5.1 below shows how to specify non-zero thresholds and non-standard residuals. Similarly, an ordered probit model requires that you either do not specify an intercept, or fix one threshold to any fixed value. In other words, you cannot estimate both all thresholds and an intercept. (Try it; aML will keep one parameter at its initial value.) Also, if you do not specify a (non-integrated) residual among the model's building blocks, aML assumes that you want an *iid* N(0,1) residual.

Outcomes May Span Multiple categories

aML also supports ordered probit and logit models in which the outcome may span several categories. For example, an ordered outcome may take on values 0 through 4 and we wish to

estimate an ordered probit model with four thresholds. However, for some observations the outcome may be either 3 or 4. The likelihood is:

$$\begin{aligned} L &= \{\Phi(\tau_3 - \beta'X) - \Phi(\tau_2 - \beta'X)\} + \{1 - \Phi(\tau_3 - \beta'X)\} \\ &= 1 - \Phi(\tau_2 - \beta'X) \end{aligned}.$$

More generally, the probability that an outcome ($y = 0,\ldots,n-1$) is in the interval (*i, ..., j*) is

$$L = P(i \le y \le j) = \Phi(\tau_{j+1} - \beta'X) - \Phi(\tau_i - \beta'X),$$

where implicitly $\tau_0 = -\infty$ and $\tau_n = \infty$. The model is specified analogously to the case where outcomes could only be in a single category, as discussed above. However, since the outcome may lie between two thresholds that are not contiguous, the outcome must be specified more generally:

```
ordered probit model;
   outcomes = lowvar upvar;
   thresholds = <vectorname>;
   model = <building blocks>;
```

where *lowvar* and *upvar* are data variables (or expressions) which indicate the threshold numbers between which the outcome lies. The difference between those variables, *upvar–lowvar*, is equal to the number of categories between which the outcome lies.

As mentioned above, Section 5.2 discusses how to estimate ordered probit or ordered logit models when the thresholds are known and data-dependent. Sections 13.6 and 13.8 contain many more details on ordered probit and ordered logit models, respectively.

2.9. Tobit Model

This section illustrates the steps that are required to estimate a simple tobit (censored normal density) model. The main steps to estimate tobit models are similar or identical to those for probit and continuous models, as explained in Sections 2.1 and 2.3. We therefore only highlight differences between estimation of probit or continuous models on the one hand and tobit models on the other. If you have not read Sections 2.1 and 2.3 yet, please do so now.

In some models, one only observes a continuous outcome if that outcome falls in a certain range. If the outcome is outside that range, there are two possibilities. First, one may not observe anything about those cases, in which event the truncated normal model is appropriate (Section 5.3). Second, one may observe everything about those cases, except for the outcome. In that event, the censored normal density model, better known as the Tobit model (Tobin 1958; Judge et al., 1988) applies.

Consider an analysis of the number of hours worked in a year. If we were to analyze current workers, we would only observe individuals with strictly positive number of hours worked. Nothing would be known about individuals who do not work in the formal sector. Under those circumstances, a truncated normal model may be appropriate. If instead we have a sample of both workers and non-workers, we observe both positive and zero hours worked, and a Tobit model may apply. We illustrate a Tobit model on such data.

We use the "`Samples\Chapter2\work.dat`" data. The variable of interest, annual hours worked, is "`hours`". It may be zero or positive. Other variables of interest are education (coded as `educ=1` for individuals without a high school diploma, `educ=2` for high school graduates, and `educ=3` for college graduates) and the number of young children that the respondent has at home, `children`. There is only one record per person.

The model is:

$$h = \begin{cases} 0 & \text{if } \beta' x + v \le 0 \quad \text{(the individual does not work);} \\ \beta' x + v & \text{if } \beta' x + v > 0 \quad \text{(the individual does work).} \end{cases}$$

The likelihood function is:

$$L = \begin{cases} \Phi\left(\dfrac{-\beta' x}{\sigma_v}\right) & \text{if } h \le 0; \\ \phi\left(\dfrac{h - \beta' x}{\sigma_v}\right) & \text{if } h > 0, \end{cases}$$

where Φ and ϕ denote the cumulative normal probability function and the normal density function, respectively.

Model Specification and Estimation

The following control file (`Samples\Chapter2\tobit.aml`) defines the building blocks and specifies the tobit model:

```
 1  dsn = work;
 2
 3  define regressor set BetaX;
 4     var = 1 (educ==1) (educ==3) children;
 5
 6  define normal distribution; dim=1;
 7     name=v;
 8
 9  tobit model;
10     outcome = hours;
11     lower limit=0;
12     model = regset BetaX + res(draw=1, ref=v);
13
14  starting values;
15
16  Constant   T    1377
17  dropout    T    0
18  college    T    0
19  children   T    0
20  SigmaV     T    681
21  ;
```

Line 1 specifies the input data set, as created by raw2aml and discussed in Section 2.1.

Lines 3-4 define a regressor set which contains the explanatory covariates. Note that we use variable transformations. Education categories 1 and 3 denote high school drop-outs and college graduates, respectively.

Lines 6-7 define a univariate normal distribution, as discussed above for continuous models (Section 2.3).

Lines 9–12 specify the model:

```
tobit model;
   outcome = hours;
   lower limit=0;
   model = regset BetaX + res(draw=1, ref=v);
```

The model specification is very similar to that of a continuous model (Section 2.3), with an additional statement for the tobit threshold, "`lower limit=0;`". This statement indicates that the outcome is left-censored, i.e., censored from below. The lower limit may be specified as any expression of variables (of at least the same level of aggregation as the outcome variable). For example, "`lower limit=abc`", where `abc` is a variable name, is permissible, as is "`lower limit=15*sqrt(abc-16)`", should that make any sense. (Expressions are discussed in Section

13.9.) aML also supports right-censored tobit models (`upper limit=...;`) and dual-censored models; see Section 13.13.[17]

Lines 14-21 specify the parameter values that aML will use in its first iteration of the search process:

```
starting values;

Constant   T    1377
dropout    T    0
college    T    0
children   T    0
SigmaV     T    681
;
```

The first four parameters correspond to the regressors in regressor set "`BetaX`". We initialized the intercept to the mean outcome (see `work.sum`). Since the outcome is censored from below, the true mean will be less than 1377, but the censored mean is not a bad starting value. Similarly, we initialized the standard deviation of `v` at the standard deviation of the censored outcome (681), even though the uncensored standard deviation will be greater. (The uncensored mean and standard deviation may be found by running the model with an intercept but without covariates.)

File "`tobit.out`" contains the output. Its structure is very similar to output files from other model types. The main difference is in information about the outcome (lines 53-66):

```
 53     Summary statistics of the outcome and selected variables:
 54
 55     Tobit outcome type    |       Freq.    Percent
 56     ----------------------+------------------------
 57     outcome < lower limit |          0        0.00
 58     outcome = lower limit |        342       30.37
 59     uncensored outcome    |        784       69.63
 60     ----------------------+------------------------
 61                     Total |       1126      100.00
 62
 63     Uncensored outcomes:
 64
 65              |      #        Mean     Std Dev         Min         Max
 66     ---------+-------------------------------------------------------
```

[17] As discussed elsewhere, aML supports multivariate normal distributions. Tobit models with residuals from a multivariate normal distribution will be censored in multiple dimensions, leading to multivariate tobit models. There needs to be a tobit model specification for every dimension; make sure that the residual draws are the same in all such models, so that the residuals are correlated (e.g., Section 13.3.6). Multivariate tobits are supported up to trivariate. Experienced users with a need for higher-dimensional tobit models may specify independent residuals with additional residuals that induce correlation and that are integrated-out.

```
 67       outcome  |     784     1377.337      680.737          8         3227
```

The first table shows the extent to which the outcome is censored. There are no cases for which the outcome is smaller than the lower limit. We would not expect any such cases here, but there may be applications for which the lower limit is not truly the lowest value in the data. aML will treat such cases as censored at the user-specified lower limit and proceed without complaints, but report their frequency. The table may also serve to catch errors in data preparation or model specification. In our application, 30 percent of the sample was censored (worked zero hours) and 70 percent uncensored (worked some positive number of hours).

The second table provides more detail about the 784 uncensored outcomes.

Section 13.13 contains many more details about (multilevel) tobit models.

2.10. Multinomial Logit Model

This section illustrates the steps that are required to estimate a multinomial logit model. The main steps are similar or identical to those for logit models, as explained in Section 2.2. We therefore only highlight differences between estimation of logit and multinomial logit models. If you have not yet read Section 2.2, please do so now.

Multinomial logit models are appropriate when the outcome of interest may take on a limited number of values that are not ordered. (If they are ordered, the ordered logit or ordered probit model is more appropriate.) For example, someone may choose among various modes of transportation (walking, bicycle, own motorized transportation, public transportation); a household may own a residency, rent a residency, or be otherwise accommodated; a worker may be employed in agriculture, manufacturing, services, or other industry; et cetera. The categories must be mutually exclusive and exhaustive, possibly through the inclusion of a residual category.

Denote the probability that choice j (j=1,…,J) is selected by P_j. The likelihood function is:

$$P_j = \frac{\exp\{\beta_j' X\}}{\sum_{i=1}^{J} \exp\{\beta_i' X\}}, \quad j = 1,...,J,$$

where J is the number of choices. This model is defined by J sets of parameters β_j. However, not all parameters are statistically identified. We typically normalize all parameters associated with a certain category to be zero.[18] Suppose this omitted category is the J-th category ($\beta_J = \underline{0}$), so that the likelihood function reduces to:

$$P_j = \begin{cases} \dfrac{\exp\{\beta_j' X\}}{1 + \sum_{i=1}^{J-1} \exp\{\beta_i' X\}} & \text{if } j = 1,...,J-1; \\ \dfrac{1}{1 + \sum_{i=1}^{J-1} \exp\{\beta_i' X\}} & \text{if } j = J \text{ (omitted category).} \end{cases}$$

The choice of omitted category is entirely arbitrary. Also, while the potential choices in our example are numbered consecutively from 1 to J, no order is implied and any set of integer-valued

[18] aML does not require that you normalize in this way. You may specify an exhaustive set of choices and impose any number of identifying restrictions in the starting values. Alternatively, in a multiprocess model, you could identify one set of parameters in a related model and estimate the full set of J parameters β_j. While multinomial probit models (Section 2.6) are very similar to multinomial logit models, multinomial probit models must have an omitted category.

choices may be specified (see below). Indeed, multinomial models treat all categories on the same footing; any reversal of assigned categories results in equivalent estimates.

Most model estimation software packages require that the same set of explanatory covariates X enters in all choice equations. While not expressed in the above likelihood equations, aML is more general in that it accepts different regressors in all choice equations.

Interpretation of multinomial logit parameters is not straightforward. For example, the derivative of the probability that choice j will be selected with respect to the k-th element of covariate vector X is:

$$\frac{\partial P_j}{\partial X_k} = P_j\left(\beta_{jk} - \sum_{i=1}^{J}\beta_{ik}P_i\right),$$

where β_{ik} is the k-th element of coefficient vector β_i. This probability depends on the point of evaluation, just like it does in the standard logit model. However, it depends on all probabilities P_1 through P_J, and can change signs depending on those probabilities. In other words, the multinomial logit model does not share the monotonicity property of standard logit models, where larger values of a covariate with a positive coefficient imply larger values of the probability.

Many people find it easier to interpret multinomial logit parameters through the concept of log-odds ratios. For example, the log-odds ratio of categories j and k is:

$$\log\left(\frac{P_j}{P_k}\right) = \log\left(\frac{\exp\left(\beta_j'X\right)\Big/\sum_{i=1}^{J}\exp\left\{\beta_i'X\right\}}{\exp\left(\beta_k'X\right)\Big/\sum_{i=1}^{J}\exp\left\{\beta_i'X\right\}}\right) = \left(\beta_j - \beta_k\right)' X .$$

This log-odds ratio depends only on parameters β_j and β_k. If the choice set were to expand or contract from the current J options, the log-odds ratio of categories j and k would not be affected. This property is known as independence from irrelevant alternatives (IIA). It is an undesirable property, because it also applies to highly relevant additional alternatives.[19] The multinomial probit model, by contrast, does not suffer from this property (Section 2.11).

[19] The standard example is the "red bus, blue bus" example. Suppose one chooses among several modes of transportation, say, non-motorized (n), own motorized (o), and public transportation by bus (r), with certain probabilities. Suppose all buses are red. The odds ratio of red bus versus, say, non-motorized transportation, P_r/P_n, reflects the relative preference of riding a red bus versus using non-motorized transportation. Now suppose a new public bus service (b) is introduced that is identical to the existing bus service, except that its buses are blue in color. We estimate a new multinomial logit model that includes the new blue bus service as a separate choice. The interpretation of the log-odds ratio of red bus versus non-motorized transportation, P_r/P_n, remains the same as before, namely the relative preference of riding a red bus versus using non-motorized transportation. Its value *should* therefore not change. However, no change in the odds ratio can only occur if the blue buses gain market share by proportional decreases of the market

2.10.1. Model Specification and Estimation

As an example, consider occupational choice. We distinguish white collar jobs (occ=1), clerical jobs (occ=2), services (occ=3), and blue collar jobs (occ=4). The outcome is distributed as follows:

```
        occ |      Freq.     Percent
------------+-----------------------
          1 |       569       28.45
          2 |       531       26.55
          3 |       344       17.20
          4 |       556       27.80
------------+-----------------------
      Total |      2000      100.00
```

Given these four categories, we wish to estimate three sets of parameters. Explanatory variables are sex, educational attainment, and health status. The following control file (Samples\Chapter2\mlogit.aml) defines the building blocks and specifies a multinomial logit model:

```
 1  dsn=occ;
 2
 3  define regset Clerical;   var= male (educyrs<12) (educyrs>=16) (health<=2) 1;
 4  define regset Services;   var= male (educyrs<12) (educyrs>=16) (health<=2) 1;
 5  define regset BlueCollar; var= male (educyrs<12) (educyrs>=16) (health<=2) 1;
 6
 7  multinomial logit model;
 8     outcome=occ;
 9     model 2 = regset Clerical;
10     model 3 = regset Services;
11     model 4 = regset BlueCollar;
12
13  starting values;
14
15  male      T    0
16  dropout   T    0
```

shares of other transportation modes. This is implausible. It is much more likely that blue buses dent the market share of red buses by far more than of other modes. Indeed, in practice, the introduction of a new choice that is similar to an existing choice does affect the odds ratios. This unpatable result is a consequence of the fact that the multinomial probit treats all outcomes on the same footing, without any account of similarities or dissimilarities among potential outcomes. Then why do analysts not worry very much about the undesirable IIA property? Key is to interpret the coefficients in the context of the empirical setting. In other words, instead of assigning a very specific interpretation to an odds ratio ("relative preference of riding a red bus versus using non-motorized transportation"), one should interpret the coefficients as reflective of relative preferences among the alternatives offered. If some alternatives are clearly very similar, consider estimating a multinomial probit model (Section 2.11) or a nested logit model (not supported by aML).

```
 17  college   T    0
 18  goodhlth  T    0
 19  constant  T    0
 20  male      T    0
 21  dropout   T    0
 22  college   T    0
 23  goodhlth  T    0
 24  constant  T    0
 25  male      T    0
 26  dropout   T    0
 27  college   T    0
 28  goodhlth  T    0
 29  constant  T    0
 30  ;
```

Line 1 specifies the input data set, as created by raw2aml and discussed in Section 2.1.

Lines 3-5 define regressor sets with explanatory covariates. Note that we use variable transformations. Variable `educyrs` contains years of educational attainment, so that `(educyrs<12)` and `(educyrs>=16)` capture high school drop-outs and college graduates, respectively. Variable `health` represents general health status, ranging from 1 (excellent) to 5 (poor). Transformation (health<=2) thus captures excellent or very good health.

Lines 7–11 specify the model:

```
multinomial logit model;
   outcome = occ;
   model 2 = regset Clerical;
   model 3 = regset Services;
   model 4 = regset BlueCollar;
```

The model specification is very similar to that of a logit model (Section 2.2), except that there now are multiple model statements, each tagged with an integer number. These integers correspond to the categories being modeled. Outcome variable `occ` takes on values 1, 2, 3, and 4. We specify models for values 2, 3, and 4, i.e., the omitted category is 1. More than one value may be omitted: any value for which there is no model statement ends up in the residual category. As mentioned above, the choice of omitted category is arbitrary. We chose the most common category, but we could have specified a model for outcome 1 and omitted one (or more) of the others.

The three regressor sets all contain the same variables. This results in the standard multinomial model, as supported by several other software packages. However, aML does not require that all variables are the same. Should this make sense, you may specify different variables. (Similarly, you may specify different residual structures—in multilevel or multiprocess contexts, aML permits residuals in multinomial probits, exactly in the same way as in other types of models. See Chapter 4, Section 13.6, and elsewhere throughout in this manual.)

Lines 13-30 specify the parameter values that aML will use in its first iteration of the search process. As always, they appear in the order of building block definitions. The first five

parameters correspond to regressor set `Clerical`, the next five to `Services`, and the last five to `BlueCollar`. For lack of better ideas, we initialized all parameters to zero. We could have been smarter about initial values for the intercepts (see Section 6.10), but the multinomial logit model is well-behaved so we did not bother.

File "`mlogit.out`" contains the output. Its structure is very similar to output files from other model types. The main difference is in information about the model and the outcome (lines 61-77):

```
 61  multinomial logit model;
 62     outcome = occ;
 63     model 2 = regset Clerical;
 64     model 3 = regset Services;
 65     model 4 = regset BlueCollar;
 66     /*  Omitted category = 1  */
 67
 68     Summary statistics of the outcome and selected variables:
 69
 70         outcome |       Freq.    Percent
 71     ------------+------------------------
 72               1 |        569       28.45
 73               2 |        531       26.55
 74               3 |        344       17.20
 75               4 |        556       27.80
 76     ------------+------------------------
 77           Total |       2000      100.00
```

The output file restates the model specification. It adds a comment, "`Omitted category = 1`". This serves as a reminder or a check. The output file also contains a tabulation of the outcome values in the data.

2.10.2. Coding of Outcomes

The mathematical notation above assumed that outcomes are coded with values 1 through *J*. In the example, the outcomes followed this scheme with codes 1, 2, 3, and 4. However, any other set of unique integer-valued codes is equivalent. There is no need for category codes to be consecutive, no need for a numbering scheme that involves a zero or one. For example, if occupation code `occ` were coded 26 (white collar), 7 (Clerical), -52 (Services), and 14 (blue collar), the following model specification would yield exactly the same results as the above specification:

```
multinomial logit model;
   outcome = occ;
   model 7 = regset Clerical;
   model -52 = regset Services;
   model 14 = regset BlueCollar;
```

What would happen if a model equation is specified for every possible outcome, i.e., if there were no omitted category? Such a model is not identified. aML would write out a warning to that effect but continue nevertheless, under its usual assumption that you know what you are doing. It may be, for example, that the parameters of one equation are identified off other parts of a multiprocess model, or identification may be derived from non-zero restrictions, or from equality restrictions across equations, or from some other restriction. In short, it is the responsibility of the user to ensure that the model is identified.

2.11. Multinomial Probit Model

This section illustrates the steps that are required to estimate a multinomial probit model. The main steps are similar or identical to those for simple probit or multinomial logit models, as explained in Sections 2.1 and 2.10. We therefore only highlight differences between estimation of simple probit and multinomial logit models on the one hand and multinomial probit models on the other. If you have not yet read Sections 2.1 and 2.10, please do so now.

Multinomial probit models are appropriate when the outcome of interest may take on a limited number of values that are not ordered. (If they are ordered, the ordered probit model is more appropriate.) For example, someone may choose among various modes of transportation (walking, bicycle, own motorized transportation, public transportation); a household may own a residency, rent a residency, or be otherwise accommodated; a worker may be employed in agriculture, manufacturing, services, or other industry; et cetera. The categories must be mutually exclusive and exhaustive, possibly through the inclusion of a residual category.

The main difference between a multinomial probit and a multinomial logit model is that the probit allows correlation among its residuals. This feature enables it to capture the degree of similarity across alternatives, so that the multinomial logit's undesirable independence of irrelevant alternatives (IIA) property does not apply (see page 79). Unfortunately, the multinomial probit allows only up to four alternatives (including the omitted category); the multinomial logit does not have this limitation.

The idea behind a multinomial probit is as follows. Consider an outcome that may take four distinct values. Each alternative has a pay-off (utility, value, degree of attractiveness):

$$y_1^* = \beta_1' X + u_1$$
$$y_2^* = \beta_2' X + u_2$$
$$y_3^* = \beta_3' X + u_3$$
$$y_4^* = \beta_4' X + u_4$$

The alternative with the highest pay-off is chosen:

$$Y = j \quad \text{if} \quad y_j^* = \max\left[y_1^*, y_2^*, y_3^*, y_4^* \right].$$

We only observe which alternative is chosen. This implies that two normalizations are needed. Only relative statements can be made, so the pay-off of one alternative needs to be fixed, typically by setting its value to zero. For example, we could normalize $y_4^* = 0$. Here we arbitrarily normalized the last category, i.e., the last category is the omitted category. Any other omitted category would yield equivalent results. Since the magnitude of pay-offs is unknown and irrelevant, the scale needs to be normalized, typically by setting the standard deviations of residuals to one: $\sigma_1 = \sigma_2 = \sigma_3 = 1$, where σ_j is the standard deviation of u_j.

The coding of alternatives and the choice of omitted category are entirely arbitrary. We coded the alternatives 1, 2, 3, and 4, but any other set of integer-valued codes is acceptable. Regardless of the codes assigned to alternatives, no order is implied.

2.11.1. Model Specification and Estimation

As an example, consider the occupational choice problem already discussed in Section 2.10.1. We distinguish white collar jobs (occ=1), clerical jobs (occ=2), services (occ=3), and blue collar jobs (occ=4). Given these four categories, we wish to estimate three sets of parameters. Explanatory variables are sex, educational attainment, and health status. The following control file (Samples\Chapter2\mprobit.aml) defines the building blocks and specifies a multinomial probit model:

```
 1  dsn=occ;
 2
 3  define regset WhiteCollar; var=male (educyrs<12) (educyrs>=16) (health<=2) 1;
 4  define regset Clerical;    var=male (educyrs<12) (educyrs>=16) (health<=2) 1;
 5  define regset Services;    var=male (educyrs<12) (educyrs>=16) (health<=2) 1;
 6
 7  define normal distribution; dim=3;
 8     name=u1;  name=u2;  name=u3;
 9
10  multinomial probit model;
11     outcome = occ;
12     model 1 = regset WhiteCollar + res(draw=1, ref=u1);
13     model 2 = regset Clerical    + res(draw=1, ref=u2);
14     model 3 = regset Services    + res(draw=1, ref=u3);
15
16  starting values;
17
18  male      FTT    0
19  dropout   FTT    0
20  college   FTT    0
21  goodhlth  FTT    0
22  constant  TTT    0
23  male      FTT    0
24  dropout   FTT    0
25  college   FTT    0
26  goodhlth  FTT    0
27  constant  TTT    0
28  male      FTT    0
29  dropout   FTT    0
30  college   FTT    0
31  goodhlth  FTT    0
32  constant  TTT    0
33  SigmaU1   FFF    1
34  SigmaU2   FFF    1
35  SigmaU3   FFF    1
36  RhoU1U2   FFT    0
37  RhoU1U3   FFT    0
38  RhoU2U3   FFT    0
```

```
39  ;
```

Line 1 specifies the input data set, as created by raw2aml and discussed in Section 2.1.

Lines 3-5 define regressor sets with explanatory covariates, exactly as in the multinomial logit example (Section 2.10.1).

Lines 7-8 define a trivariate normal distribution. So far, we have only encountered univariate distributions. The multivariate extension is very straightforward: just specify the dimension and define a residual for every dimension. In this case, we define three residuals `u1`, `u2`, and `u3`.

Lines 10-14 specify the model:

```
multinomial probit model;
   outcome = occ;
   model 1 = regset WhiteCollar + res(draw=1, ref=u1);
   model 2 = regset Clerical    + res(draw=1, ref=u2);
   model 3 = regset Services    + res(draw=1, ref=u3);
```

The model specification is very similar to that of a multinomial logit model (Section 2.10), except that there now are explicitly specified residuals. As in multinomial logit models, there is a model statement for every alternative, except for the omitted category. As mentioned above, the choice of omitted category is arbitrary.

Residuals `u1`, `u2`, and `u3` enter in the three choice equations. A potentially important feature of multinomial probit models is the ability to have correlated residuals across the alternatives. As introduced in Section 2.3, correlation is present for residuals that have the same "draw." Think of a draw as a realization of a distribution. Here we have a trivariate distribution. Each draw generates a set of values for `u1`, `u2`, and `u3`. These values are correlated, following the covariance structure of the trivariate distribution. Since we specified the same draw for u1, u2, and u3 in the three equations, the residuals are correlated. Section 4.1 explains residuals in elaborate detail. It may be summarized as follows:

Residuals are correlated across equations if and only if (a) they were defined as part of the same multivariate distribution and (b) they have the same draw.

The three regressor sets all contain the same variables. This results in the standard multinomial probit model, as supported by several other software packages. However, aML does not require that all variables are the same. Should this make sense, you may specify different variables. (Similarly, you may specify different residual structures—in multilevel or multiprocess contexts, aML permits additional residuals in multinomial probits, exactly in the same way as in other types of models. See Chapter 4, Section 13.15, and elsewhere throughout in this manual.)

Lines 18-38 specify the parameter values that aML will use in its first iteration of the search process. As always, they appear in the order of building block definitions. The first five parameters correspond to regressor set `WhiteCollar`, the next five to `Clerical`, and the last five to `Services`. For lack of better ideas, we initialized all regression and correlation parameters to zero. Unlike multinomial logit models, multinomial probit models are somewhat difficult to estimate, so we need to be careful in the search process. Using multiple rounds of optimization (see Section 2.1.6), we first estimate intercepts only; then intercepts and regressors; and finally intercepts, regressors, and correlations.

In practice, estimating correlations can be very tedious. This is true in many model types, not just multinomial probit models, but multinomial probit models can be particularly challenging. Essentially, you are asking quite detailed information about unobserved things, based on data that only reveal rankings. Before freeing up correlations, always estimate everything else (regressors) first, as in the example. If you still encounter difficulty in the correlation stage, we recommend that you estimate regressors first, then free up correlations but fix regressors again, then free up correlations and intercepts, then free up everything. More generally, estimating correlations may require freeing up only a few parameters at a time and carefully guiding the process until everything is freely estimated. It is a good idea to monitor the weighted gradient norm during the search process. It is supposed to decrease throughout the search process. It may increase every now and then, but if it increases in two consecutive iterations, consider stopping the search and trying alternative subsets of freely estimated parameters. Unfortunately, simple rules are not always sufficient to find the likelihood function's maximum. Sometimes it takes patience and experience.

2.11.2. Coding of Outcomes

The mathematical notation above assumed that outcomes are coded with values 1 through *J*. In the example, the outcomes followed this scheme with codes 1, 2, 3, and 4. However, any other set of unique integer-valued codes is equivalent. There is no need for category codes to be consecutive, no need for a numbering scheme that involves a zero or one. For example, if occupation code `occ` were coded 26 (white collar), 7 (Clerical), -52 (Services), and 14 (blue collar), the following model specification would yield exactly the same results as the above specification:

```
multinomial probit model;
   outcome = occ;
   model 26 = regset WhiteCollar + res(draw=1, ref=u1);
   model 7 = regset Clerical    + res(draw=1, ref=u2);
   model -52 = regset Services    + res(draw=1, ref=u3);
```

What would happen if a model equation is specified for every possible outcome, i.e., if there were no omitted category? Such a model is not identified. In multinomial logit models, aML would write out a warning to that effect but continue nevertheless, under its usual assumption that

you know what you are doing. By contrast, the algorithms of the multinomial probit model require that there is an omitted category.

3. Data Preparation

3.1. Overview

3.1.1. Introduction

This chapter provides an overview of the data preparation process and general strategies for data preparation. It assumes that you are familiar with the basics as described in Section 2.1. If you have not read that section, please do so now.

Chapter 2 gave simple examples of each model type that aML supports. With the exception of hazard models with time-varying covariates, the sample data were trivially simple, with just a single level. The example of a hazard model with time-varying covariates introduced data with two levels (Section 2.4.4). The current section explains more generally how to construct data with any number of levels. In addition, it introduces so-called data structures, which offer a particularly convenient way to organize data with information on multiple processes.

The process of preparing data for analysis with aML is an integral part of the overall modeling and estimation process because the form of the data directly affects the model and interpretation of results. Ideally, the structure of the data clearly reflects the model(s) to be estimated. While for any particular model the data may be prepared in several ways to achieve the same result, some forms are easier to use and interpret. More importantly, the data must be in a form appropriate to the intended model to be estimated.

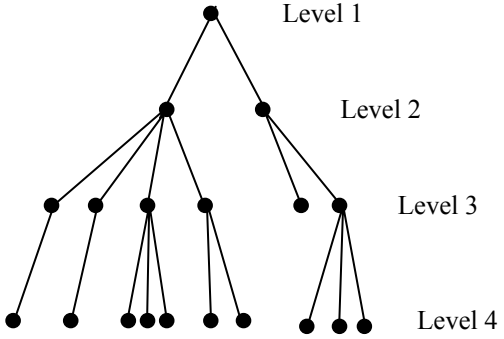
Apart from the ability to transform data variables on the fly, aML has few data manipulation features. You should therefore select your sample, create outcome measures, clean explanatory covariates, and otherwise prepare your data using a third-party data management package such as SAS, Stata, SPSS, or other. aML data preparation relates to the process of converting these (SAS, Stata, SPSS) data into a format that is most efficient to aML. This is done by writing out the data in ASCII format and pre-processing them with raw2aml, an important program that is bundled with the aML package. Its inputs are a control file, one or more ASCII data files, and, optionally, an ID file to link observations across files. Its outputs are a data set (in binary format), and a summary data documentation file to be read by the user. The output data set includes not just the actual data, but also variable names and information about the data's internal organization.

Before getting into the specifics of data preparation, we define multilevel data and discuss various types of files, data constructs, and types of variables. Section 3.2 describes how to create data with multiple levels and/or for multiple processes. Section 3.3 describes how to deal with so-called rectangular data, i.e., data that appear to have the same number of subbranches for every branch in ASCII form, but that are in fact unbalanced. Finally, as a general rule, you should organize your data into levels corresponding to their conceptual nesting patterns, but there are

circumstances under which you may save yourself some programming effort. Section 3.4 reflects on such circumstances and suggests alternative ways to organize your data into levels. Section 3.5 discusses missing data and character variables.

Multilevel Data and Multilevel Models

Multilevel data are data that are nested in some natural manner. For example, we may have data on a sample of schools; each school contains one or more students; each student attended the school one or more years; and there are zero or more tests in each year in school. The outcome of interest may be, for example, student test scores, but since there may be correlation across test scores in a particular school, the unit of observation is a school. In aML, the unit of observation is always level 1. Level 1 is thus the highest level, the most aggregated level. In the example, a school is level 1, a student level 2, a year in school level 3, and a test level 4.



Multilevel Data

The most aggregated data unit is level 1 in aML. It represents the highest level. Level 1 is thus the unit of observation. There may be zero or more level 2 units ("branches") in each level 1 unit, zero or more level 3 branches in each level 2 branch, et cetera. See the figure. Lower level branches are often called "replications," and outcomes in those branches "repeatedly observed." Careful: some other multilevel software packages assign level 1 to the most disaggregated level, i.e., they count levels in reverse order. The order has no practical implications.

Before you create ASCII files with data to be converted by raw2aml, you need to decide how many levels your data have, and which variables are at which level. Multilevel data have a natural nesting pattern which should tell you how many levels there are, but you have a fair amount of discretion. In particular, *aML does not require that a three-level model be based on three-level data*. You can estimate almost any level model on data with just two levels. Also, you can estimate a one-level or two-level model on data with lots of levels.

This may sound confusing but will actually make the data creation process much simpler. The trick is to distinguish between a conceptual and a technical number of levels. A four-level model typically requires data with four conceptual levels. However, it is often convenient to create a data set with just two (technical) levels. We like to think of levels as subscripts. For example, let S_{ijtk} be the test score of school i, student j, year t, test k. You may create data with just two levels (school and test), as long as you include identifiers i, j, t, and k as variables in the

data, to keep track of the nesting structure. A four-level model explaining S_{ijtk} may be correctly specified by using the variables that correspond to i, j, t, and k.[20]

To successfully model a multilevel issue, you must have a very clear understanding of the conceptual level structure. The best way to structure aML data is, almost always, the same as the way your (SAS, Stata, SPSS) data are organized. If the unit of observation in your (SAS, Stata, SPSS) data is a test, so that many observations pertain to the same school, then it will be most convenient to write out one ASCII record per (SAS, Stata, SPSS) observation. Each record is a level 2 branch, tied together by a school ID. The result is a perfectly fine data set with two levels.

The number of levels of your model and your data need not be the same. Almost always, it is best to maintain the organization of your (SAS, Stata, SPSS) data. It is rarely needed to do much data manipulation.

Section 3.4 elaborates on this issue and spells out the very few exceptions under which you must keep levels distinct. The raw2aml control file, discussed below, must indicate which variables are at which (technical) level.

3.1.2. Types of Files

ASCII Data Files

ASCII data files contain the actual data that are the input to raw2aml. You create them using your favorite data management package (SAS, Stata, SPSS, or other). Each raw ASCII file must conform to a specific syntax described below. There may be multiple raw ASCII files used to create one aML data file. If more than one is used then they are linked by the observation ID, perhaps in an ID file described below. A raw ASCII file may contain more than one data structure (defined below) and a data structure may appear in more than one raw ASCII file. When multiple raw ASCII files are used, each must be sorted by ID.

Raw data files may be either in "compressed" or "rectangular" format. These terms refer to multilevel data that are "unbalanced." In "balanced" multilevel data, there is the same number of subbranches under a certain branch. For example, there are five level 2 branches in every observation, and each of those five level 2 branches has two level 3 subbranches, and each of those ten level 3 branches has four level 4 subbranches. There are very few multilevel data sets that are balanced. Most have varying numbers of subbranches, i.e., most are unbalanced. The

[20] The ability to store four conceptual levels of data in two technical levels opens the door to cross-classified problems. For example, we could study test scores in schools with students and teachers, organized such that students and teachers are not nested. See Section 3.4.1.

figure above, for example, shows unbalanced data. If the ASCII data contain varying numbers of subbranches, we call them "compressed." If the ASCII data contain a fixed number of subbranches for every observation and every branch, they are "rectangular." If the data truly are balanced, there is no difference between the compressed and rectangular formats. However, if the data really are unbalanced, the rectangular data contain lots of irrelevant filler branches. That seems an inefficient way to store data, but sometimes you have little choice but to write out those irrelevant branches along with relevant information. Raw2aml handles either format, as explained in detail in Section 3.3.

ID File

aML supports multiprocess models. It is often convenient to create raw data from separate processes into separate ASCII files. An ID file may be used to link together multiple raw ASCII files into one final aML data file. When multiple raw ASCII files are linked by an ID file, it and each of the raw ASCII files must be sorted by ID.

Raw2aml Control File

The raw2aml control file controls the execution of raw2aml, which converts raw ASCII data into an aML data file to be used for analysis. It specifies the name(s) of ASCII input file(s), the optional ID file, and lists variable names at each level and, optionally, for each data structure. For rectangular data, it also tells raw2aml how many subbranches to read. (Information on the number of relevant subbranches is observation-specific and must thus be part of the ASCII data, just like is the case for compressed data.) We recommend extension "`.r2a`" for the raw2aml control file.

Raw2aml Output Files: aML-formatted Data File and Data Summary File

Raw2aml uses raw ASCII data file(s) and the ID file, if any, to create data for aML which are efficient for model formulation and estimation. Raw2aml also generates a human-readable data summary file for user documentation. By default, aML-formatted data file names have extension "`.dat`"; the corresponding file with summary statistics has extension "`.sum`".

3.1.3. Data Structures

aML supports multiprocess modeling with various types of outcomes and different sets of explanatory variables. It is often convenient to organize one's data into several data structures (subsets of variables), each dealing with one or more processes. A data structure is thus a collection of variables, including both outcome variables and explanatory variables and possibly including multiple levels of data. One data structure may have a different number of levels than another.

If you are new to aML, you do not need to be concerned with data structures. Most examples in this manual do not involve data structures. Even the data for multiprocess examples are

typically not separated into data structures. However, if the variables of one process are very different from those of another process, it may be convenient to place one set of variables in one data structure and the other in another data structure. Or perhaps one set of programs prepared the data for one outcome and another set of programs produced data for another outcome; in that case, too, it may be convenient to keep the variables into separate entities (data structures) rather than merge them before creating ASCII data. You may also separate data purely because you want to run separate models on them. For example, if you intend to run separate models for males and females, you could consider putting data pertaining to males in one data structure and those pertaining to females in another, even when the variables are the same. The model statements, in the aML control file, will then specify to which data structure they apply.

Data structures are numbered with strictly positive integers. A structure number may be used in more than one ASCII data file, and any one ASCII data file may contain more than one data structure.

3.1.4. Concepts of Variables

In aML, variables may be categorized as either "control variables" or "data variables." Data variables typically contain substantive information, often derived in one form or another from survey responses or experimental observation. Control variables are variables in the data, but they do not contain substantive information. Their sole purpose is to provide structure to the data. You must list all data variables in the raw2aml control file, and you must not list any control variables. As a user, you are responsible for writing out three control variables to the ASCII data file(s): an observation ID variable, an optional data structure variable, and (for data with three or more levels) one or more variables telling raw2aml how many subbranches to read. (Internally, raw2aml and aML create many more control variables, mostly to improve computational efficiency and to check on data integrity.) This section discusses the two control variables and several data variables.

Control Variable: ID

Every observation in multilevel data may contain potentially many subbranches with repeated outcomes and explanatory variables. An observation consists of anything that is potentially correlated and is itself, by definition, statistically independent from other observations. It is very important that aML knows which outcomes and explanatory variables belong to the same observation. This is achieved by assigning all data belonging to a particular observation a unique ID. These IDs must be strictly positive integer numbers.

As you create ASCII data files to be converted by raw2aml, you must write the ID variable as the very first item on every logical record. However, you must not list the ID variable among the variable list(s) in the raw2aml control file. Raw2aml knows that it is always there. In fact, it creates a variable in the output data, "`_id`" which is equal to the ID variable. The output data set all variables that you listed in the raw2aml control file, plus one additional variable, "`_id`".

All data that belong to the same observation must appear contiguously in any one ASCII data file. For example, you may not write out part of the data for the first observation, then part of the second observation, and then the rest of the first observation, to the same ASCII file. If that appears problematic, consider writing out your data to separate ASCII files. All ASCII files must be sorted in the same ID order. They do not necessarily need to be sorted in increasing ID order, although it tends to minimize errors if you do sort your (SAS, Stata, SPSS) by ID before creating ASCII files.

Control Variable: Data Structure Number

As explained briefly in Section 3.1.3 above and more extensively in Section 3.2 below, you may organize your data into subsets called data structures. Each data structure may have its own unique set of levels and variables. There may be records belonging to multiple data structures in any one ASCII data file. You must therefore tell raw2aml to which data structure a particular ASCII record belongs.

Data structures are identified by number. This number needs to be the second variable in logical ASCII records, immediately following the ID. It needs to be on the same line (physical record) as the ID.

The raw2aml control file must list variables for every data structure. If raw2aml encounters a data structure that is not documented in the control file, it generates an error message. If the raw2aml control file specifies one or more data structures, raw2aml knows that every ASCII record contains a data structure in its second field. The data structure *number* thus appears in the raw2aml control file, but the data structure *variable* must not be listed as a variable.

Control Variable(s): Number(s) of Subbranches

Number(s) of subbranches are relevant only for data with three or more technical levels. Apart from data for hazard models with time-varying covariates, such data are not common. See Section 3.4.

aML supports unbalanced multilevel data, i.e., data in which the number of subbranches may vary across observations. You therefore need to tell raw2aml how many subbranches to expect when it reads a particular record in your ASCII file(s). This is achieved by writing out a control variable that is equal to the number of subbranches.

As will be explained in detail below, each ASCII data logical record corresponds to a level 2 unit or "branch."[21] If your data contain three levels, you need to specify in each level 2 branch how many level 3 subbranches to expect. This one extra number is a control variable, because it only serves to organize the data. If your data contain four levels, you must not only specify how

[21] Each level 2 branch may contain very many variables which may wrap over multiple lines in the ASCII file. Each line constitutes a physical record, so there may be multiple physical records for each logical record.

many level 3 subbranches to expect, but also how many level 4 subbranches there are within each level 3 branch. Suppose the number of level 3 subbranches is `n3`; there must then be `n3` additional control variables for the numbers of level 4 subbranches, for a total of `1+n3` control variables. If your data contain five levels, additional control variables are needed to specify the numbers of level 5 subbranches within each of the level 4 branches; et cetera. If your data contain only two levels, no control variables are needed, because each level 2 branch is written as a separate logical ASCII record.

In ASCII data sets, the number or numbers of subbranches must immediately follow the ID variable and the optional data structure variable.

Variables that specify how many subbranches to expect are in the data and may vary across observations and (sub)branches. However, they are control variables without substantive information and must not be listed in the raw2aml control file.

Data Variables: Level-Specific Variables

All data variables live at a certain level. Level 1 variables are constant for the entire observation. For example, in school test data, there may be schools (level 1), students (level 2), years in school (level 3), and tests (level 4). Any variable that is school-specific (private/public, geographic location) may be a level 1 variable. Student-specific variables (sex, date of birth) may be level 2 variables, et cetera.

In ASCII data files, control variables (ID, data structure, numbers of subbranches) come first, followed by level 1 variables, followed by level 2 variables, followed by level 3 variables, et cetera.

If you want, you may replicate higher-level variables for each subbranch and list them as lower-level variables. For example, you could replicate the sex of a student for every test and list variable `sex` as a level 4 variable. It would require more storage space for your data and serve little purpose, but it illustrates that the (technical) levels that you determine need not be the same as the conceptual levels in the data. aML does not care, generally speaking, at what level a variable appears, provided that its level is at least as high as the outcome variable. We note one exception: in weighted optimization ("`option weight`" or "`option normweight`"), the weight variable must be a level 1 variable.

The raw2aml control file must list which variables are at which level. In the ASCII data, each level-specific variable must be present for each replication at its level. There can be no missing value for any variable. In other words, missing values in your (SAS, Stata, SPSS) data set must first be resolved before creating ASCII data files; see Section 3.5.

Data Variables: Outcome variables

Outcome variables may be specified at any level. The level of the outcome variable(s) specified in a model statement determine the "level of the outcome." Any variable (explanatory variable in a regressor set, reference variable, draw variable, etc) used in a model statement for that outcome must be at the same level or higher in the data. For example, if the outcome of a probit model is at level 3, then all explanatory variables, reference variables, et cetera, must be at level 1, 2 and/or 3. (There is one exception: in hazard models with time-varying covariates, variables in regressor sets may be one level below the outcome variables.)

Raw2aml and aML do not know which variables are outcome variables. From their perspective, outcome variables are just like all other data variables.

Data Variables: Draw Variables

Some variables are used for the special purpose of determining independence of residuals. Residuals in model statements are specified as "`res(draw=`*`varname`*`, ...)`". Every time a residual appears in an equation, aML evaluates its draw and determines whether the residual is independent from residuals in other equations or whether it is correlated with residuals in other equations. Residuals with the same draw variable are correlated and independent from residuals with another draw variable. (See Section 4.1.2 for details.)

Draw variables must be at the same level or higher as the outcome variable. They help control the model specification, not the organization of the data, and are thus ordinary data variables. They must be listed in the raw2aml control file, just like all other data variables. Draw variables must be strictly positive and integer-valued.

Data Variables: Indirect Reference Variables

Model statements specify models in terms of previously defined building blocks. There are two ways to refer to those building blocks: directly and indirectly. Direct referencing means that the model specification simply contains the name of the building block. For example,

```
model = regset BetaX + res(draw=varname, ref=eps)
```

contains two directly referenced building blocks, regressor set `BetaX` and residual `eps`. Indirect referencing is used when building blocks need to enter an equation conditional on the value of some so-called reference variable. For example:

```
model = regset(refvar=educ) + res(draw=varname, refvar=sex);
```

Both `educ` and `sex` are reference variables. Their values determine which regressor set and residual enter the model equation; a zero reference variable implies that a building block does not enter the equation at all. Indirect referencing requires that building block definitions contain so-called reference numbers, so that they may be identified not just be name, but also by number. See Section 13.3.4 for a detailed discussion.

Reference variables must be at the same level or higher as the outcome variable. They help control the model specification, not the organization of the data, and are thus ordinary data variables. They must be listed in the raw2aml control file, just like all other data variables. Reference variables must be non-negative and integer-valued.

3.2. Multilevel and Multiprocess Data

Most examples above used data with just a single level. One of aML's strengths, though, is its ability to store multilevel data and to estimate multilevel models. This section illustrates how to create multilevel data with raw2aml. Throughout this section, we assume that you are familiar with the conversion of single-level data (Section 2.1), and with concepts of files and variables (Section 3.1).

There are two important aspects to the creation of multilevel data: the structure of ASCII data file(s) and the syntax of the raw2aml control file. We illustrate both aspects for two-level and three-level sample data and for data with two data structures. All files are included with aML's sample data and programs under `Samples\Section3`. The samples illustrate the most commonly used options and features. For a complete discussion see Chapters 9 and 10.

Sections 3.2.1, 3.2.2, and 3.2.3 illustrate how to create data with two, three, and more levels, respectively. Section 3.2.4 discusses multiprocess data.

3.2.1. Two-Level Data

Suppose we are interested in the decision to deliver babies in a hospital versus at home or elsewhere, and we have data on one or more births per female respondent. Also see Section 4.1.1. The data contain information on 501 mothers with a total of 1060 births.

First we need to decide on the level structure. Children are nested within mothers, so mothers form level 1 and children are at level 2. In other words, each child corresponds to a level 2 branch. If the (SAS, Stata, SPSS) data do not yet contain unique mother IDs, we need to create them now. Suppose the ID variable is `momid`. Other variables include `educ` (maternal education), `income` (parental family income), `distance` (distance to nearest hospital, in km), and `hospital` (indicator for whether the child was delivered in a hospital). Maternal education is measured after the last child is born and does not vary across births. It is therefore a level 1 variable. All other variables may vary from birth to birth.

Suppose the unit of observation in the data is a birth. The records for the first four mothers (11 births) contain:

momid	educ	income	distance	hospital
4	3	76	1.7	0
17	2	275	7.9	0
		200	1.8	1
		47	6.2	0
		1665	1.0	1
		95	4.8	0
18	3	196	1.8	1
20	2	80	3.7	0
		169	10.6	1
		51	3.2	1
		190	3.1	0

The four mothers shown here had one, five, one, and four children, respectively.

The basic rule for creating ASCII data files is that each (logical) record corresponds to a level 2 branch, i.e., to a birth. The rationale and full implications of that rule will become clear below. Writing out information pertaining to one level 2 branch (birth) at a time is particularly easy in the example, because the unit of observation is a birth. In SAS, the code may be:

```
data _null_;
   set dataname;
   file 'hospital.raw';
   put momid educ income distance hospital;
```

and in Stata:

```
outfile momid educ income distance hospital using hospital.raw
```

In both cases, the ASCII file that is created is "`hospital.raw`". The first 11 records are:

```
4 3 76 1.7 0
17 2 275 7.9 0
17 2 200 1.8 1
17 2 47 6.2 0
17 2 1665 1 1
17 2 95 4.8 0
18 3 196 1.8 1
20 2 80 3.7 0
20 2 169 10.6 1
20 2 51 3.2 1
20 2 190 3.1 0
```

This shows the SAS version. Stata, by default, neatly lines up the data in columns. As a result, it places up to ten blank space between values, thus creating ASCII files that are far larger than needed. We recommend that you use Stata's "comma" option to the outfile command:

```
outfile momid educ income distance hospital using hospital.raw, comma
```

All data fields are now separated by a comma and (in this example) take about two-thirds less disk space. Raw2aml accepts either delimiter.

Data fields in ASCII data files may be delimited by blank spaces, commas, tab characters, and/or line feeds (carriage returns). The latter simply means that data fields may wrap over multiple lines. There is one exception: the observation ID and the data structure number, if any, must be on the same line; see Section 3.2.4.

The ASCII data are now ready to be converted into aML-format by raw2aml. Raw2aml reads its required information from a control file (hospital.r2a):

```
1  ascii data file = hospital.raw;
2
3  level 1 var = educ;
4  level 2 var = income distance hospital;
```

The raw2aml control file specifies the name(s) of the ASCII input data and lists variable names, by level. Note carefully that the ID variable, momid, does not appear in the control file.

The raw2aml control file lists the names of data variables only. Control variables, such as the observation ID, are not listed.

Run raw2aml with "hospital.r2a" as control file:

```
raw2aml hospital
```

This produces an aML-formatted data file and a human-readable data documentation file. By default, the data file has the same name as the control file, but with extension ".dat" rather than ".r2a". The documentation file also has the same stem but with extension ".sum". Raw2aml thus produced data file "hospital.dat", in binary form, and documentation file "hospital.sum", a text file. Its contents are also written to standard output:

```
1  Documentation for 'hospital.dat'
2  Created on Sun Feb 13 11:03:20 2000 with raw2aml version 1.00.
3  Ascii data set: 'hospital.raw'
4
5  Number of observations:     501
```

```
 6  Maximum number of level 2 branches in any observation:     10
 7
 8  ------------------------------------------------------------
 9
10  LEVEL 1 VARIABLES:
11  Variable     N       Mean    Std Dev        Min        Max
12  _id        501   1221.261   723.3883        4.0     2472.0
13  educ       501   1.552894   .6447454        1.0        3.0
14
15  LEVEL 2 VARIABLES:
16  Variable     N       Mean    Std Dev        Min        Max
17  income    1060   1097.325   2386.767        4.0    29481.0
18  distance  1060   3.918396   2.875465        0.1       17.0
19  hospital  1060   .2971698   .4572277        0.0        1.0
20
21  ------------------------------------------------------------
22
23  NOTE: there is variation in all data variables.
```

Please verify that the number of observations is correct. If it is too high or low, the IDs were not properly linked. This will be the case if there are more or fewer variables in the ASCII file than specified in the raw2aml control file. Also verify that the maximum number of level 2 branches corresponds to the maximum in the (SAS, Stata, SPSS) data. In this case, there are up to ten births per woman.

The only level 1 variable that we specified in the raw2aml control file is `educ`. Raw2aml adds another variable, `_id`. Its value corresponds to the observation ID, as was stored in (SAS, Stata, SPSS) variable `momid`. You may use it like any other level 1 variable in model statements, if you desire.

The summary statistics indicate the number of values (`N`) upon which the mean, standard deviation, minimum, and maximum are based. Please verify that these numbers are correct. There are 501 level 1 values. There were 501 women in the sample, so that is correct. Note that we wrote out educ for every ASCII record (birth), i.e., we duplicated it. Raw2aml knows, however, that it is a level 1 variable, and only counts them once for every observation. (It also checks that its value is the same for every birth and generates an error message if there is variation within observation.)

The data are now ready for analysis by aML; Section 4.1.1 estimates a probit model with unobserved heterogeneity at the mother level using these data.

What If the Unit of Observation is Level 1?

The two-level hospital delivery example was straightforward, in part because the unit of observation in the (SAS, Stata, SPSS) data set was level 2, i.e., the level at which every ASCII record needs to be written out. What if the unit of observation were a woman? Recall that there were up to ten births per mother (line 6 in "`hospital.sum`"). Suppose the actual number of children is given by variable `numkids`. The data would contain 501 observations with the

following variables: momid, as before; educ, as before; numkids, with the number of births of the index mother; income1-income10, with income values at the time of the birth of each of the up-to-ten children; dist1-dist10, with distances to the nearest hospital; and hosp1-hosp10, with hospital delivery indicators. Many of the array variables (income1-income10, dist1-dist10, hosp1-hosp10) will be missing, as only a few women had ten births.

To be clear: it will be highly unusual for multilevel data to be organized such that each (SAS, Stata, SPSS) observation corresponds to the most aggregate level. We discuss it for illustration only.

The creation of the ASCII data file will be more complicated now that the unit of observation is not level 2. In SAS, the code may still be relatively straightforward:

```
data _null_;
   set dataname;
   array income(*) income1-income10;
   array dist(*)   dist1-dist10;
   array hosp(*)   hosp1-hosp10;
   file 'hospital.raw';
   do i=1 to numkids;
      put momid educ income(i) dist(i) hosp(i);
   end;
```

In Stata, by contrast, all commands are applied to all observations without allowing for looping over observation-specific limits, such as numkids. We therefore need to transform the data such that each observation corresponds to a child:

```
reshape long income dist hosp, i(momid) j(child)
```

This creates a "long" data set with 1060 observations, each corresponding to a birth. It is identical to the data that we described earlier in this section (except that the distance variable is dist rather than distance, the hospital variable is hosp rather than hospital, and a child index number variable was created, child.) The data may now be written out as before:

```
outfile momid educ income dist hosp using hospital.raw, comma
```

3.2.2. Three-Level Data

Section 2.4 described estimation of a hazard model of disruption of a first marriage. Suppose we are interested in the hazard of any marital disruption, not just disruption of the first marriage. We have data on the dissolution of one or more marriages per respondent. One of the explanatory variables is the number of children that the couple has (possibly from prior relationships). This covariate may change during a marriage and is thus time-varying.

We first need to decide on the level structure. Each respondent may have multiple marriages, and time-varying covariates are nested within marriage spells. Consistent with these conceptual

levels, we define the level 1 unit to be a respondent, the level 2 unit to be a marriage, and the level 3 unit to be a time interval within a marriage.

If the (SAS, Stata, SPSS) data do not yet contain a respondent ID, we need to create one. Suppose the ID variable is `personid`. Other variables include sampling weight (`weight`); a variable indicating whether the marriage was still intact as of the last known date (`censor`); two variables indicating the lower and upper bound of the duration of the hazard spell (`lower` and `upper`); the marriage number of the index marriage (`marnum`); respondent's age as of the beginning of the marriage (`age`); educational attainment of the husband and wife (`hiseduc` and `hereduc`); indicators for whether husband and wife are African-American (`heblack` and `sheblack`); the age difference between husband and wife (`agediff`); the number of time intervals in a marriage spell (`numint`); the number of children that live with the couple (`numkid`), and the duration between the birth of a child and the beginning of the marriage (`time`). These variables were explained more fully in Section 2.4 which illustrated a simple hazard model. The difference with those data is that we now add information on second and higher order marriages. In Section 2.4, levels 1 and 2 corresponded to a marriage and a time interval, respectively. We now insert the respondent at level 1 and move marriages and time intervals to levels 2 and 3, respectively. The important additional variables are the marriage number (`marnum`) and the number of time intervals within a spell (`numint`). The latter is not really new, but it was not needed in Section 2.4. It is now.

The novelty of this section is the introduction of a third level. As always, each logical ASCII record must correspond to a level 2 branch, i.e., to a marriage spell. The sequence in which variables must be written out is: control variables, level 1 data variables, level 2 data variables, level 3 data variables. With three levels, we need to tell raw2aml how many level 3 subbranches to read in the current level 2 branch. The information is contained in the variable for number of time intervals within a spell (`numint`). This variable is needed to provide structure to the data, and is thus a control variable, just like the observation ID.

Every ASCII record contains all information pertaining to a level 2 branch. ASCII data with three levels (and without data structure) need to be ordered as follows: control variables (ID and number of level 3 subbranches in this level 2 branch); level 1 data variables; level 2 data variables; and level 3 data variables. Level 3 variables are sorted by level 3 subbranch, i.e., first all variables for the first level 3 subbranch, then all variables for the second level 3 subbranch, et cetera.

For now, assume that the unit of observation in the (SAS, Stata, SPSS) data is a marriage. There are 3,371 respondents in the data with a total of 4,238 marriages, so that the number of (SAS, Stata, SPSS) observations is 4,238. Suppose the maximum number of time intervals (level 3 subbranches) in any one marriage is 17, i.e., $1 \leq \text{numint} \leq 17$. (The minimum is one, not zero, because even if the couple did not have any children during the marriage, the number of children would be constant, and there would still be one value for `numkid`.) The level 3 variables are thus

in fact arrays, as they take on 17 different values over the life of a spell: `numkid1-numkid17` and `time1-time17`.

The following table shows data values for five respondents whose experiences are illustrative:

personid	weight	marnum	censor	lower	upper	hiseduc	hereduc	heblack	sheblack	age	agediff	numint	time1	numkid1	time2	numkid2	time3	numkid3	time4	numkid4
9	23	1	1	10.5	10.5	12	12	0	0	21.0	1.0	2	3.7	0	10.5	1	.	.	.	.
11	23	1	1	34.9	34.9	3	3	0	0	24.5	.7	3	.8	0	32.5	1	34.9	2	.	.
15	23	1	0	15.0	20.1	7	7	0	0	19.5	2.4	1	20.1	0	.	.	.	.	.	.
		2	1	13.9	13.9	7	3	0	0	68.3	15.1	1	13.9	0	.	.	.	.	.	.
43	23	1	1	16.7	16.7	16	16	0	0	22.5	.2	4	3.1	0	4.6	1	6.7	2	16.7	3
77	26	1	1	4.1	4.1	16	16	0	0	24.8	2.4	2	3.8	0	4.1	1	.	.	.	.
		2	0	6.5	6.6	16	16	0	0	31.8	2.4	2	3.4	0	6.6	1	.	.	.	.
		3	0	.7	.8	16	16	0	0	38.5	2.4	1	.8	0	.	.	.	.	.	.

The example pertains to hazard data with time-varying covariates. If you are not specifically interested in hazard models, you may skip the narrative interpretation of the data and go to the SAS code about one-half of a page below. The important aspect is that there are three levels. Each observation has potentially different numbers of level 2 branches, and each level 2 branch has potentially different numbers of level 3 subbranches.

Each row in the data table above corresponds to a marriage, i.e., to an observation in the (SAS, Stata, SPSS) data set. Variable `censor` indicates whether a marriage was disrupted (`censor=0`) or whether it survived to the last survey date or until widowhood (`censor=1`). The respondent with `personid=15` was married twice; his first marriage ended in divorce (`censor=0`), and his second marriage survived. The respondent with `personid=43` was married once and had three children at 3.1, 4.6, and 6.7 years after the wedding. The respondent with `personid=77` was married twice. The first marriage survived but ended in widowhood (`censor=1`); the second and third marriages both ended in divorce (`censor=0`).

Variable `numint` was generated to indicate how many level 3 subbranches there are, i.e., how many nonmissing (relevant) sets of level 3 variables (`time` and `numkid`) there are. None of the respondents shown in the table had more than four level 3 subbranches, so we only showed `time1-time4` and `numkid1-numkid4`, but there are individuals with up to seventeen level 3 subbranches and nonmissing values of `time17` and `numkid17`.

Note that the highest nonmissing time value is always equal to variable upper which denotes the upper bound of the event duration. The highest time mark must always be at least as high as the upper bound, because otherwise aML would not know what the values of time-varying covariates are beyond the highest time mark.

In SAS, the data may be written out as follows:

```
data _null_;
   set dataname;
   array time(*)   time1-time17;
   array numkid(*) numkid1-numkid17;
   file 'divorce3.raw';
   put personid numint                  /* control variables */
       weight                                    /* level 1 */
       marnum censor lower upper hiseduc         /* level 2 */
       hereduc heblack sheblack age agediff;     /* level 2 */
   do i=1 to numint;
      put time(i) numkid(i);                     /* level 3 */
   end;
```

Note the control variables `personid` and `numint`. The latter tells raw2aml how many replications of level 3 variables (`time` and `numkid`) to read in. Also carefully note the order in which level 3 variables are written out: first all variables for the first level 3 subbranch, then all variables for the second level 3 subbranch, et cetera. Do not write out all `numint` values of `time`, followed by all `numint` values of `numkid`!

The ASCII records corresponding to the data table above contain the following (`divorce3.raw`). Note that non-integer numbers in the data table were rounded; the records below are from "`Samples\Chapter3\divorce3.raw`" (with added line numbers):

```
 1   9 2 23 1 1 10.546 10.546 12 12 0 0 20.953 1.013 3.734 0 10.546 1
 2   11 3 23 1 1 34.943 34.943 3 3 0 0 24.498 .687 .767 0 32.512 1 34.943 2
 4   15 1 23 1 0 15.012 20.052 7 7 0 0 19.499 2.352 20.052 0
 5   15 1 23 2 1 13.944 13.944 7 3 0 0 68.29 15.064 13.944 0
 8   43 4 23 1 1 16.706 16.706 16 16 0 0 22.5 .162 3.083 0 4.578 1 6.664 2 16.706 3
16   77 2 26 1 1 4.085 4.085 16 16 0 0 24.835 2.352 3.833 0 4.085 1
17   77 2 26 2 0 6.557 6.634 16 16 0 0 31.819 2.352 3.428 0 6.634 1
18   77 1 26 3 0 .709 .786 16 16 0 0 38.5 2.352 .786 0
```

In Stata, the procedure is not quite as straightforward, because Stata does not allow looping over observation-specific values, such as `numint`. In other words, it cannot write out two sets of level 3 variables for one observation and some other number of sets for the next. The solution is write out all 17 sets of variables for all observations. Of course, there will then be many missing values. Section 3.3 explains how to solve this issue.

The data may be converted with raw2aml using the following control file (`divorce3.r2a`):

```
1  ascii data file = divorce3.raw;
2
3  level 1 var = weight;
4  level 2 var = marnum censor lower upper hiseduc
5                hereduc heblack sheblack age agediff;
6  level 3 var = time numkids;
```

This creates data file "`divorce3.dat`" and corresponding data documentation file "`divorce3.sum`":

```
 1  Documentation for 'divorce3.dat'
 2  Created on Sun Feb  6 10:19:34 2000 with raw2aml version 1.00.
 3  Ascii data set: 'divorce3.raw'
 4
 5  Number of observations:    3371
 6  Maximum number of level 2 branches in any observation:      6
 7  Maximum number of level 3 branches in any observation:     17
 8  Maximum number of level 3 branches in any level 2 branch:  17
 9
10  ------------------------------------------------------------
11
12  LEVEL 1 VARIABLES:
13  Variable      N       Mean    Std Dev        Min        Max
14  _id        3371   7761.647   4959.523        9.0    17302.0
15  weight     3371   15.88876   10.23573        1.0       32.0
16
17  LEVEL 2 VARIABLES:
18  Variable      N       Mean    Std Dev        Min        Max
19  marnum     4238   1.241387   0.522801        1.0        6.0
20  censor     4238   .7067013   .4553279        0.0        1.0
21  lower      4238   16.33635   14.46773       0.06     73.068
22  upper      4238   17.05656   14.51844      0.063     73.068
23  hiseduc    4238   11.54719   3.041219        1.0       21.0
24  hereduc    4238   11.52053    2.88509        1.0       21.0
25  heblack    4238   .2177914   .4127936        0.0        1.0
26  sheblack   4238   .2328929   .4227244        0.0        1.0
27  age        4238   25.94252   9.421657      5.342     86.335
28  agediff    4238   2.447035   5.392547    -39.663     38.081
29
30  LEVEL 3 VARIABLES:
31  Variable      N       Mean    Std Dev        Min        Max
32  time      10082   10.54854   11.78084      0.003     73.068
33  numkids   10082   1.372148   1.832152        0.0       16.0
34
35  ------------------------------------------------------------
36
37  NOTE: there is variation in all data variables.
```

As always, please carefully check that the number of observations is correct. Recall that there were 3,371 respondents in the data with a total of 4,238 marriages. The number of observations that the data documentation file reports (line 5) is the "true" number of observations, not the number of observations in the (SAS, Stata, SPSS) data set, where the unit of observation was a marriage. Line 6 reports that there were up to 6 level 2 branches in any observation, i.e., up to six marriages per respondent. Line 7 reports that there were up to 17 level 3 branches per observation and line 8 that there were up to 17 level 3 branches per level 2 branch. Always verify that these numbers correspond to the maximum dimensions in your data.

In the summary statistics of level 2 variables, note that they are based on 4,238 replications, which corresponds to the number of marriages.

The data are now ready for model estimation by aML.

So far, we assumed that the unit of observation in the (SAS, Stata, SPSS) data set is a marriage. What if the unit were a respondent, i.e., a level 1 unit? Since there are up to six marriages per person, each level 2 variable is stored in six variables. There should be a variable indicating the number of marriages for the index person, `nummar`. Furthermore, since there are up to 17 intervals (level 3 subbranches) in any one marriage (level 2 branch), each level 3 variable is stored in 6*17=102 variables.

Again, it would be highly unusual for the (SAS, Stata, SPSS) data to be organized such that each observation corresponds to the most aggregate level. We discuss the case for illustrative purposes only.

As always, each logical ASCII record needs to contain all information pertaining to a level 2 branch. The ASCII data should thus be identical to the data described above. The issue is thus purely a (SAS, Stata, SPSS) coding issue. Assuming that the variables are neatly organized in arrays, the data may be written out using SAS as follows:

```
data _null_;
   set dataname;
   array censor(6)   censor1-censor6;
   array lower(6)    lower1-lower6;
   array upper(6)    upper1-upper6;
   array hiseduc(6)  hiseduc1-hiseduc6;
   array hereduc(6)  hereduc1-hereduc6;
   array heblack(6)  heblack1-heblack6;
   array sheblck(6)  sheblck1-sheblck6;
   array age(6)      age1-age6;
   array agediff(6)  agediff1-agediff6;
   array numint(6)   numint1-numint6;
   array time(6,17)  time1-time102;
   array numkd(6,17) numkd1-numkd102;
   file 'divorce3.raw';
   do i=1 to nummar;
      put personid numint(i)          /* control variables */
          weight                                /* level 1 */
          i censor(i) lower(i) upper(i) hiseduc(i) /* level 2 */
          hereduc(i) heblack(i) sheblck(i) age(i) agediff(i);
      do j=1 to numint(i);
         put time(i,j) numkd(i,j);              /* level 3 */
      end;
   end;
```

The resulting ASCII data file is identical to the file created earlier in this section. Note that the ASCII data creation process is substantially simpler when the unit of observation in the (SAS, Stata, SPSS) data corresponds to a level 2 unit. Also note that the data are stored more efficiently when the unit of observation is a level 2 unit, because there is no need to allocate space for six sets of level 2 variables and 6*17=102 sets of level 3 variables for each observation.

As before, Stata users will need to write out all six sets of level 2 variables and all 6*17=102 sets of level 3 variables, even though many will be irrelevant and missing. See Section 3.3.

Finally, what if the unit of observation in the (SAS, Stata, SPSS) data corresponds to a level 3 unit? The data would be even more compact, because there is no need to allocate space for 17 sets of level 3 variables. As before, the ASCII data need to contain the very same information, so the data creation issue is a (SAS, Stata, SPSS) coding issue. Assuming that the data are sorted by respondent ID (`personid`) and marriage number (`marnum`) and time interval, the ASCII data may be created as follows in SAS:

```
data _null_;
   set dataname;
   by personid marnum;
   file 'divorce3.raw';
   if (first.marnum) then put
      personid numint                /* control variables */
      weight                                 /* level 1 */
      marnum censor lower upper hiseduc      /* level 2 */
      hereduc heblack sheblack age agediff;  /* level 2 */
   put time numkid;                          /* level 3 */
```

If the unit of observation is lower than level 2, Stata users must first reshape their data such that the unit of observation corresponds to a level 2 branch. This is most conveniently done with the "reshape" command. It requires that the data contain a counter indicating the level 3 branch number within each level 2 branch (marriage):

```
quietly by personid marnum: generate interval=_n
reshape wide time numkid, i(personid marnum) j(interval)
```

The data now have a marriage as the unit of observation and variables `time` and `numkid` are converted into `time1-time17` and `numkid1-numkid17`. As before, all 17 sets of level 3 variables need to be written out, including many irrelevant (missing) values; see Section 3.3.

3.2.3. Four and Higher Level Data

It is rarely (if ever) necessary to create data with four or more (technical) levels. Even if your model features four or more conceptual levels, it is almost always more convenient to create aML data with two levels (three if there are hazard models with time-varying covariates). See Section 3.4.

The rule that every logical ASCII record contains all information pertaining to a level 2 branch remains applicable to data with four or more levels.

Every ASCII record contains all information pertaining to a level 2 branch. ASCII data with four or more levels (and without data structure) need to be ordered as follows: control variables (ID and numbers of level 3+ subbranches in this level 2 branch); level 1 data variables; level 2 data variables; level 3 data variables, level 4 data variables, level 5 data variables, et cetera. Level 3 variables are sorted by level 3 subbranch, level 4 variables are sorted by level 4 subbranch, et cetera.

With four or more levels, additional control variables are required to tell raw2aml how the data are organized. With one or two levels, the only control variable is the observation ID. With three levels, the ID needs to be followed by the number of level 3 subbranches in the current level 2 branch. With four levels, you need to add the numbers of level 4 subbranches within each level 3 subbranch. For example, if a level 2 branch contains four level 3 subbranches, and those four level 3 subbranches have 2, 3, 3, and 0 level 4 subbranches, then the ID variable must be followed by "`4 2 3 3 0`". With five levels, you need to add the numbers of level 5 subbranches within each level 4 subbranch. Continuing the example, the four level 3 subbranches had a total of eight level 4 subbranches, and eight additional numbers of level 5 subbranches thus need to be specified. (Note that the fourth level 3 subbranch did not have any level 4 subbranches, and you must not specify anything about further nonexisting level 5 subbranches.) The control variables thus are:

```
id   n3   n4...n4   n5...n5   n6...n6   <et cetera>
```

where n3 is the number of level 3 subbranches in the current ASCII record (level 2 branch), `n4...n4` are the numbers of level 4 subbranches, et cetera. As the number of levels increases, the numbers of subbranches that need to be specified tends to increase rapidly.

Consider an example of school test data. The data contain a sample of 286 schools (level 1). From each school, one or more students are sampled (level 2). The students attended the school one or more years (level 3). In each year, they took one or more tests (level 4). The level of observation in the (SAS, Stata, SPSS) data set is a student. Schools are identified by variable `schoolid`. We also know whether they are innercity (`innercty`) or private (`private`) schools. Level 2 variables include the student's ID (`student`), sex (`male`), father's education (`dadeduc`), and mother's education (`momeduc`). Level 3 variables include grade level (`grade`), whether the student was held back last year (`retain`), and class size (`clssize`). The only level 4 variable is the test score (`score`).

There are up to 18 students per school, up to six years in school per student (including grade duplication), and up to seven tests per school year. Since the unit of observation in the (SAS, Stata, SPSS) data is a student, level 3 and level 4 variables are stored as arrays. For example, there are up to six years in school per student, so there are six class size variables (`clssize1-clssize6`). There are up to seven tests per school year, so there are 6*7=42 test score variables (`score1-score42`). The number of years in school is given by variable `numyears`; the numbers of tests in each year are in `numtst1-numtst6`. (Variable `numyears` corresponds to `n3` above,

and numtst1-numtst6 correspond to n4...n4, above.) In SAS, the ASCII data file may be written out as follows.

```
data _null_;
   set dataname;
   array numtst(6)  numtst1-numtst6;
   array grade(6)   grade1-grade6;
   array retain(6)  retain1-retain6;
   array clssize(6) clssize1-clssize6;
   array score(6,7) score1-score42;
   file 'school.raw';

   put schoolid numyears;           /* control vars ID and n3 */
   do i=1 to numyears;
      put numtst(i);                  /* control vars n4...n4 */
   end;
   put innercty private                          /* level 1 */
       student male dadeduc momeduc;             /* level 2 */
   do i=1 to numyears;
      put i grade(i) retain(i) clssize(i);       /* level 3 */
   end;
   do i=1 to numyears;
      do j=1 to numtst(i);
         put score(i,j);                         /* level 4 */
      end;
   end;
```

Note carefully that all level 3 variables are written out before all level 4 variables. The resulting file is "Samples\Chapter3\school.raw". The raw2aml control file (school.r2a) is:

```
1  ascii data file = school.raw;
2
3  level 1 var = innercty private;
4  level 2 var = student male dadeduc momeduc;
5  level 3 var = year grade retain clssize;
6  level 4 var = score;
```

As always, the only variables that are listed are data variables. Raw2aml expects to find control variables, and they must not be listed. The student ID variable (student) is not an observation ID and is thus not a control variable. Its use will become clear in Section 4.1.2. Similarly, we wrote out the counter variable (i) to identify year-in-school. It, too, will be needed in Section 4.1.2. In raw2aml, we gave the variable a more meaningful name (year).

Raw2aml produced aML-formatted file "school.dat" and its documentation file "school.sum":

```
 1  Documentation for 'school.dat'
 2  Created on Sun Feb 20 14:14:30 2000 with raw2aml version 1.00.
 3  Ascii data set: 'school.raw'
 4
 5  Number of observations:     286
 6  Maximum number of level 2 branches in any observation:     18
 7  Maximum number of level 3 branches in any observation:     70
 8  Maximum number of level 4 branches in any observation:    343
 9  Maximum number of level 3 branches in any level 2 branch:   6
10  Maximum number of level 4 branches in any level 2 branch:  33
11  Maximum number of level 4 branches in any level 3 branch:   7
12
13  ------------------------------------------------------------
14
15  LEVEL 1 VARIABLES:
16  Variable      N       Mean    Std Dev        Min        Max
17  _id         286   479.2517   283.9388        1.0      969.0
18  innercty    286   .2027972   .4027875        0.0        1.0
19  private     286   .3146853   .4652044        0.0        1.0
20
21  LEVEL 2 VARIABLES:
22  Variable      N       Mean    Std Dev        Min        Max
23  student    2699   5.735087    3.45229        1.0       18.0
24  male       2699   .4972212   .5000849        0.0        1.0
25  dadeduc    2699   1.909226   0.704944        1.0        3.0
26  momeduc    2699   1.904409   .7092886        1.0        3.0
27
28  LEVEL 3 VARIABLES:
29  Variable      N       Mean    Std Dev        Min        Max
30  year       9210   2.383713   1.147876        1.0        6.0
31  grade      9210     10.538   1.117643        9.0       12.0
32  retain     9210   .0338762   .1809204        0.0        1.0
33  clssize    9210   21.98067    2.01516       15.0       29.0
34
35  LEVEL 4 VARIABLES:
36  Variable      N       Mean    Std Dev        Min        Max
37  score     46130   6.583481   .9453058        3.0       10.0
38
39  ------------------------------------------------------------
40
41  NOTE: there is variation in all data variables.
```

As always, verify that the documentation corresponds to your knowledge of the data. For example, the maximum numbers of (sub)branches in any observation or lower-level branch indicate that there are up to 18 students per school, up to six years in school per student, and up to seven tests per school year. The documentation also shows that there are up to 70 school years and up to 343 tests per school, and up to 33 tests per student. The data will be used to estimate a four-level model in Section 4.1.2.

Raw2aml and aML support any level of nested data. For more details see Chapter 10.

3.2.4. Multiprocess Data with Multiple Data Structures

aML supports multiprocess estimation, that is, estimation of models with different types of outcomes. For example, you may jointly estimate probit and hazard outcomes, or continuous and binomial outcomes, or any combination of any of the types of outcomes that aML supports. And, of course, you may also estimate models with two or more conceptually different types of outcomes that follow the same type of model, such as two hazard processes, or two logit models, et cetera.

You would typically estimate multiple processes using data that are created as explained in the preceding subsections, without additional complications. Suppose, for example, that you are interested in the effect of hospital delivery on infant survival, and you are concerned about the potential endogeneity of the choice to deliver babies in hospitals. (You are worried that you will underestimate the effect of hospital care, because women with high-risk pregnancies are more likely to go to a hospital.) You would create data with infant survival outcomes (perhaps a hazard model, or a probit or logit for survival until the first birthday) with explanatory variables that include an indicator of whether the baby was delivered in a hospital. At the same time, that hospital indicator variable is the outcome of a decision process that you wish to model jointly, so the data should also include explanatory variables for that outcome. The data contain two outcomes and any number of explanatory variables and may be prepared using the approach explained in the preceding subsections. You may estimate your multiprocess model on these data. There is nothing new here from the perspective of data creation.

However, as explained in Section 3.1.3, it is sometimes necessary and other times convenient to separate your data into so-called data structures. This type of data organization does have implications for data preparation. This subsection explains how to create data with multiple data structures. To be clear: data are sometimes split up into data structures because they contain different variables or levels, and thus reflect multiple processes. However, multiprocess models may perfectly well be estimated off data without data structures.

Consider an example where you are more or less forced to define multiple data structures. Suppose you are interested in modeling the determinants of wages. Your data contain individuals with one or more jobs over a period of time. On each job, annual wages are reported. One of the covariates of interest is job tenure, i.e., the length of time on the job. Job tenure may be endogenous, because both tenure and wages may be related to the unobserved quality of the match between job and employee. You therefore decide to jointly estimate job tenure (a hazard model) and wage determination (a continuous model). How should the data be structured? The unit of observation (level 1) is the individual, and jobs are level 2 units. Annual wage outcomes are nested within jobs, so wages are level 3 units. But the job tenure hazard model may contain time-varying covariates, such as marital status. Those covariates are constant over subintervals of job spells and change discretely from one subinterval to the next. The subintervals over which they are constant are not likely to coincide with calendar years to which annual wages refer. In other words, you would like to create two different types of levels 3, one for annual wages and one for subintervals over which time-varying covariates are constant. It is possible to do so, but it would

require some trickery that we do not advocate. A better solution is to place variables related to job tenure as an outcome in one data structure and variables related to wages as an outcome in another.

From the example it may be clear that applications that require separation of data into data structures tend to be very complex. There is no reason to introduce any complexities to illustrate the creation of data with multiple structures, which is actually very straightforward. So let's continue with an example that could have been done without data structures.

Suppose you wish to estimate a model of marriage formation. Your data contain information on both first and subsequent marriages. You anticipate that the model specification for first marriages will be different from the specification of subsequent marriages, if only because several determinants of second and higher order marriage formations are absent for the first marriage. For example, duration since the last marriage ended does not apply. You therefore anticipate specifying separate models for first and subsequent marriages. It may then be convenient to define one data structure for first marriages, and another for subsequent marriages. Let's assign data structures 10 and 20 to first and subsequent spells, respectively. In the aML estimation stage, you will then specify one model for first marriages, applicable to data structure 10, and another model for subsequent marriages, applicable to data structure 20.

The variable list of data structures 10 and 20 are different: the list of data structure 20 includes all variables of data structure 10, plus duration since the last marriage ended, an indicator for marital status (divorced or widowed; for first marriages it would always be never married and thus without variation), and perhaps more. You therefore need to tell raw2aml whether the current logical ASCII record belongs to data structure 10 or 20. This is done by inserting the data structure number immediately after the observation ID.

Every ASCII record contains all information pertaining to a level 2 branch. ASCII data with data structures need to be ordered as follows: control variables, level 1 data variables, level 2 data variables, level 3 data variables, level 4 data variables, et cetera. The control variables are the observation ID, data structure, and numbers of level 3 and lower subbranches in this level 2 branch. The ID and the data structure number must be on the same line (physical ASCII record); all other variables may wrap over multiple lines.

The raw2aml control file needs to specify variable lists at all levels for all data structures. Level 1 variables pertain to the observation, the most aggregated unit, and must thus be equal for all data structures. Data structures may therefore differ only at level 2 or lower. This is the rationale for the rule that each logical ASCII records must pertain one level 2 branch, and must include all information for the index level 2 branch and all its subbranches. The raw2aml control file may be as follows:

```
ascii data file = filename.raw;

level 1 var = <varlist>;
```

```
data structure = 10;
   level 2 var = <varlist>;
   level 3 var = <varlist>;

data structure = 20;
   level 2 var = <varlist>;
   level 3 var = <varlist>;
```

Now suppose that the data for data structures 10 and 20 were created by different series of (SAS, Stata, SPSS) programs. It is then convenient to create one raw ASCII file for the data of data structure 10 and another for data structure 20. Suppose the files are "`data10.raw`" and "`data20.raw`". Raw2aml knows to merge records from the two files on the basis of the observation ID with which every record begins. But what if not every ID is present on both files? For example, "`data10.raw`" may have records for ID=1, 3, 4, etc., whereas "`data20.raw`" has records for ID=1, 2, 5, etc. In such cases, an ID file provides raw2aml with all possible IDs. The ID file, say, "`id.raw`", should have one record for every ID. Each record should contain an ID only, no other variables. The raw2aml syntax would be:

```
ascii data files = data10.raw  data20.raw;
id file = id.raw;
```

et cetera; the remainder is as above. Raw2aml will first read an ID from the ID file; then turn to "`data10.raw`" and read records until the ID no longer matches the ID from the ID file; then turn to "`data20.raw`" and read records until the ID no longer matches the ID from the ID file; then merge records pertaining to the same ID and write them out into a single aML record; then read a second ID from the ID file, et cetera.

The example illustrated data with three levels, but the syntax is similar for any data level. Different data structures may have different number of levels and different variable lists. The variable lists only contain data variables. None of the control variables (ID, data structure, numbers of level 3 and lower subbranches) is listed. Raw2aml knows that they are in the ASCII data.

The raw2aml control file specifies separate variable lists for each of the data structures. The data structure number itself must be specified (`data structure=`*n*), but the control variable in the ASCII data that contains the data structure number must not appear in any of the variable lists. The data structure number is a control variable and is therefore not listed in the raw2aml control file. We have encountered three types of control variables: observation ID, data structure number, and numbers of level 3 and lower subbranches. They serve to communicate the organization of the data to raw2aml, and are the only control variables for which the user is responsible.

There is no hard limit to the number of data structures that you may distinguish. Data structure numbers must be strictly positive and integer-valued.

3.3. Rectangular Data

The term "rectangular data" refers to data that, in the ASCII data file, contain values for a fixed and constant number of subbranches, even though the numbers of subbranches varies from observation to observation or from (sub)branch to (sub)branch. Rectangular data thus contain potentially many irrelevant values. The term contrasts with "compressed data," which only contain data values for existing subbranches, i.e., only relevant data. Raw2aml is capable of converting either format into exactly the same aML-formatted file. The aML-formatted file never contains any irrelevant data values.

The distinction between rectangular and compressed data only applies to data with three or more levels. ASCII records always pertain to level 2 units, so there may only be irrelevant data at level 3 and lower.

In SAS, there is no reason to ever create rectangular data. It permits looping over the actual number of subbranches and only writing out relevant data. In Stata, however, this is not always possible. We encountered this in Section 3.2.2, where there were up to 17 subintervals (level 3 branches) in any one marriage spell (level 2 branch). If a particular marriage spell only contains, say, four subintervals, then the remaining 13 sets of level 3 variables will be missing. See, for example, the data table on page 104. Stata does not permit writing out just four sets of level 3 variables for one marriage and some other number of sets for another. It requires that you write out the same variables for every Stata observation.

Stata represents missing values as a period character ("`.`"). Unfortunately, raw2aml does not accept missing values in the form of period characters. You must therefore set irrelevant values equal to some other number (we recommend –99) and then write out all 17 sets of level 3 variables. If a marriage spell contains four level 3 subbranches, then its ASCII record will contain four sets of relevant level 3 variables and 13 sets of irrelevant level 3 variables that are all equal to –99. Everything else in the ASCII data file(s) is the same as with compressed data. In particular, you must still write out the control variable(s) that specify the numbers of level 3 and lower subbranches to read in. Those numbers now refer to relevant subbranches only. Raw2aml will read in 17 sets, but keep only the number that is relevant.

Continuing the divorce data example of Section 3.2.2, the Stata commands to create ASCII data file "`divorce4.raw`" are:

```
#delimit ;
/* Replace missing values of level 3 variables by -99 */
mvencode time1-time17 numkid1-numkid17, mv(-99) override;
/* Create the ASCII data file */
outfile personid numint                /* control variables */
       weight                                    /* level 1 */
       marnum censor lower upper hiseduc         /* level 2 */
       hereduc heblack sheblack age agediff
```

```
time1  numkid1                                  /* level 3 */
time2  numkid2
time3  numkid3
time4  numkid4
time5  numkid5
time6  numkid6
time7  numkid7
time8  numkid8
time9  numkid9
time10 numkid10
time11 numkid11
time12 numkid12
time13 numkid13
time14 numkid14
time15 numkid15
time16 numkid16
time17 numkid17   using divorce4.raw, replace comma wide;
```

The ASCII records corresponding to the data table on page 104 contain the following. Note that non-integer numbers in the data table were rounded; the records below are from the actual "`Samples\Chapter3\divorce4.raw`" (with added line numbers and wrapped lines):

```
 1  9,2,23,1,1,10.546,10.546,12,12,0,0,20.953,1.013,3.734,0,10.546,1,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99
 2  11,3,23,1,1,34.943,34.943,3,3,0,0,24.498,.687,.767,0,32.512,1,34.943,2,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99
 4  15,1,23,1,0,15.012,20.052,7,7,0,0,19.499,2.352,20.052,0,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99
 5  15,1,23,2,1,13.944,13.944,7,3,0,0,68.29,15.064,13.944,0,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99
 8  43,4,23,1,1,16.706,16.706,16,16,0,0,22.5,.162,3.083,0,4.578,1,6.664,2,
    16.706,3,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99
16  77,2,26,1,1,4.085,4.085,16,16,0,0,24.835,2.352,3.833,0,4.085,1,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99
17  77,2,26,2,0,6.557,6.634,16,16,0,0,31.819,2.352,3.428,0,6.634,1,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99
18  77,1,26,3,0,.709,.786,16,16,0,0,38.5,2.352,.786,0,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,-99,
    -99,-99,-99,-99,-99,-99,-99,-99,-99
```

The second control variable, `numint`, tells raw2aml how many relevant sets of level 3 variables there are. It also needs to be told to read a total of 17 sets. This is done in the raw2aml control file (`divorce4.r2a`):

```
 1  ascii data file = divorce4.raw;
 2
 3  level 1 var = weight;
 4  level 2 var = marnum censor lower upper hiseduc
 5                hereduc heblack sheblack age agediff;
 6  level 3 var (nb=17) = time numkids;
```

This file differs from its counterpart that handles compressed data (`divorce3.r2a`) in the "`nb=17`" specification on line 6. This specification tells raw2aml to always read 17 sets of level 3 variables. The syntax applies analogously to data with more than three levels:

```
level 1 var = <varlist>;
level 2 var = <varlist>;
level 3 var (nb=n3) = <varlist>;
level 4 var (nb=n4) = <varlist>;
level 5 var (nb=n5) = <varlist>;
et cetera…
```

where `n3`, `n4`, `n5`, et cetera, are the number of level 3, 4, 5, et cetera subbranches to read in within any level 2, 3, 4, et cetera, (sub)branch.

The resulting data file (`divorce4.dat`), is identical to the file created with compressed data (`divorce3.dat`). (Identical except for the creation date and time, which is stored toward the end of the data file.) The corresponding data documentation file (`divorce4.sum`) is also identical, except for two additional lines:

```
 1  Documentation for 'divorce4.dat'

    et cetera...

39  NOTE: all irrelevant level 3+ variables are equal to -99.
40  NOTE: relevant level 3+ variables are never equal to -99.
```

By default, raw2aml checks that all irrelevant sets of level 3 variables are –99, as they should be, and that none of the relevant values is equal to –99. If any irrelevant value is not –99, it generates an error message. If any relevant value is –99, it issues a warning. The two notes on lines 39 and 40 report on these data integrity checks.

If any relevant values are equal to –99, it is better to select another "special" value to represent irrelevant values. This may be done by:

```
option irrelevance check = n;
```

where *n* is the number you select as special value. The default value is –99. If you do not want raw2aml to check the values of relevant and irrelevant values at all, specify:

```
option irrelevance check = no;
```

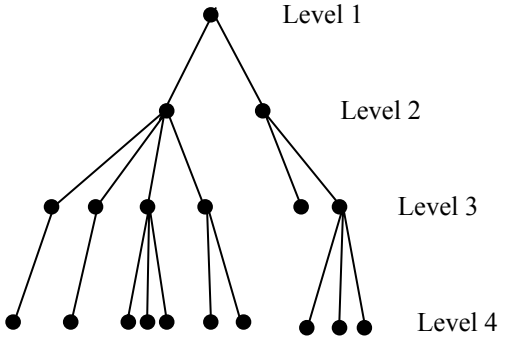
We recommend that you always allow raw2aml to check on the integrity of the data.

It is important to distinguish irrelevant values from missing values. Both types may be represented by a period (".") in your (SAS, Stata, SPSS) data, but they represent fundamentally different things. Irrelevant values arise when a number of subbranches is lower than the maximum in the data; missing values arise when a respondent is not able or willing to answer a question. Irrelevant values should be set to –99 (or other) before creating the ASCII data file. Raw2aml checks their value and discards them. The aML-formatted data will only contain relevant data. Missing values need to be resolved before creating the ASCII data file, by imputation or otherwise (Section 3.5).

Section 10.7 contains additional detail on rectangular data.

3.4. Choosing Appropriate Level Units

Multilevel data tend to be nested in some natural order. At the top of the hierarchy is the observation unit, which in aML is level 1. Anything within an observation may be modeled jointly; conversely, any two outcomes that belong to different observations are considered independent. If you think that they may be correlated, then you should redefine observation IDs such that the outcomes belong to the same observation, and retransform the data with raw2aml. By definition, observations are statistically independent, whereas outcomes within an observation are potentially correlated.



Multilevel Data

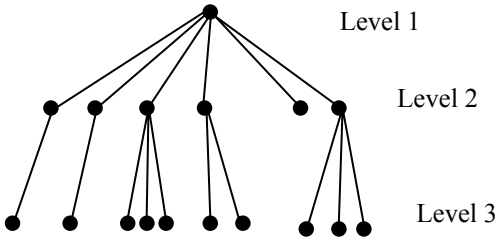
You should have a clear understanding of the conceptual levels in your data, because the model specification should explicitly account for corresponding correlations. For example, suppose you are interested in wage rates, particularly in any correlation between husbands and wives. Your panel data provide information on the annual wages that husbands and wives have earned on potentially multiple jobs over their career. As visualized in the figure, at level 1 is the couple, at level 2 the individuals (husband and wife). The husband held four jobs, the wife two (level 3 branches). For the husband's first and second jobs, one annual wage is available in the data; for his third job, we have three wage rates; for his fourth two wage rates (level 4 branches). For the wife's first job, no wage rate is available; for her second job, three wage rates are known.

It is perhaps most intuitive to structure your data in raw2aml and aML in correspondence with the conceptual level structure. In other words, the "technical" levels would correspond to the "conceptual" levels. However, *you do not need to adhere to the conceptual level structure*. In particular, it is often convenient to collapse your conceptual multilevel structure into fewer data layers. This section discusses that alternative and outlines the circumstances under which you may wish to deviate from the conceptual structure. With few exceptions, aML's model specifications are controlled by the values of variables, not by the level at which they are stored, and model estimation results will be identical under alternative ways of organizing the data into levels. For example, a person-specific heterogeneity component is specified by a residual which is constant across all outcomes of the individual. This is achieved by specifying a draw variable which is constant across all repeated outcomes of the individual—it does not matter whether that variable is at the outcome level or higher. We will briefly describe the exceptions toward the end of this section.

Consider again the example of households (level 1) with persons (level 2), jobs (level 3), and wage records (level 4). You could write out the data into ASCII data files such that the level

structure in the data corresponds to the conceptual level structure. Recall that an ASCII record must contain all information pertaining to a level 2 branch, i.e., all person-, job-, and wage-specific variables. If the unit of observation in your (SAS, Stata, SPSS) data is a person, this should not pose any problems. If the unit of observation is not a person, you may or may not need to go through data transformations, depending on the features of the data management package that you are using. If such data transformations are cumbersome, you may wish to consider an alternative level structure in which, for example, level 2 corresponds to a job.

Compare the figure to the right with the one above. The household remains unchanged at level 1, but we collapsed the level structure such that the six level 3 branches (jobs) in the conceptual structure are technically level 2 branches. The ten wage observations have become level 3 subbranches. From the figure, it is no longer possible to identify persons, because they are collapsed with jobs. However, in the data it is still possible to identify persons. The list of level 2 variables consists of both the conceptual level 2 (person-specific) and level 3 (job-specific) variables. All person-level variables are thus still in the data. Their values have been duplicated corresponding to the number of jobs that they contribute. In other words, the first person had four jobs and his characteristics are now four times in the data, duplicated for each job; the second person had two jobs, and the variables pertaining to her are twice in the data. The information is thus still in the data, and may be used to specify models.



Partially Collapsed Multilevel Data

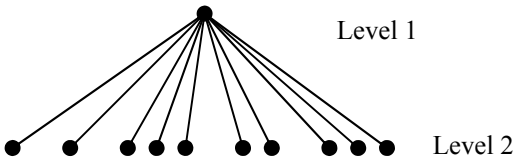
To illustrate this further, suppose wage rates are in variable `wage`, and the sex of a person is flagged by variable `male`. You may wish to specify different models for male and female wage determination. This may be done as follows:

```
continuous model; keep if male==1;  /* model for men */
   outcome = wage;
   model = ...;
continuous model; keep if male==0;  /* model for women */
   outcome = wage;
   model = ...;
```

As a result, the seven wage observations of the first person are captured by the first model statement, whereas the second model statement captures the three wage observations of the second person. If level 2 corresponds to a person (as in the conceptual level structure), there would only be two occurrences of the `male` variable; if level 3 corresponds to a job, there would be six. aML does not care either way. The first model's outcome is stored in variable `wage`. aML determines at what level that variable lives and loops over all subbranches at that level. For every subbranch, it checks the "`keep`" condition to determine whether this subbranch contributes to the model. The keep condition involves variable `male`, which happens to be a higher-level variable. This makes

no difference to aML; it simply evaluates the variable and decided whether to keep of drop each subbranch. The same procedure is carried out for the second model statement.

You could also collapse the data such that level 2 corresponds to a wage rate. aML would determine that outcome variable wage is at level 2, and loop over all ten level 2 branches. It would evaluate variable `male` in the keep condition, which would happen to be at the same level as the outcome. The first seven wage outcomes would be captured by the first model, and the next three was outcomes by the second model, just like before. It does not matter at what level the outcome is, and at what level variables are that enter the model equation(s).



Fully Collapsed Multilevel Data

The situation would be different if the outcome were not at the lowest level. For example, suppose you wish to analyze duration on each of the six jobs in the data, and the data are collapsed such that wage records are level 2 branches. There would then be ten level 2 branches, with all job-specific variables duplicated as many times as there are wage records for that job. The first person had three wage records on his first job, so that job's characteristics, including job duration, enters three times in the data. Unless you are very careful to only keep one occurrence, you would implicitly weight the outcomes by the number of wage records. This may bias the estimates and yield overly optimistic standard errors. (An additional problem is that the second person's first job did not have any wage information, and would have been dropped from data that were completely collapsed for wage analysis.) If the data were partially collapsed such that each job is a level 2 branch, there would be no duplication of job-specific variables, and the job duration analysis would not raise any special issues.

Collapsing levels requires duplication of variables, which implies that the data occupy more storage space than if no levels were collapsed. Raw2aml and aML's run times will be marginally higher because of additional time to read data. However, with relatively inexpensive disk storage and computer time, the reduced programming effort probably more than offsets this inefficiency.

In summary, each record in ASCII data files must contain all information pertaining to a level 2 branch. If the unit of observation in your (SAS, Stata, SPSS) data corresponds to a conceptually lower level, you may wish to collapse levels and create data in which the technical level 2 corresponds to the unit of observation in the (SAS, Stata, SPSS) data. However, in some cases, this may lead to undesired results:

- If one of the outcomes of interest is not at the lowest level, its values are being duplicated as it becomes the level 2 unit. Duplication implies that the same outcome enters multiple times in the model, leading to inflated t-statistics and potentially biased estimates of the parameters themselves. This may be solved by only keeping one occurrence of the duplicated outcomes. However, if the duplicated variables are censor and duration variables from a hazard model with time-varying covariates, you should not keep just one occurrence, because time-varying

information would be lost. Time-varying covariates should remain one level below the censor and spell duration variables.

- If your model contains autoregressive moving average (ARMA) or cumulative autoregressive (CAR) residuals, the level structure does matter for independence of residuals across branches of a specified level. ARMA and CAR residuals are autocorrelated across outcomes within a specified level, and independent across branches of the specified level. It is thus important to maintain conceptual levels in the data. See Sections 5.10, 13.2.7 and 13.2.8.

Barring those circumstances, the level structure may be chosen as convenient. The results of estimation will be equivalent.

Section 10.5 further discusses the choice of levels.

3.4.1. Cross-Classification

Cross-classification arises when subjects are not nested. For example, suppose we wish to analyze test scores and allow for both student and teacher effects. One student may have several teachers; every teacher has more than one student, and several students may have several teachers in common. In other words, the data are not nested.

aML can estimate (some) models with cross-classification, provided that there is some clustering at the highest level. For example, while students and teachers are not nested among each other, they are nested within schools. We may therefore specify a model along the following lines (see Section 4.1.2 for details on draws):

```
continuous model;
   outcome = score;
   model = ... +
           res(draw=1, ref=eps) +
           res(draw=teacher, ref=eta) +
           res(draw=student, ref=delta)+
           res(draw=_iid, ref=u);
```

Residual `eps` is drawn with the same draw (value) for all outcomes within an observation, i.e., the same value for all tests in a school; there is a different value of residual `eta` for every teacher; a different (transitory) residual `delta` for every student; and a different residual `u` for every test. The fact that teachers and students are not nested does not matter.

The ability to estimate models on cross-classified data breaks down when the cross-classification spreads across observations (level 1 units). For example, if teachers teach at multiple schools or if students switch between schools.

The ability to estimate models on cross-classified data also breaks down when residuals are integrated-out. The numerical integration algorithms require nesting (or independence across

draws, so that two distributions may be termed siblings rather than parent-child). This limits cross-classified models to continuous models only.

3.5. Missing Data and Character Variables

Most third-party statistical packages internally reserve one or more special numerical values to represent the "missing value." Upon input and output, such missing values are typically represented by a period (".."). Raw2aml and aML do *not* support such missing values. All numerical values must be legitimate numbers on the real line.

This does not imply that you may not use aML if your data contain missing values. It only implies that you need to resolve missing values before transforming your data into aML-format using raw2aml. One method for resolving missing values is to impute them. Another common method is to generate a separate indicator variable which flags whether the original variable is missing. If the original variable is missing, the indicator variable is one; else, it is zero. Now that there is a flag for missing values, missing values of the original variable may be set to some legitimate numerical value, such as zero or the mean over nonmissing values. (The latter provides a test for whether the variable is missing randomly.)

Before writing out ASCII data, make sure that all missing values are resolved. If periods (".") inadvertently enter ASCII data, raw2aml will abort with an error message. It will attempt to diagnose the problem and communicate its findings.

In rectangular data, it is important to distinguish between missing and irrelevant values. Irrelevant values occur when a level 2 or lower branch has fewer subbranches than the maximum number in the data. Irrelevant should be set to -99 before creating ASCII data files. Raw2aml will recognize that they are irrelevant and discard them. The aML-formatted data will only contain relevant values. Missing values occur, for example, when a respondent is not able or willing to respond to a question. These values must be resolved before creating ASCII data files.

Finally, aML does not support character string variables with values such as "`abc`" or "`yourname`". Only numerical variables are supported. If a character string inadvertently enters ASCII data, raw2aml will abort with an error message. Note that values like "`1.23E+02`" (without the double quotes) are interpreted correctly as numerical, not string values.

4. Multilevel and Multiprocess Models

This chapter discusses multilevel and multiprocess models. It assumes that you are familiar with the basics of model formulation and estimation (Section 2.1) and multilevel data preparation (Chapter 3). Section 4.1 explains multilevel models with unobserved heterogeneity, including multiple levels of unobserved heterogeneity. Section 4.2 explains multiprocess models, i.e., models with correlation across multiple types of outcomes. All sample data and control files used in this chapter may be found in the "`Samples\Chapter4`" directory.

4.1. Multilevel Modeling with Unobserved Heterogeneity

This section discusses models with unobserved heterogeneity. Section 4.1.1 explains how to estimate normally distributed unobserved heterogeneity in a two-level model. Section 4.1.2 extends the discussion to models with more than two levels and multiple nested heterogeneity components. Section 4.1.3 explains how to specify heterogeneity that follows a finite mixture distribution.

4.1.1. Two-Level Modeling with Unobserved Heterogeneity

This section explains how to incorporate stochastic variation (heterogeneity) in its most widely used form, a univariate normally-distributed residual.

Statistical models always include some error term to account for an imperfect fit between explanatory covariates and the outcome of interest. The error term is often explicitly written down in the form of a residual; in some cases, such as in hazard models, the error term is implicit. By their very nature, not much information is available on error terms. In most cases, we make some assumptions on residuals, such as that they are distributed according to some specific distribution. In many applications, however, particularly in multilevel settings, more information is available. For example, we may know that part of the error stems from unmeasured factors at some aggregated unit of observation, whereas the remaining stochastic variation is at the lower level. More generally, there may be multiple sources of stochastic variation, often corresponding to nested levels.

An important implication of stochastic variation at multiple levels is that repeated outcomes may not be independent. For example, suppose a mother's decision to deliver her baby in a hospital (as opposed to at home or elsewhere) is in part determined by her long-run health status, observed by the mother but unobserved by the analyst. Treating multiple hospital deliveries per mother as independent observations may lead to biased estimates of parameters and/or their standard errors. Instead, one must take into account that the multiple outcomes are correlated.

Capturing correlation across outcomes requires two important steps. First, in the data preparation stage, the user must indicate which outcomes are correlated by assigning them the same value of an identifying variable, i.e., the same ID. Second, in the model specification, the user must specify separate residuals for various levels that are the source of stochastic variation.

We discuss two examples, one probit and one hazard. Other processes are analogous.

Example: Probit Model with Heterogeneity

We are interested in the decision to deliver babies in a hospital versus at home or elsewhere, and have data on one or more births per female respondent. Section 3.2.1 above described the two-level data creation process. If you have not yet read that section, please do so now. Children are nested within mothers, so mothers form level 1 and children are at level 2. In other words, each child corresponds to a level 2 branch. The data, "`hospital.dat`" and their documentation file "`hospital.sum`" are in "`Samples\Chapter3`"; the aML control and output files are in "`Samples\Chapter4`".

We model the decision to deliver in a hospital as a probit:

$$H_j^* = \beta' X_j + \varepsilon + u_j ,$$

where H_j^* indicates the propensity that a woman delivers baby j ($j = 1,\ldots,J$) in a hospital. We suppress the woman-subscript. If $H_j^* < 0$, the baby is not delivered in a hospital ($H_j = 0$), and if $H_j^* \geq 0$, the baby is delivered in a hospital ($H_j = 1$). Observed characteristics at the woman and child level are captured by X_j; unmeasured characteristics are in part woman-specific and constant across all her J births (ε), and in part specific to individual births (u_j). Denote the standard deviation of ε by σ_ε; we normalize the standard deviation of the transitory residual, $\sigma_u = 1$.

In order to identify the standard deviation of the heterogeneity component, we need multiple outcomes on at least a subset of the observations; that requirement is fulfilled, because the data contain 501 mothers with a total of 1,060 children. As indicated by "`hospital.sum`", the maximum number of children per woman (maximum number of level 2 branches in any observation) is ten. Not shown is that 48 percent of the women in the data had only one child. These observations do not contribute to the identification of unobserved heterogeneity, but they do provide information on parameters β.

The outcome of interest (H_j) is variable `hospital`; explanatory variables are `income` (income), `distance` (from home to the nearest hospital, in km), and `educ` (education level of the

mother, coded 1 for less than high school, 2 for high school graduates, and 3 for college graduates). Maternal education is a level 1 variable; all others may vary from birth to birth.

To ensure smooth convergence of the maximum likelihood process, it is important to specify good starting values (Chapter 6). This, in turn, requires that you build up models step-by-step. In particular, start with only one residual and add heterogeneity at a later stage. aML control file `hosp1.aml` specifies the first model, a simple probit without heterogeneity. Having estimated this model, we move to `hosp2.aml`:

```
 1  option title = "Hospital delivery with heterogeneity";
 2
 3  dsn = ..\Chapter3\hospital.dat;
 4
 5  define regressor set BetaX;
 6     var = 1 log(income) distance (educ==1) (educ==3);
 7
 8  define normal distribution; dim=1; number of integration points = 6;
 9     name = eps;
10
11  probit model;
12     outcome = hospital;
13     model = regset BetaX +
14        intres(draw=1, ref=eps);
15
16  starting values;
17
18  Constant    TT   -1.5731679635
19  lnIncome    TT     .2642240564
20  distance    TT    -.0403876943
21  dropout     TT    -.9210922669
22  college     TT     .5029232968
23  SigmaEps    FT     .6
24  ;
```

Line 1 specifies an optional title. It only serves to document the main purpose of this run and will be printed at the beginning of the output file. Line 3 specifies the data set.

Line 5-6 define the regressors. We specify the logarithm of income and indicator variables for high school drop-outs and college graduates as explanatory variables.

Lines 8-9 specify the distribution of the heterogeneity component. The normal distribution was first introduced for continuous models (Section 2.3). New is the optional "`number of integration points=6`". It is used here because we will be integrating ε out numerically. See below.

Lines 11-14 specify the probit model. Line 14 indicates that residual `eps` needs to be integrated out ("`intres`" is short for "`integrated residual`"); see below.

Since ε is specific to the mother, its value must be constant across all hospital outcomes of any one woman. In other words, the same "draw" applies to all hospital outcomes. We therefore wrote `draw=1`, but `draw=247` or `draw=_id` or `draw=`*`expr`* where *`expr`* is any positive integer-valued expression or variable that takes on the same value within any one observation would give identical results. Section 4.1.2 provides details on correlated and independent draws.

The starting values were taken from converged estimates of `hosp1`. There is one new parameter, the standard deviation of `eps`. We initialize it arbitrarily at 0.6, which tends to be the right order of magnitude in most qualitative outcome models. We often find that the maximum likelihood search procedure has trouble improving the likelihood if standard deviations are initialized at (very close to) zero, as one may be inclined to do.

Note that we optimize this model in two stages. At first, the standard deviation of the heterogeneity component is fixed and the probit regressors are allowed to settle in. In the second stage, all parameters are free. Experience shows that freeing up too many parameters at once often leads to failure to converge.

The control file does not specify transitory residual u_j. If no non-integrated residual is specified in a probit model, aML implicitly assumes a standard normally distributed residual. The output file, `hosp2.out`, does write out this residual explicitly (line 54, not shown here): "`res(draw=_iid, ref=N(0,1))`". We could have included this line in the control file; `draw=_iid` automatically generates a new draw for every subbranch; `ref=N(0,1)` references an implicitly defined standard normal residual. We could have defined and used an explicit residual, which would need to be drawn using a variable that takes unique values for every child.

The stochastic variation that generates heterogeneity is typically specified as a residual which is specific to a unit that is at a higher conceptual level than the outcome. In this case it is a mother. The unobserved heterogeneity component (residual) takes on the same value (draw) for all outcomes of a particular mother. The covariance matrix of $\varepsilon + u_j$ is

$$\Sigma_{\varepsilon+u,\varepsilon+u} = \sigma_\varepsilon^2 1_J 1_J' + \sigma_u^2 I_J ,$$

where 1_J is a *J*-dimensional vector of ones, and I_J is the *J*-by-*J* identity matrix. For example, with *J*=3,

$$\Sigma_{\varepsilon+u,\varepsilon+u} = \begin{pmatrix} \sigma_\varepsilon^2 + \sigma_u^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 + \sigma_u^2 & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 + \sigma_u^2 \end{pmatrix}$$

The likelihood of *J* probit outcomes thus involves a *J*-dimensional cumulative normal integral, which is computationally very intensive for higher values of *J*. aML only supports such integrals for $J \le 3$. For higher values, residual ε must be integrated out numerically. The joint likelihood of *J* probit outcomes may be written as:

$$L = \int_{\varepsilon=-\infty}^{\infty} f(\varepsilon) \prod_{j=1}^{J} \Phi\left(\left(2H_j - 1\right) \left(\frac{\beta' X_j + \varepsilon}{\sigma_u} \right) \right) d\varepsilon ,$$

where $f(\varepsilon) = \phi(\varepsilon) / \sigma_\varepsilon$ is the probability density function of ε. The *J*-dimensional cumulative normal distribution has thus been reduced to a product over *J* univariate cumulative normal distributions, which is far easier to compute.

In short, unobserved heterogeneity in probit models may be specified as

`res(draw=`*`expr`*`, ref=`*`residualname`*`)`

provided that there are no more than three probit outcomes in any one observation. In most applications, there are more than three outcomes, and the residual must then be integrated out:

`intres(draw=`*`expr`*`, ref=`*`residualname`*`)`

Even with three or fewer probit outcomes, you may tell aML to integrate-out the residual. We recommend that you always integrate-out unobserved heterogeneity components in probit models, because it reduces errors and because the integration algorithm tends to be faster than bivariate and trivariate cumulative normal probability algorithms. The same holds for ordered probit and normal interval models.

In continuous models, a closed-form solution to the likelihood function exists, and you have a choice between specifying the residual directly ("`res`") or integrated-out ("`intres`"). In (ordered) logit, hazard, binomial, and negative binomial models, no closed-form solution exists, and the residual must be integrated out ("`intres`"). We recommend that you specify the residual directly ("`res`") in continuous model statements, even though the same residual may be integrated out ("`intres`") in other, jointly estimated model statements.

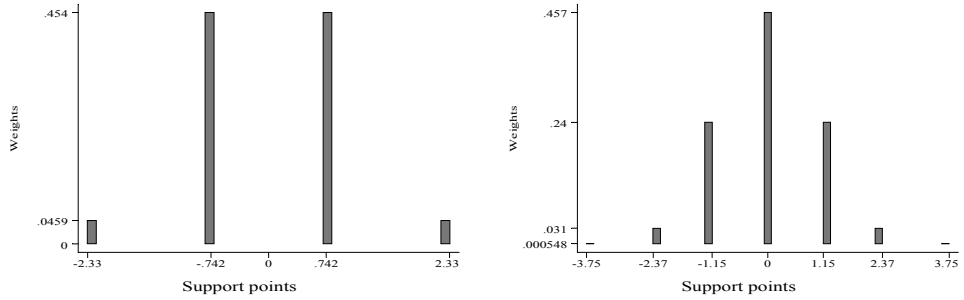
aML integrates-out residuals using a numerical integration algorithm based on Gauss-Hermite Quadrature (e.g., Abramowitz and Stegun, 1972, pp 890 and 924). This algorithm selects a number of support points and weights such that the weighted points approximate the normal distribution. The higher the number of support points, the more accurate the approximation, but the slower the computation. By default, aML approximates univariate normal distributions by 12 points (Section 13.2.6). In the example, we instead instructed aML to use only six points.

Technical Note: Numerical integration

Many models require that one or more residuals are integrated out. Where a closed form solution to the integral does not exist, the likelihood may be computed by approximating the normal integral by a weighted sum over "conditional likelihoods," i.e., likelihoods conditional on certain well-chosen values of the residual. aML offers Gauss-Hermite Quadrature to approximate normal integrals (e.g., Abramowitz and Stegun, 1972, pp 890 and 924):

$$\int_{-\infty}^{\infty} \phi(\varepsilon) L(\varepsilon) dx \approx \sum_{l=1}^{k} w_l L(e_l)$$

where w_l and x_l are Gauss-Hermite weights and support points, respectively. The user has control over the number of support points k; the higher the number of points, the more precise the approximation (and the more time it takes aML to compute the likelihood). The figures below illustrate the support points (horizontal axis) and weights (height of the bars) implied by four and seven support points, respectively.



Utility program points, bundled with the aML package, provides support points and weights for the univariate standard normal distribution. It is documented in Section 15.4. For example, typing "`points 4`" from the command line gives the approximation with four points (also see the left figure above):

	point	weight
1	-2.334414	0.0458759
2	-0.741964	0.4541241
3	0.741964	0.4541241
4	2.334414	0.0458759

In `hosp2.aml`, we replace the integral over ε by a sum over six support points (`number of integration points=6`). The higher the number of support points, the more accurate the approximation of the normal distribution. In practice, we find that four or six points provide estimates that are close to accurate; for the final run, a higher number, like twelve, is recommended. To illustrate the differences, the table below shows parameter estimates without heterogeneity and with heterogeneity that is integrated-out with four, six, and 20 support points:

	no heterogeneity	4 points	6 points	20 points
`Constant`	-1.5732 ***	-1.8814 ***	-1.8869 ***	-1.8957 ***
	(0.1984)	(0.2444)	(0.2453)	(0.2458)
`lnIncome`	0.2642 ***	0.3136 ***	0.3139 ***	0.3148 ***
	(0.0301)	(0.0354)	(0.0356)	(0.0356)
`distance`	-0.0404 ***	-0.0424 **	-0.0423 **	-0.0420 **
	(0.0142)	(0.0175)	(0.0176)	(0.0176)
`dropout`	-0.9211 ***	-1.1405 ***	-1.1428 ***	-1.1449 ***
	(0.0787)	(0.1403)	(0.1431)	(0.1439)

	no heterogeneity	4 points	6 points	20 points
college	0.5029 ***	0.6021 ***	0.6016 ***	0.6006 ***
	(0.1453)	(0.2286)	(0.2307)	(0.2319)
SigmaEps		0.6871 ***	0.7043 ***	0.7100 ***
		(0.0961)	(0.1010)	(0.1034)
ln-L	-538.49	-523.77	-523.65	-523.59

The table shows that most parameter estimates are substantially different with and without heterogeneity. However, the differences between estimates based on four, six, and 20 support points are relatively minor. The column with six points is based on "hosp2.out"; it shows that σ_ε is estimated at a very significant 0.7043.

You may, of course, define multivariate normal distributions and integrate those numerically. The number of required function evaluations increases rapidly with the dimension of the distribution. For example, with six integration points per dimension, a trivariate distribution requires 6^3=216 function evaluations. By default, aML uses 12 points for univariate distributions, eight points for bivariate distributions (implying 64 function evaluations), and six points for distributions with three or more dimensions.

Example: Hazard Model with Unobserved Heterogeneity

Stochastic variation (heterogeneity) in hazard and other types of models is implemented in the same way as in probit models. In all models except continuous models (and probit models with up to three replications), the heterogeneity residual must be integrated out. In continuous models, the residual may be integrated out, but estimation is often faster when the residual is included in the model as a (non-integrated) res, rather than an intres.

We provide one more example, for hazard models. Recall Section 2.4, in which we analyzed the timing of divorce for first marriages, and Section 3.2.2, in which we prepared three-level data for multiple marriages with time-varying covariates. If you have not read those sections, please do so now. Here we estimate all marriages jointly, including second and higher order. Directory "Samples\Chapter3" contains the data (divorce3.dat); directory "Samples\Chapter4" contains the aML control and output files discussed below. The data documentation file (divorce3.sum) indicates that there are 3,371 respondents with a total of 4,238 marriages, i.e., there are multiple marriages for at least a subset of the respondents. The maximum number of marriages for any one respondent is six.

The model is

$$\ln h_j(t) = \gamma T(t) + \beta' X_j(t) + \delta,$$

where subscript j to the marriage number; the respondent subscript is suppressed. Level 1 is a respondent, level 2 a marriage, and level 3 an interval of a marriage spell. Time-varying variables are constant over intervals, but may differ across intervals.

Building on our earlier estimates of the divorce hazard of the first marriage in "div2.out" (Section 2.4), we apply the model specification to the expanded data in "div3.aml", include two dummy variables for marriage order, and estimate it. (We first only estimate the intercept to allow for major differences between first marriages and subsequent ones. In the second round, all parameters are freed up.) We then add heterogeneity in "div4.aml":

```
 1  option normweight = weight;
 2
 3  dsn = ..\Chapter3\divorce3.dat;
 4
 5  define spline DurMar; nodes = 1 4 15 25;
 6
 7  define regressor set Getdiv;
 8     var = 1 (marnum==2) (marnum>=3) heblack (hiseduc<12) (hiseduc>=16)
 9      (agediff>10) (agediff<-10) (heblack!=sheblack)
10      numkids;
11
12  define normal distribution; dim=1; number of integration points = 4;
13     name= delta;
14
15  hazard model;
16     censor=censor; duration=lower upper; timemarks=time;
17     model = durspline(origin=0, ref=DurMar) +
18        regset Getdiv +
19        intres(draw=1, ref=delta);
20
21  starting values;
22
23  dur0-1      TT     1.818046395
24  dur1-4      TT     .0909935637
25  dur4-15     TT     -.038456674
26  dur15-25    TT    -.0238131819
27  dur25+      TT    -.1297594318
28  Constant    TT   -5.6057629476
29  mar2        TT     .2381876859
30  mar3+       TT     .6093417913
31  heblack     TT      .010438415
32  mixrace     TT     -.316479448
33  dropout     TT    -.3047659902
34  college     TT    -.5210636007
35  heolder     TT     .1990443217
36  sheolder    TT     .3600968963
37  numkids     TT    -.0788632828
38  SigDelta    FT      .6
39  ;
```

Note line 8, where we added two dummy variables to the regressors. The first, (marnum==2) captures second marriages; the second, (marnum>=3) captures third and higher order marriages. The omitted category is first marriages.

Lines 12-13 define the heterogeneity distribution in the same way as seen in the probit model. Line 19 includes δ (delta) in the hazard model. Since δ is specific to the respondent, its value

is the same in all marriages. This is achieved by `draw=1`, i.e., draw is the same for all marriage spells. See Section 4.1.2 for more details on correlated and independent draws.

Lines 21-39 specify the starting values. We took converged estimates from `div3` and added the standard deviation of the heterogeneity distribution, `SigDelta` at an initial value of 0.6. As before, we were careful not to estimate all parameters at once. First we allowed all parameters to settle in under the specification with heterogeneity, and only then did we estimate all parameters. The results are in "`div4.out`"; we summarize them along with those from `div3` using the mktab utility:

	div3	div4
dur0-1	1.8180 ***	1.8266 ***
	(0.4587)	(0.4615)
dur1-4	0.0910 **	0.1361 ***
	(0.0439)	(0.0459)
dur4-15	-0.0385 ***	-0.0112
	(0.0108)	(0.0120)
dur15-25	-0.0238	-0.0152
	(0.0163)	(0.0165)
dur25+	-0.1298 ***	-0.1253 ***
	(0.0207)	(0.0208)
Constant	-5.6058 ***	-6.0942 ***
	(0.4144)	(0.4240)
mar2	0.2382 ***	-0.2664 **
	(0.0875)	(0.1249)
mar3+	0.6093 ***	-0.5792 ***
	(0.1439)	(0.2107)
heblack	0.0104	-0.0358
	(0.1385)	(0.1718)
dropout	-0.3165 ***	-0.3447 ***
	(0.0634)	(0.0794)
college	-0.3048 ***	-0.3488 ***
	(0.0874)	(0.1068)
heolder	-0.5211 ***	-0.5075 ***
	(0.1729)	(0.1868)
sheolder	0.1990	0.2332
	(0.2477)	(0.2975)
mixrace	0.3601 **	0.4982 ***
	(0.1526)	(0.1904)
numkids	-0.0789 ***	-0.1120 ***
	(0.0252)	(0.0291)
SigDelta		0.9996 ***
		(0.1099)

```
ln-L         -7176.11       -7162.88

NOTE: Asymptotic standard errors in parentheses;
      Significance: '*'=10%;  '**'=5%;  '***'=1%.
```

Note that the standard deviation of δ is 0.9996 and significantly different from zero. In other words, there are unmeasured respondent-specific characteristics which affect all marriages in which respondents engage. Failure to account for these has several consequences. First, a specification in which correlation across marriages of the same person is ignored (`div3`) tends to underestimate the standard errors of parameter estimates, creating false impression of precision. Second, estimates of the baseline duration pattern are biased in downward direction.[22] Third, estimates of indicators for second and higher order spells are biased upward. The conclusion from `div3` would be that higher order marriages are more prone to divorce, whereas `div4` shows that this is entirely due to heterogeneity. Individuals with unmeasured characteristics which help stabilize marriages tend to remain in their first marriage; second and higher order marriages are dominated by individuals with "risky" characteristics. For the average person, higher order marriages are in fact more likely to survive, as the negative coefficients on variables `mar2` and `mar3+` in `div4` show.

4.1.2. Multilevel Modeling with Multilevel Unobserved Heterogeneity

The previous section illustrated models with repeated outcomes at level 2 and a single unobserved heterogeneity component that accounted for their correlation. We now show how to specify models with multiple levels of heterogeneity. The heterogeneity components illustrated here are nested but independent; Section 4.2 explains correlated heterogeneity components in a multiprocess environment.

Consider data on schools (level 1) with one or more students (level 2). Each student was enrolled one or more years (level 3) and took multiple tests each year (level 4). Suppose you are interested in analyzing test scores. In addition to controls for various observed explanatory covariates, you wish to allow for school-specific effects, for example because some schools are more successful at creating an environment that fosters student learning than others. You also wish to allow for student-specific effects, for example because some students are more intelligent than others. You also wish to allow for effects that are specific to year-in-school, for example because students' home situation may be more conducive to learning in some years than in others. Finally, you wish to allow for remaining variation at the test level. The first three effects are

[22] The best way of understanding this is by imagining a process with a constant hazard, say, repeated throws with a dice where the players stop playing when they throw a six. If there is heterogeneity, for example because the dice are manipulated, players with "lucky" dice will be among the first to stop playing. Initially, the aggregate hazard of throwing a six is 1/6, but after a while there are mostly "unlucky" dice left, so that the hazard appears to be lower than 1/6.

stochastic variation, or heterogeneity at levels higher than the specific test; the test-specific residual captures the lowest level of variation. The data are included with the aML files under "`Samples\Chapter3`" and explained in detail in Section 3.2.3 above. The control and output files of this section are in "`Samples\Chapter4`".

The model is a simple continuous outcome model with four nested residuals:

$$s_{ijk} = \beta' x_{ijk} + \delta + \varepsilon_i + \eta_{ij} + u_{ijk},$$

where outcome s_{ijk} is the test score of student i in year j on test k. The school (observation) subscript has been suppressed. Explanatory covariates x_{ijk} include indicators for whether the school is located in an innercity and whether it is a private school, the student's sex and his parental education, whether he is duplicating the current school year, and class size. Heterogeneity component δ captures unobserved school-specific characteristics, δ_i unobserved student-specific characteristics, and η_{ij} unobserved year-specific characteristics. Residual u_{ijk} captures any remaining variation. The likelihood function of the vector of all test scores for a school is the multivariate normal density function:

$$L = (2\pi)^{-k/2} |\Sigma|^{-1/2} \exp\left\{-\tfrac{1}{2}(S - X\beta)' \Sigma^{-1} (S - X\beta)\right\},$$

where S is the vector of stacked score outcomes, X the matrix of covariates, and Σ the joint covariance matrix. For example, consider a school with two students who attended two and three years and took two tests each every year. Its equations may be written as:

$$S = X\beta + \Lambda_\delta \delta + \Lambda_\varepsilon \varepsilon + \Lambda_\eta \eta + U, \text{ i.e.,}$$

$$\begin{pmatrix} s_{111} \\ s_{112} \\ s_{121} \\ s_{122} \\ s_{211} \\ s_{212} \\ s_{221} \\ s_{222} \\ s_{231} \\ s_{232} \end{pmatrix} = \begin{pmatrix} x'_{111} \\ x'_{112} \\ x'_{121} \\ x'_{122} \\ x'_{211} \\ x'_{212} \\ x'_{221} \\ x'_{222} \\ x'_{231} \\ x'_{232} \end{pmatrix} \beta + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \delta + \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \\ \eta_5 \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \end{pmatrix}$$

and the joint covariance matrix is

$$\Sigma = \Lambda_\delta \Sigma_{\delta\delta} \Lambda'_\delta + \Lambda_\varepsilon \Sigma_{\varepsilon\varepsilon} \Lambda'_\varepsilon + \Lambda_\eta \Sigma_{\eta\eta} \Lambda'_\eta + \Sigma_{uu},$$

where $\Sigma_{\delta\delta} = \sigma_\delta^2$, $\Sigma_{\varepsilon\varepsilon} = I_2\sigma_\varepsilon^2$, $\Sigma_{\eta\eta} = I_5\sigma_\eta^2$, $\Sigma_{uu} = I_{10}\sigma_u^2$, and I_n is the n-dimensional identity matrix.

Note that the same draw of δ applies to all outcomes in the school (observation). Similarly, one draw of ε_i applies to all tests of student *i*, but independent draws apply to other students at the same school. One draw of η_{ij} applies to all tests of student *i* in year *j*, but independent draws apply to tests in other years and of other students. All draws of u_{ijk} are test-specific and thus independent from everything else. In other words, the residual components are nested.

In the data, students are identified by a student ID variable, `student`, which is unique within a school but not across schools. Years in school are identified by variable `year`. There are up to 18 students per school, up to six years in school per student, and up to seven tests per year in school. The model may be specified as follows (`school.aml`):

```
option maximum number of residual draws = 400;
option maximum model space = 15000;
dsn = ..\Chapter3\school.dat;

define regset BetaX;
   var = 1 innercty private male (dadeduc==1) (dadeduc==3)
         (momeduc==1) (momeduc==3) retain clssize;

define normal distribution; dim=1; name=delta;
define normal distribution; dim=1; name=eps;
define normal distribution; dim=1; name=eta;
define normal distribution; dim=1; name=u;

continuous model;
   outcome = score;
   model = regset BetaX +
     res(draw=1, ref=delta) +
     res(draw=student, ref=eps) +
     res(draw=10*student+year, ref=eta) +
     res(draw=_iid, ref=u);

starting values;

Constant    TTTT      6.532314
innercty    TTTT     -.1763978
private     TTTT      .3150571
male        TTTT     -.1417327
dad<HS      TTTT     -.2965861
dad>HS      TTTT      .3286746
mom<HS      TTTT     -.3543066
mom>HS      TTTT      .4040352
retain      TTTT     -.3711627
clssize     TTTT      .0054275
SigDelta    FTTT      .43
SigEps      FFTT      .43
SigEta      FFFT      .43
SigU        TTTT      .43
```

```
38  ;
```

For now please ignore lines 1 and 2. Line 3 specifies the data set and lines 5-7 defines the explanatory covariates. Lines 9-12 define four independent univariate distributions corresponding to δ, ε_i, η_{ij}, and u_{ijk}, respectively. Lines 14-20 specify the continuous outcome model. aML determines that the outcome, `score`, is a level 4 variable, and will loop over all level 4 branches in the data to compile an equation.

The important portion of this control file is in the residual draw specifications of lines 17-20. The same δ, i.e., the same draw of δ applies to all test score equations of a particular school (observation). This is achieved by "`draw=1`": the draw is the same for all equations. The actual draw value is irrelevant; the important thing is that the draw value is the same for all test scores. We could thus have specified "`draw=348`" or "`draw=_id`" or anything else that evaluates to the same value for all equations. Residual ε_i is specified in line 18 with "`draw=student`". Recall that variable student is the student ID. It takes on different values for each student, and aML accordingly draws a new ε_i for each student. Residual η_{ij} is specified in line 19 with "`draw=10*student+year`". We needed to specify a draw expression that takes on the same value for all tests taken in a particular year, but different from all other years and students. A "`draw=year`" would have been incorrect, because this would have assigned the same draw to all students' test score equations in a given year, i.e., we would have specified η_j instead of η_{ij}. The maximum number of years any one student is in school is six, i.e., `1<=year<=6`; this ensures that `10*student+year` is unique for every student and every year. Residual u_{ijk} is specified on line 20 with "`draw=_iid`". This "`_iid`" is not a variable or expression; it is a keyword which ensures that the residual is drawn independently in every equation. We could have specified "`draw=100*student+10*year+testnum`", where `testnum` is the test number, with equivalent results.

A thorough understanding of residual draws is critical to proper specification of multilevel models in aML. Sections 4.1.2 and, in particular, 13.3.6 provide additional detail.

Lines 22-38 specify starting values. We first ran an ordinary least squares regression model using the (SAS, Stata, SPSS) data preparation package. That model assumes a single transitory residual and ignores heterogeneity, but the resulting estimates tend to provide good starting values for more complicated specifications. Alternatively, we could have started the constant at the mean of `score` and all other covariates at zero, but the maximum likelihood search procedure tends to be slow in finding solutions to simple continuous models. See Section 6.5 for more tips on selecting starting values for continuous models. The ordinary least squares regression estimated the root mean squared error of its residual to be 0.85625, i.e., a mean squared error (variance) of 0.733. For lack of better ideas, we roughly divided this variance over the four residuals: the square root of one-fourth of 0.733 is about 0.43, as on lines 34-37.

We specified four rounds of optimization. In the first, only the covariates and standard deviation of the transitory residual are estimated; in subsequent rounds, standard deviations of the

other residuals are freed up one-by-one. Never be too ambitious when initially estimating complex multilevel models, as the search is prone to fail. Instead, nudge the model in small steps to its full specification.[23]

Lines 1 and 2 were inserted in response to aML's complaints that insufficient scratch storage space was available for the model. As shown in the data documentation file, "`school.sum`" (Section 3.2.3), there are as many as 343 tests in any one school, across all students and years in school combined. This implies that there are as many as 343 independent draws of u_{ijk}. By default, aML only allows up to 100 independent draws of any one distribution. When we first ran this model, aML reported that this default value was exceeded and suggested that we increase it using "`option maximum number of residual draws`". In the next run, aML complained that insufficient scratch space was allocated to store (up to 343) model equation specifications. It suggested that we increase the space using "`option maximum model space`". Lines 1 and 2 allocate sufficient scratch space. You do not have to worry about such allocations; aML will tell you what to do when it needs more space allocated.

Output file "`school.out`" contains the results of estimation.

4.1.3. Finite Mixture Heterogeneity

In addition to normally distributed residuals, aML offers finite mixture distributions. Only the univariate asymmetric finite mixture has been implemented. Asymmetry refers to the lack of any restriction which forces symmetry of support points or weights around zero. The mean of the distribution therefore depends on the location of support points and the weights. This implies that one of the support points (or equivalently, the intercept) is not identified and must be fixed in the estimation procedure.

Both support points and weights are implemented using aML's vector building block. A vector is a set of parameters and may be used directly to represent support points. For example, the following defines a vector representing three support points:

```
define vector Points; dim=3;
```

Support points must be strictly increasing, which is achieved by the restriction that parameters in a vector definition are strictly increasing. This restriction is imposed by default.

Weights are also specified using a vector. Since weights must always add up to one, a finite mixture distribution with n support points only has n-1 weights and thus requires a vector with dimension n-1. With three support points:

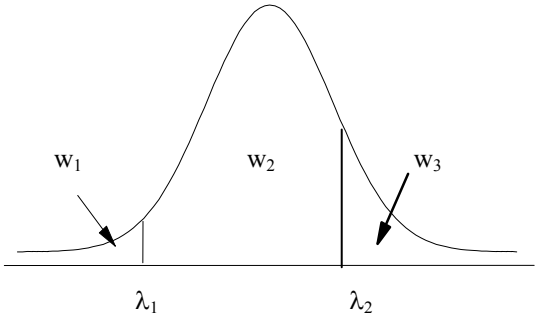
[23] If the model requires substantial computing time, as in the school test score example, you may wish to loosen convergence criteria for intermediate steps. When we developed the example, we initially specified "`option converge = wgn<4`" (page 276). This yielded results in the vicinity of the maximum likelihood parameters. In a final run, we reverted to the default criterion (`wgn<0.1`).

```
define vector Weights; dim=2;
```

This vector implies two parameters. For technical reasons, these two parameters are not equal to the weights themselves. Instead, the weights are transformations of the two parameters. Recall that weights must always add up to one. This restriction is implemented by computing weights as (cumulative normal) transformations of the elements of the vector forming the basis for weights. Denote elements of the vector by λ_1 through λ_{n-1}; weights w_1 through w_n are:

$$
\begin{aligned}
w_1 &= \Phi(\lambda_1) \\
w_2 &= \Phi(\lambda_2) - \Phi(\lambda_1) \\
&\vdots \\
w_{n-1} &= \Phi(\lambda_{n-1}) - \Phi(\lambda_{n-2}) \\
w_n &= 1 - \Phi(\lambda_{n-1})
\end{aligned}
$$

Also see the figure. Clearly, vector elements to be mapped into weights must be strictly increasing, which is aML's default behavior for vectors. A future release may allow for direct specification of weights; at present, you will need to transform vector estimates into weights yourself.



Mapping from *n*-1 vector elements to *n* weights

A finite mixture distribution is made up of support points and weights. Points are given directly from a vector; weights are derived from another vector through a cumulative normal transformation. The dimension of the weights vector is equal to the dimension of the points vector minus one. Note that there is no relationship between the pints and weights vectors, except that they are both part of a finite mixture distribution.

Section 4.1.1 showed an example in which hospital deliveries were modeled including a normally distributed heterogeneity component. Building on that example, "`hosp3.aml`" shows the syntax incorporating finite mixture heterogeneity instead:

```
 1  option title = "Hospital delivery with finite mixture heterogeneity";
 2
 3  dsn = ..\Chapter3\hospital.dat;
 4
 5  define regressor set BetaX;
 6     var = 1 log(income) distance (educ==1) (educ==3);
 7
 8  define vector Points;  dim=3; initial = GHQ points(std=.7042604784);
 9  define vector Weights; dim=2; initial = GHQ weights;
10
11  define finite mixture distribution; dim=1;
12     form = asymmetric;
13     points = Points;
```

```
14      weights = Weights;
15      name = eps;
16
17   probit model;
18      outcome = hospital;
19      model = regset BetaX +
20         intres(draw=1, ref=eps);
21
22   starting values;
23
24   Constant     T    -1.8868717781
25   lnIncome     T     .31390391051
26   distance     T    -.04228419725
27   dropout      T    -1.1428187936
28   college      T     .60158167422
29   point1       F     auto
30   point2       T     auto
31   point3       T     auto
32   weight1      T     auto
33   weight2      T     auto
34   ;
```

Lines 8 and 9 define the vectors for finite mixture support points and weights. (Ignore for now the optional "initial" statements in the vector definitions.) Lines 11-15 define the finite mixture distribution. It must always be univariate (`dim=1`) and asymmetric (`form=asymmetric`). For the support points we specified vector `Points`, for the weights vector `Weights`. Since `Points` was defined as a vector with three elements (`dim=3`), the finite mixture distribution will have three support points. Correspondingly, the weights vector has two elements, because the third weight follows from the other two such that they add up to one. Lines 17-20 specify the probit model. The finite mixture residual (line 20) enters as an integrated residual—finite mixture residuals must always be integrated out. The use of finite mixture residuals in model statements is exactly the same as the use of normally distributed residuals.

The starting values of regressors were taken from converged values of the model specification with a normally distributed heterogeneity component in "`hosp2.aml`". Lines 29-33 instruct aML to automatically generate the starting values of the `Points` and `Weights` vectors. This is an optional feature. Starting values of vectors that serve as finite mixture support points and weights are the only building blocks for which you do not need to specify numerical starting values. The "auto" starting values are invoked by the optional "`initial`" statements in the vector definitions; see lines 8 and 9:

```
define vector Points;  dim=3; initial = GHQ points(std=.7042604784);
define vector Weights; dim=2; initial = GHQ weights;
```

The optional "`initial=GHQ points`" statement in line 8 indicates that the initial values (starting values) of this vector should be equal to Gauss-Hermite Quadrature points that approximate a univariate normal distribution with standard deviation equal to 0.7042604784. That standard deviation was the converged estimate of the standard deviation of the normally distributed heterogeneity component in "`hosp2.out`". Similarly, the optional "initial=GHQ

weights" statement on line 9 indicates that the initial values of this vector should be such that its elements map into Gauss-Hermite Quadrature weights. In other words, we initialized the points and weights such that they correspond to the numerical approximation that would be used if we had specified a normally distributed residual with three points of support.

File "hosp3.out" contains the output, including the starting values that apply:

```
 82  Starting values:
 83        Name       Est?      Value
 84     1  Constant    T      -1.886872
 85     2  lnIncome    T       0.313904
 86     3  distance    T      -0.042284
 87     4  dropout     T      -1.142819
 88     5  college     T       0.601582
 89     6  point1      F      -1.219815
 90     7  point2      T       0.000000    (must be larger than predecessor)
 91     8  point3      T       1.219815    (must be larger than predecessor)
 92     9  weight1     T      -0.967422
 93    10  weight2     T       0.967422    (must be larger than predecessor)
```

Lines 89-91 indicate the automatically generated starting values of the support points. Since we estimate an intercept, the first point must be fixed. Lines 92-93 show the automatically generated starting values for the vector being transformed into weights. The actual weights are $w_1 = \Phi(-0.967422) = 0.1667$, $w_2 = \Phi(0.967422) - \Phi(\text{-}0.967422) = 0.6667$, and $w_3 = 1 - \Phi(0.967422) = 0.1667$. The mean of the distribution is zero, and the standard deviation $\sigma = \sqrt{w_1(p_1)^2 + w_2(p_2)^2 + w_3(p_3)^2} = \sqrt{0.1667*(-1.22)^2 + 0.6667*0^2 + 0.1667*(1.22)^2} = 0.7043$, as it should be.

The converged support points and weights differ somewhat from the values that best approximate the normal distribution with three points. Unfortunately, no formal test is available to test for any significant departure from normality. Heuristically, one may compare estimates of the regressors (which tend to be of most substantive interest) under normality (hosp2.out) and 3-point finite mixture (hosp3.out):

	hosp2	hosp3
Constant	-1.8869 ***	-0.9026 ***
	(0.2453)	(0.3080)
lnIncome	0.3139 ***	0.3083 ***
	(0.0356)	(0.0352)
distance	-0.0423 **	-0.0419 **
	(0.0176)	(0.0177)
dropout	-1.1428 ***	-1.1481 ***
	(0.1431)	(0.1377)
college	0.6016 ***	0.5422 ***
	(0.2307)	(0.1972)

	hosp2	hosp3
SigmaEps	0.7043 ***	
	(0.1010)	
point1		-1.2198
point2		-0.0754
		(0.7125)
point3		0.9976
		(1.1443)
weight1		0.6825
		(0.5744)
weight2		1.7147
		(1.2015)
ln-L	-523.65	-521.91

For all practical purposes, in the current example, it does not matter much whether normality is assumed to be normal or 3-point finite mixture. This result is specific to the current example and does not generalize.

4.2. Multiprocess Modeling

All examples we discussed so far featured one or more repeated outcomes of the same type. This section illustrates multiprocess models, i.e., models with two or more substantively different outcomes. Some examples of two-process models include health status and mortality; getting pregnant and finishing high school; marriage formation and marriage dissolution; job duration and annual wage income; fertility and divorce; use of health care and health outcomes; labor force participation and wage rate; et cetera. Each process may, but need not, include repeated outcomes.

We provide two illustrations involving two types of models; the concepts generalize to other types of models. Section 4.2.1 discusses how correlation across equations from substantively different outcomes may be captured using multivariate normal distributions. It limits the discussion to recursive models, i.e., models in which one type of outcome affects the other, but not vice versa. Section 4.2.2 explains how to estimate fully simultaneous models in which all types of outcomes may affect all other types.

4.2.1. Recursive Relationships of Replicated Outcomes

Consider an extension of the hospital delivery example of Section 4.1.1. We want to assess the benefits of delivering in a hospital for the health of the baby. We start by estimating a child mortality hazard equation in which we control for whether the child was delivered in a hospital, but in which we treat hospital delivery as an exogenous covariate. Data set "`children.dat`" is identical to the hospital delivery data (`hospital.dat`) used in Section 4.1.1, but also contains information on the children's subsequent survival. Variables `censor`, `durvar1` and `durvar2` are mortality hazard spell censor and duration variables. The mortality model is:

$$\ln h_j(t) = \gamma T(t) + \alpha' X_j + \delta$$

where subscript *j* indicates child number. The subscript for mothers (level 1) is suppressed. The baseline log-hazard $\gamma T(t)$ is assumed to be piecewise linear in the child's age; X_j represents regressors, including hospital delivery; and ε captures unobserved heterogeneity at the mother level, $\delta \sim N(0, \sigma_\delta^2)$. We build up the model in stages: "`Samples\Chapter4\child1.aml`" estimates a Gompertz baseline hazard only; "`child2.aml`" adds nodes and regressors; and "`child3.aml`" adds heterogeneity:

```
option title = "Child mortality w/ heterogeneity";

dsn=children;

define spline Age; intercept; nodes = .25 1 10;
define regressor set AlphaX; var = (educ==1) (educ==3) boy hospital;
define normal distribution; dim=1; number of integration points=4;
   name=delta;
```

```
 9
10  hazard model;
11     censor = censor;
12     duration = durvar1 durvar2;
13     model = durspline(origin=0, ref=Age) +
14             regset AlphaX +
15             intres(draw=1, ref=delta);
16
17  starting values;
18
19  Constant    TT     .3390926032
20  slope0      FT   -12.750487112
21  slope1      FT   -2.8845587617
22  slope2      FT    -.2507683224
23  slope3      FT     .0894649445
24  dropout     FT     .2666907388
25  college     FT   -1.7933428577
26  boy         FT     .1634632382
27  hospital    FT     -.346834987
28  SigDelta    TT      .6
29  ;
```

The specification is similar to those of hazard models discussed above. The "Age" spline with nodes at 3 months, 1 year, and 10 years serves as baseline duration dependency. The regressors include maternal education (less than high school, college), sex of the child, and whether the child was delivered in a hospital. The normal distribution's residual represents δ. It must be integrated out in hazard models (Section 4.1.1); we specify that the approximation uses four support points. The starting values were taken from converged parameters of the specification without heterogeneity (`child2.out`). We initialize the standard deviation of δ to 0.6, which tends to be the right order of magnitude. The intercept and the standard deviation of the heterogeneity component are allowed to settle in before all parameters are freed up. The results of estimation are (`child3.out`):

```
Constant     0.1707
            (0.2551)
slope0     -12.6961 ***
            (1.6691)
slope1     -2.8445 ***
            (0.6313)
slope2     -0.2515 ***
            (0.0673)
slope3      0.0902 **
            (0.0385)
dropout     0.2919
            (0.1960)
college    -1.7943 **
            (0.7561)
```

```
boy          0.1880
            (0.1592)
hospital    -0.3817 *
            (0.2255)
SigDelta     0.5887 ***
            (0.2244)

ln-L        -818.74
```

The hazard of mortality decreases sharply between birth and age three months; further decreases until the first birthday; slowly continues to decrease until the tenth birthday; and subsequently increases. Children of better-educated women experience lower mortality risks and boys are at insignificantly elevated risks. The coefficient of primary interest indicates that children that were delivered in hospitals face lower mortality risks (by a factor $\exp(-0.3817) = 0.68$, i.e., by 32 percent), but the effect is not very precisely estimated. There is significant evidence that unobserved mother-specific characteristics affect children's survival chances.

Next consider incorporating the potential endogeneity of hospital delivery due to unmeasured mother attributes which affect both child mortality and the use of a hospital for delivery. It is quite likely that the mothers themselves are aware of at least some of those characteristics. What if they respond to this private knowledge such that those women who are at above-average risk of losing their baby decide to reduce the risks by delivering in a hospital? If that is the case, variable `hospital` is correlated with residual δ, i.e., `hospital` may be endogenous to mortality risk. Correlation between explanatory covariates and residuals leads to biased estimates (e.g., Pindyck and Rubinfeld, 1991). If the only source of correlation is at the mother level, then allowing for the correlation will eliminate the bias. (Additional correlation at the child level would require the introduction of instruments, i.e., child-specific covariates affecting use of a hospital, but not directly affecting child mortality.)

In the example, the same draw of `delta` applies to all children of a particular woman, and many women have multiple children. The variance of mother-specific unmeasured characteristics may thus be determined, and we can account for mother-specific propensities to deliver in a hospital. Women in fragile health (or otherwise aware that they are at above-average risk of losing their children) may at the same time have above-average propensity to deliver in hospitals. Hospital deliveries are thus disproportionately high-risk, but since we know the unobserved risk distribution, we also know the excess portion of high-risk cases that enter hospitals, and can correct accordingly. In short, mother-specific unobservables may be captured by heterogeneity, and any biases they introduce may be eliminated.

The bias due to mother-specific unobservables is eliminated by making the source of the bias part of the model. In the example, the effect of hospital deliveries on mortality may be biased because of non-random hospital delivery decisions. We therefore estimate a joint or multiprocess

model of child survival and the decision to deliver in a hospital. We estimated a hospital delivery model with mother-specific heterogeneity (`hosp2.aml`) in Section 4.1.1 and combine it with converged estimates from "`child3.aml`" in `child4.aml`":

```
option title = "Child mortality w/ endogenous hospital delivery";

dsn=children;

/* Child mortality equation */
define spline Age; intercept; nodes = .25 1 10;
define regressor set AlphaX; var = (educ==1) (educ==3) boy hospital;

/* Hospital delivery equation */
define regressor set BetaX;
   var = 1 log(income) distance (educ==1) (educ==3);

/* Mother-specific correlation across equations */
define normal distribution; dim=2; number of integration points=4;
   name=delta;
   name=eps;

hazard model;
   censor=censor;
   duration = durvar1 durvar2;
   model = durspline(origin=0, ref=Age) +
           regset AlphaX +
           intres(draw=1, ref=delta);

probit model;
   outcome = hospital;
   model = regset BetaX +
      intres(draw=1, ref=eps);

starting values;

Constant    TT     .1707386712
slope0      FT   -12.696140815
slope1      FT   -2.8445023664
slope2      FT    -.2514847318
slope3      FT     .0902330242
dropout     FT     .2919458262
college     FT   -1.7942720568
boy         FT     .1880428793
hospital    TT    -.3816766978
Constant    TT    -1.8868717781
lnIncome    FT     .31390391051
distance    FT    -.04228419725
dropout     FT    -1.1428187936
college     FT     .60158167422
SigDelta    TT     .5886617221
SigmaEps    TT     0.7042604784
Rho         TT      0
;
```

Lines 14-16 are new and specific to multiprocess models. They define a bivariate normal distribution. Lines 23 and 28 use the residuals, `delta` and `eps`, in model specifications in the same way as they were used in single process models. Both are specified with "`draw=1`", i.e., with the same draw. This ensures that they will be correlated.

Residuals are correlated across equations if and only if (a) they were defined as part of the same multivariate distribution and (b) they have the same draw.

As before, the actual draw number is irrelevant. We could have specified "`draw=143`" or "`draw=_id`" or even "`draw=educ`"—all that matters is that they have the same draw number for all equations.

We specified four points of integration for the bivariate distribution. This approximation is taken in both dimensions, so the number of support points is 16. All probit and hazard modules are thus evaluated 16 times, which you will notice in the computing time. Higher-dimensional integrated distributions tend to slow down aML substantially. We therefore recommend that you explore models with relatively few integration points, such as four in each dimension, and specify a higher number for the final run. Since "`child4.aml`" represents our final specification, we re-ran it in "`child5.aml`" with ten support points.

The starting values of regressors are converged values of separate specifications of the two component models in "`hosp2.out`" and "`child3.out`".

When specifying joint models, it tends to be very useful to initialize parameters at the converged values of single equation models.

The only new parameter is the correlation between δ and ε, "`Rho`". We initialize the correlation to zero, so that the initial parameterization of the joint model is the same as the two separate models. (This may be verified by checking the log-likelihoods of the converged separate models and the first iteration of the joint model: -523.65 + -818.74 = -1342.39.)

A zero-mean bivariate normal distribution is fully specified with two standard deviations and a correlation; the standard deviations are initialized first, followed by the correlation. See Section 13.2.6 for trivariate and higher-dimensional distributions.

To compare the estimates in the single equation probit (`hosp2.out`) and hazard (`child3.out`) models to the joint model with four support points (`child4.out`) and with ten points (`child5.out`), we type:

```
mktab  child3 + hosp2  child4  child5
```

	child3+hosp2	child4	child5
Constant	0.1707	0.2727	0.2540
	(0.2551)	(0.2648)	(0.2665)
slope0	-12.6961 ***	-12.5711 ***	-12.6035 ***
	(1.6691)	(1.6877)	(1.6881)
slope1	-2.8445 ***	-2.8735 ***	-2.8509 ***
	(0.6313)	(0.6363)	(0.6373)
slope2	-0.2515 ***	-0.2483 ***	-0.2527 ***
	(0.0673)	(0.0692)	(0.0692)
slope3	0.0902 **	0.0907 **	0.0926 **
	(0.0385)	(0.0394)	(0.0394)
dropout	0.2919	0.2121	0.2277
	(0.1960)	(0.2081)	(0.2098)
college	-1.7943 **	-1.7383 **	-1.7267 **
	(0.7561)	(0.7765)	(0.7771)
boy	0.1880	0.1795	0.1808
	(0.1592)	(0.1594)	(0.1597)
hospital	-0.3817 *	-0.6680 **	-0.6395 **
	(0.2255)	(0.2838)	(0.2878)
Constant	-1.8869 ***	-1.9177 ***	-1.9149 ***
	(0.2453)	(0.2518)	(0.2513)
lnIncome	0.3139 ***	0.3181 ***	0.3175 ***
	(0.0356)	(0.0365)	(0.0365)
distance	-0.0423 **	-0.0432 **	-0.0426 **
	(0.0176)	(0.0186)	(0.0186)
dropout	-1.1428 ***	-1.1389 ***	-1.1392 ***
	(0.1431)	(0.1466)	(0.1466)
college	0.6016 ***	0.6120 **	0.6080 **
	(0.2307)	(0.2382)	(0.2376)
SigDelta	0.5887 ***	0.6376 ***	0.6466 ***
	(0.2244)	(0.2197)	(0.2137)
SigmaEps	0.7043 ***	0.7214 ***	0.7213 ***
	(0.1010)	(0.1053)	(0.1053)
Rho		0.5180 *	0.4698
		(0.2873)	(0.2941)
ln-L	-1342.38	-1340.95	-1340.99

```
NOTE: Asymptotic standard errors in parentheses;
      Significance: '*'=10%;  '**'=5%;  '***'=1%.
```

Compare the separate equations in the first column with joint estimates in the second. We boxed the coefficient of primary interest capturing the effect of hospital deliveries on child

mortality risk. This coefficient increases substantially when we estimate the model jointly. Joint estimation reveals that the beneficial effects are substantially understated if endogeneity of hospital delivery is ignored. Indeed, we find a positive correlation coefficient between unobservables affecting the hospital decision and the mortality risks ($\rho_{\delta\varepsilon} = 0.52$). In other words, women with above-average risks of losing a baby ($\delta > 0$) also tend to have above-average propensities to deliver in a hospital ($\varepsilon > 0$); and vice versa.

None of the other coefficients changes much when we estimate the model jointly. This was to be expected for the hospitalization probit equation, as there is no endogeneity bias issue in this equation. Not much changed in the mortality equation either, but this result does not generalize. In general, endogeneity of a single covariate may bias the coefficient estimates of all covariates in that equation.

Comparing the second and third columns, it did not make much difference whether we approximated the bivariate normal distribution by four or ten points per dimension. The main difference is that the correlation coefficient loses its already marginal significance. Despite its insignificance, the correlation induces a bias in the effect of hospitalization. We recommend to always run a final model with a relatively large number of approximation points.

4.2.2. Classical Simultaneous Equations in Continuous Outcomes

The previous section explained how to estimate simultaneous equations which are recursive (and have multiple replications), i.e., in which the outcome of one equation enters another equation, while the other outcome does not affect the first equation. The current section explains fully simultaneous equations, in which two or more outcomes mutually affect each other. This section illustrates the use of the matrix building block and the elements of its inverse.

The specification of simultaneous models is non-trivial and requires some careful algebra on the user's part prior to model estimation. Consider the following system of two simultaneous continuous equations:

$$\begin{aligned} y_1 &= \gamma_1 y_2 + \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + u_1 \\ y_2 &= \gamma_2 y_1 + \beta_0 + \beta_2 x_2 + \beta_3 x_3 + u_2 \end{aligned}$$

where u_1 and u_2 are jointly normally distributed with standard deviations σ_1 and σ_2 and correlation coefficient ρ_{12}. Running simple regressions with y_2 as explanatory covariate in the equation for y_1, and vice versa, would violate the requirement that explanatory covariates are independent of the error term. Instead, the system may be written as

$$\begin{aligned} y_1 - \gamma_1 y_2 &= \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + u_1 \\ -\gamma_2 y_1 + y_2 &= \beta_0 + \beta_2 x_2 + \beta_3 x_3 + u_2 \end{aligned}$$

or, in matrix notation:

$$\Gamma\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \alpha'X \\ \beta'X \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \text{ where } \Gamma = \begin{pmatrix} 1 & -\gamma_1 \\ -\gamma_2 & 1 \end{pmatrix}.$$

Premultiply both sides by the inverse of Γ:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \Gamma^{-1}\begin{pmatrix} \alpha'X \\ \beta'X \end{pmatrix} + \Gamma^{-1}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

The two equations may be written in reduced form:

$$\begin{aligned} y_1 &= \delta_{11}(\alpha'X) + \delta_{12}(\beta'X) + \delta_{11}u_1 + \delta_{12}u_2 \\ y_2 &= \delta_{21}(\alpha'X) + \delta_{22}(\beta'X) + \delta_{21}u_1 + \delta_{22}u_2 \end{aligned},$$

where $\Gamma^{-1} = \begin{pmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{pmatrix} = \begin{pmatrix} 1 & -\gamma_1 \\ -\gamma_2 & 1 \end{pmatrix}^{-1} = \dfrac{1}{1-\gamma_1\gamma_2}\begin{pmatrix} 1 & \gamma_1 \\ \gamma_2 & 1 \end{pmatrix}$, and therefore

$$\begin{cases} \delta_{11} = 1/(1-\gamma_1\gamma_2) \\ \delta_{21} = \gamma_2/(1-\gamma_1\gamma_2) \\ \delta_{12} = \gamma_1/(1-\gamma_1\gamma_2) \\ \delta_{22} = 1/(1-\gamma_1\gamma_2) \end{cases}$$

The solution offered by aML involves the definition of a matrix Γ and the direct use of elements of the inverse matrix Γ^{-1} in model specifications. In other words, by defining matrix Γ, you have direct access to δ_{11}, δ_{12}, δ_{21}, and δ_{22}, with all implied restrictions on Γ automatically imposed.

Control file "`Samples\Chapter4\simul.aml`" shows how to set up the problem.

```
 1  dsn = simul.dat;
 2
 3  define regset AlphaX; var = 1 x1 x2;
 4  define regset BetaX;  var = 1    x2 x3;
 5
 6  define normal distribution; dim=2;
 7     name=u1;
 8     name=u2;
 9
10  define matrix Gamma; dim=(2,2);
11
12  continuous model;
13     outcome = y1;
14     model = par inv(Gamma(1,1))*regset AlphaX +
15             par inv(Gamma(1,2))*regset BetaX +
16             par inv(Gamma(1,1))*res(draw=1, ref=u1) +
17             par inv(Gamma(1,2))*res(draw=1, ref=u2);
18
19  continuous model;
```

```
 20     outcome = y2;
 21     model = par inv(Gamma(2,1))*regset AlphaX +
 22             par inv(Gamma(2,2))*regset BetaX +
 23             par inv(Gamma(2,1))*res(draw=1, ref=u1) +
 24             par inv(Gamma(2,2))*res(draw=1, ref=u2);
 25
 26  starting values;
 27
 28  Alpha0     TT    1.119339
 29  Alpha1     TT    .5028963
 30  Alpha2     TT   -.2359311
 31  Beta0      TT   -3.392511
 32  Beta2      TT    2.725859
 33  Beta3      TT    .1036613
 34  sigma_u1   TT    .46658
 35  sigma_u2   TT    1.4816
 36  rho_u1u2   FT     0
 37  One        FF     1          /*  not estimated  */
 38  MinGam2    TT     0
 39  MinGam1    TT     0
 40  One        FF     1          /*  not estimated  */
 41  ;
```

Line 10 defines matrix Γ. Its elements appear columnwise in the starting values, i.e., the elements of a matrix `A` with dimension (m,n) are initialized as follows: `A(1,1)`, `A(2,1)`, ..., `A(m,1)`, `A(1,2)`, ..., `A(m,2)`, ..., `A(1,n)`, ..., `A(m,n)`. Also see Section 13.2.5. Note the names assigned to parameters in the starting values: $\Gamma(1,1)$ and $\Gamma(2,2)$ are always unity, so we named them `One` and fixed their values to 1; $\Gamma(1,2)$ corresponds to $-\gamma_1$ and is thus named `MinGam1`; $\Gamma(2,1)$ corresponds to $-\gamma_2$ and thus named `MinGam2`. aML thus estimates matrix elements $-\gamma_1$ and $-\gamma_2$, not γ_1 and γ_2 directly. You need to reverse the signs of the parameter estimates to get γ_1 and γ_2 themselves.

Elements of the inverse of `Gamma` are directly used in the model statements (lines 14–17 and 21–24), in the same way in which parameters are used. For example, `inv(Gamma(1,1))` is δ_{11}, above; `inv(Gamma(1,2))` corresponds to δ_{12}, et cetera. It is important to realize that, for example, `inv(Gamma(1,1))` is element (1,1) of the inverse of `Gamma`; it is not the inverse of element (1,1) of `Gamma`. However, aML estimates structural parameters (`MinGam1` and `MinGam2`) directly.

Note the starting values. Prior to specifying this model, we ran simple OLS regressions of `x1` and `x2` on `y1` and of `x2` and `x3` on `y2` (not shown). The estimates are used as starting values here, with the coefficients of `y1` and `y2` cross-effects initialized at zero. The standard deviations of `u1` and `u2` are initialized at the root mean square errors of the OLS regressions.

We estimate the model in two rounds. In fully simultaneous equations models, the parameter search tends to be sensitive to good starting values, and searches may go more smoothly if

parameters are freed up stepwise. Attempts to estimate too many structural parameters at once may fail due to poor starting values. In this case, the model converged without difficulty.

5. Advanced Topics

The preceding sections explained aML's basic features. Many of these features may be combined to specify advanced models. This section describes just a subset of the more advanced models that aML supports.

5.1. Advanced Probit and Logit Models

The standard probit and logit models are given by:

$$p^* = \beta' x + u, \quad p = \begin{cases} 0 & \text{if } p^* \le 0; \\ 1 & \text{if } p^* > 0, \end{cases}$$

where p^* is a latent propensity and p an observed indicator variable whose value depends on whether p^* is above or below a zero threshold. If $u \sim N(0,1)$, this is the probit model; if u follows the logistic distribution, it is the logit model. This section illustrates how to specify other thresholds and, for the probit model only, how to specify residuals with a non-unit variance, i.e., $u \sim N(0,\sigma_u^2)$. Section 5.2 explains ordered probit and logit models.

5.1.1. Probit or Logit with Nonzero Threshold

Probit and logit models with a non-zero threshold τ are given by:

$$p^* = \beta' x + u, \quad p = \begin{cases} 0 & \text{if } p^* \le \tau; \\ 1 & \text{if } p^* > \tau. \end{cases}$$

The likelihood function is:

$$L = \begin{cases} F(\tau - \beta' x) & \text{if } p = 0; \\ 1 - F(\tau - \beta' x) & \text{if } p = 1, \end{cases}$$

where $F(u) = \Phi(u)$ for the probit (with unit variance) and $F(u) = (1 + \exp(-u))^{-1}$ for the logit.

If no threshold is specified, aML assumes a threshold at zero. Non-zero thresholds may be specified (and estimated) by using a parameter. File "`Samples\Chapter5\tau.aml`" re-estimates the high school graduation model of Section 2.1 (`Samples\Chapter2\educ1.aml`) with a threshold:

```
 1  option title = "As Samples\Chapter2\educ1, but with threshold";
 2  dsn = ..\Chapter2\education.dat;
 3
 4  define parameter Tau;
 5  define regressor set BetaX;
 6     var = female birth18 dadltHS dadcoll momltHS momcoll poorkid;
 7
 8  probit model;
 9     outcome = HSgrad;
10     threshold = Tau;
11     model = regset BetaX;
```

```
12
13  starting values;
14
15  Tau          T    0
16  female       T    0
17  birth18      T    0
18  dadltHS      T    0
19  dadcoll      T    0
20  momltHS      T    0
21  momcoll      T    0
22  poorkid      T    0
23  ;
```

Line 3 defines the threshold parameter; line 10 uses it in the probit model specification. The only other difference with the earlier model is that we suppressed the intercept. As is readily seen from the likelihood function above, the threshold and intercept and opposites and perfectly collinear.

If both a threshold and an intercept are present in a probit or logit model, they are perfectly collinear and cannot both be estimated without additional restrictions or additional specifications that identify one or the other.

The following table compares the results of Section 2.1's "`educ1.out`" and "`tau.out`":

	educ1	tau
Constant	2.1690 ***	
	(0.2648)	
Tau		-2.1690 ***
		(0.2648)
female	0.0595	0.0595
	(0.2036)	(0.2036)
birth18	-1.5982 ***	-1.5982 ***
	(0.2693)	(0.2693)
dadltHS	-0.9271 ***	-0.9271 ***
	(0.2071)	(0.2071)
dadcoll	0.3035	0.3035
	(0.3715)	(0.3715)
momltHS	-0.4303 **	-0.4303 **
	(0.1951)	(0.1951)
momcoll	0.5984	0.5984
	(0.5169)	(0.5169)
poorkid	-1.0371 ***	-1.0371 ***
	(0.1992)	(0.1992)

	educ1	tau
ln-L	-147.35	-147.35

Note that the threshold in `tau` is equal to the opposite of the intercept in `educ1`. Otherwise, all coefficients and the log-likelihoods are identical.

The syntax and approach is identical for logit models with non-zero threshold.

5.1.2. Probit with Non-Unit Standard Deviation

The standard probit model assumes a standard normally distributed residual, $u \sim N(0,1)$. aML supports both this standard probit model and its extension with non-unit variance (and non-zero threshold):

$$p^* = \beta' x + u, \quad p = \begin{cases} 0 & \text{if } p^* \le \tau; \\ 1 & \text{if } p^* > \tau, \end{cases}$$

where $u \sim N\left(0, \sigma_u^2\right)$. The likelihood function is:

$$L = \begin{cases} \Phi\left(\dfrac{\tau - \beta' x}{\sigma_u}\right) & \text{if } p = 0; \\ 1 - F\left(\dfrac{\tau - \beta' x}{\sigma_u}\right) & \text{if } p = 1. \end{cases}$$

If the aML model specification does not contain a non-integrated residual, aML inserts an *iid* $N(0,1)$ residual. Consider the following probit model specification:

```
probit model;
   outcome = varname;
   model = regset regsetname;
```

No non-integrated residual is specified, so aML inserts an *iid* $N(0,1)$ residual and makes this explicit when it repeats the model specification in the output file:

```
probit model;
   outcome = varname;
   model = regset regsetname +
      res(draw=_iid, ref=N(0,1))
      ;
```

The residual is drawn independently for every replication of the outcome (every equation). You may explicitly specify "`res(draw=_iid, ref=N(0,1))`" in the control file; the "`N(0,1)`" is not a residual name, but is understood to be a standard normal residual. It may only be combined with "`draw=_iid`", not with any explicit draw variable or expression.

The default assumption is not made if the model specification explicitly contains a non-integrated residual specification. Specifying an integrated residual, such as when introducing heterogeneity, leaves the implicit $iid\,N(0,1)$ residual assumption intact. Only a non-integrated residual automatically replaces the default term. For example:

```
define normal distribution; dim=1; name=u;

probit model;
   outcome = varname;
   model = regset regsetname +
      res(draw=_iid, ref=u);
```

Now the residual is "u", and its standard deviation may be estimated. No implicit residual is assumed.

Probit outcomes are 0 or 1 (false or true, off or on). The only thing that matters about these outcomes is that there are two distinct values; conceptually, their actual values are without meaning. As a result, it is not possible to separately identify thresholds τ and parameters β on the one hand and the standard deviation of the residual σ_u on the other. Mathematically, this may be seen from the likelihood function: it may be written as a function of τ/σ_u and β/σ_u. The usual solution is, of course, to normalize $\sigma_u = 1$, but σ_u may be estimated if it is identified by some other part of the overall model. For example, σ_u may be identified because it also appears in a continuous normal density model, as in a Tobit model (Section 2.9).

Logit residuals are always distributed according to the standard logistic distribution. aML does not permit non-standard logistic residuals. It always inserts a logistic residual in logit models.

5.2. Ordered Probit and Logit Models With Known Thresholds

Ordered probit/logit models are probit/logit models with multiple categorical outcomes that are ordered. See Section 2.8 for an introduction. In the most common case, the ordered probit/logit thresholds are the same for all observations but unknown, and need to be estimated. This was the example of Section 2.8. Alternatively, there may be cases with known thresholds that may differ across observations. This is the subject of this section. We discuss ordered probit models only. While all carries over to ordered logit models, the ordered probit is far more appealing for data with known thresholds (see page 162).

Consider a survey question that asks for range responses. For example, a survey may ask about income: is it less than $10,000 per year, between $10,000 and $40,000, between $40,000 and $100,000, or more than $100,000? Or a survey may ask about the commuting distance to school or work: is it less than or equal to 9 miles, 10-19 miles; 20-39 miles; or 40 miles or more? We take the latter example as illustration.

The (SAS, Stata, SPSS) data contain a categorical response for one of the four categories. We convert these into lower and upper bound variables to represent the thresholds. For example, an individual who indicated that his commute is between 11 and 20 miles, we code `lower=11` and `upper=20`. We need a way to represent infinity for individuals who report that their commute exceeds 40 miles: `lower=40` and `upper=999`. The value chosen for infinity is largely arbitrary; see below. The data are converted into aML-formatted "`Samples\Chapter5\commute.dat`". As indicated by its documentation file, "`commute.sum`", it includes information on the reported commuting distance (`lower` and `upper`), the respondent's age (`age`), marital status (`married`), whether he or she has any children (`children`), whether he or she is a student (`student`), and whether he or she owns a house (`ownhome`). There is only one record per individual.

Our model to relate commuting distance to demographic characteristics is:

$$y = \beta' x + v \,,$$

where y is the commuting distance, x denote covariates, and $v \sim N(0, \sigma_v^2)$ captures residual variation. If exact distances were observed, we would model this with a continuous model. Instead, we only know y's range. The likelihood function for an outcome that lies between lower bound τ_L and upper bound τ_U is:

$$L = \Phi\left(\frac{\tau_U - \beta' x}{\sigma_v}\right) - \Phi\left(\frac{\tau_L - \beta' x}{\sigma_v}\right).$$

The following control file estimates the above ordered probit model (normal interval model) of commuting distance (`commute.aml`):

```
1  option title = "Range responses on commuting distance";
2
```

```
 3  dsn = commute;
 4
 5  define regressor set BetaX;
 6     var = 1 age age^2 married children student ownhome;
 7
 8  define normal distribution; dim=1; name=v;
 9
10  ordered probit model;
11     threshold vars = lower(-Inf=-999) upper(Inf=999);
12     model = regset BetaX +
13             res(draw=_iid, ref=v);
14
15  starting values;
16
17  Constant    TT     20
18  age         FT     0
19  agesq       FT     0
20  married     FT     0
21  children    FT     0
22  student     FT     0
23  ownhome     FT     0
24  SigmaV      TT     10
25  ;
```

Lines 5 and 6 define the explanatory covariates. In addition to an intercept, we control for age, age-squared, marital status, children, student status, and home ownership.

Lines 10-13 specify the ordered probit model. The outcome specification is new:

```
threshold vars = lower(-Inf=-999) upper(Inf=999);
```

The defining characteristic of the model we illustrate is that the thresholds are known and included in the data as variables. The outcome is therefore specified in the form of threshold variables. The syntax is:

```
threshold vars = varname1 (-Inf=n1) varname2 (Inf=n2);
```

where *varname1* and *varname2* are data variables (or expressions) representing the lower and upper bounds of the range response, respectively. In the notation from above, *varname1*= τ_L and *varname2*= τ_U. Open-ended intervals involve a lower bound of $\tau_L = -\infty$ or an upper bound of $\tau_U = \infty$. You may choose any two "special values" to represent plus and minus infinity and must tell aML which special values you selected. As explained above, we opted for *n2*=999 to represent plus infinity. We did not have to choose anything for minus infinity, because commuting distance is never negative. Even though there is no minus infinity in the current application, we need to give aML a value for minus infinity. We specified "-Inf=-999"; it is without relevance. (Even though the lowest lower bound is zero, we must not specify "-Inf=0", as this would imply that zero is the same as minus infinity. In the illustration, the lowest lower bound truly is zero, not minus infinity.)

Note that there is no "outcome" or "outcomes" specification: the outcome is specified by the "threshold vars" statement. This syntax tells aML that the data variables contain thresholds, and not integer numbers corresponding to threshold parameters that delimit a categorical outcome, as is the case in ordered probit models with unknown thresholds (Section 2.8).

The starting values are chosen in accordance with the guidelines provided in Section 6.5. File "commute.out" contains the output:

```
 1  ============================================================================
 2  =                Range responses on commuting distance                     =
 3  ============================================================================

    et cetera...

50  ordered probit model;
51     threshold vars = lower (-Inf=-999) upper (Inf=999);
52     model = regset BetaX +
53        res(draw=_iid, ref=v)
54        ;
55
56     Summary statistics of the outcome and selected variables:
57
58        outcome1 |       Freq.     Percent
59     ------------+------------------------
60                0 |        150       25.95
61               10 |        277       47.92
62               20 |        104       17.99
63               40 |         47        8.13
64     ------------+------------------------
65            Total |        578      100.00
66
67        outcome2 |       Freq.     Percent
68     ------------+------------------------
69                9 |        150       25.95
70               19 |        277       47.92
71               39 |        104       17.99
72              999 |         47        8.13
73     ------------+------------------------
74            Total |        578      100.00

    et cetera...
```

Note the tabulations of "outcome1" and "outcome2", corresponding to variables lower and upper. They reflect the distribution of range responses in the data.

Further Considerations

In the illustration, we chose mutually exclusive categories (0-9, 10-19, 20-39, and ≥ 40), corresponding to the way the question was asked. Suppose the question had been: "How far do you live from school or work? Is it less than 10 miles, 10 miles or more but less than 20 miles; 20

miles or more but less than 40 miles; or 40 miles or more?" We would code the threshold variables correspondingly. For example, someone who indicates living "10 miles or more but less than 20 miles" from his work would be coded `lower=10` and `upper=20`. In other words, lower and upper bounds of adjacent categories may be the same value, if the question is worded accordingly.

Taking this issue one step further, there is no reason for the categories to be mutually exclusive. For example, the survey may first ask "Is it more or less than 20 miles?," and then ask for a finer categorization around 10 or 40 miles, depending on whether the distance is above or below 10 miles. The respondent may, for example, indicate that he lives closer than 20 miles from work, but not know the answer to the follow-up question. In that case, we simply code `lower=0` and `upper=20`, and use the information in the model. Any pair of thresholds may be processed.

Consider an extension. Suppose the survey asks for one's income. Some respondents provide an exact answer; others refuse or indicate that they do not know the answer. If no exact income figure is given, the survey launches into range questions ("unfolding brackets" in parlance of the Health and Retirement Study). Both types of answers may be used in an analysis of income, without resorting to imputations. The idea is to let aML switch between a continuous model (if an exact figure is available) and an ordered probit model (for range responses). This example is worked out in Section 13.8.2.

The above discussion of ordered models with known extensions focuses on ordered probit models. In principle, ordered logit models are analogous. However, the (implied) residual of an ordered logit model has a given standard deviation that cannot be estimated. This is typically undesirable when thresholds are given by data. For example, when the metric of the outcome changes (commuting distance in kilometers rather than miles), the parameter estimates would change. Further, an ordered logit model would be very sensitive to outliers, since any threshold outside the (-5, 5) range, or so, leads to a likelihood that is zero or one.

We recommend that you represent infinity by 999, 9999, 99999, 999999, 9999999, or other large integer values. Values in excess of 10 million and non-integer values may cause undesired results because of numerical imprecision. Your data values namely pass through ASCII representation before being processed by raw2aml, and if the ASCII representation is not the same as the value you specify in the control file, it may not be recognized as a representation of infinity.

The example illustrated how to model normal interval data in a single level setting. Needless to say, the above extends naturally into multilevel models.

5.3. Truncated Normal Regression Model

Truncated normal density models apply if nothing is present in the data on cases whose continuous normally distributed variable is outside some range. For example, suppose sample contains only individuals with strictly positive hours worked. Unlike the Tobit model (Section 2.9), nothing is known about individuals with zero hours. The model is:

$$h^* = \beta' x + v, \quad h = \begin{cases} h^* \text{ if } h^* > \tau; \\ \text{not observed if } h^* \le \tau, \end{cases}$$

where h^* is some latent continuous concept related to the partially observed continuous variable h. The observed distribution is therefore truncated, and the likelihood function is:

$$L = \frac{1}{\sigma_v}\phi\left(\frac{h-\beta' x}{\sigma_v}\right) \Bigg/ \left(1-\Phi\left(\frac{\tau-\beta' x}{\sigma_v}\right)\right),$$

where $\phi(\cdot)$ denotes the normal density function and $\Phi(\cdot)$ the cumulative normal density function. The denominator is the probability that h is observed, i.e., $P(h^* > \tau)$. The likelihood thus has a normal density in the numerator, similar to a continuous model, and a probit probability in the denominator. The model is specified using the "`numerator`" and "`denominator`" options to model statements:

```
define parameter Tau;
define regressor set XBeta; var = ...;
define normal distribution; dim=1; name=u;

continuous model;
numerator;   /* may be omitted, as it is the default */
outcome = hours;
model = regset XBeta + res(draw=_iid, ref=u);

probit model;
denominator;
threshold = Tau;
outcome = (hours==hours);  /* always evaluates to 1 */
model = regset XBeta + res(draw=_iid, ref=u);
```

The `denominator` statement is new; it ensures that the likelihood of this module is in the denominator of the overall likelihood, i.e., that its log-likelihood is subtracted from the overall log-likelihood. Several related models may be specified in a similar way. A right-truncated normal density model, in which h and x are only observed if h is *less* than some threshold, may be specified using a known threshold probit model statement in which the outcome always evaluates

to zero, or equivalently by an ordered probit model with zero thresholds in the data. A model in which the continuous outcome is both left- and right-truncated requires an ordered probit model with two thresholds and an outcome that lies between them. If the threshold is not constant, but varies across observations, an ordered probit model with threshold variables applies.

Using the work data of Section 2.9, we may drop individuals who do not work and estimate a truncated normal model of hours worked. The model statement is (`truncate.aml`):

```
10  /* positive hours observations */
11  continuous model;  keep if (hours>0);
12     outcome = hours;
13     model = regset BetaX +
14             res(draw=_iid, ref=v);
15
16  /* Probability of being in the sample */
17  probit model;  keep if (hours>0);
18     denominator;
19     outcome = (hours==hours);  /* always evaluates to one */
20     model = regset BetaX +
21             res(draw=_iid, ref=v);
```

The keep statements ensure that the model only applies to individuals that worked positive hours (`hours>0`). File "`truncate.out`" contains the results of estimation.

5.4. Heckman Selection Model

Heckman (1979) proposed a model in which a continuous outcome is only observed on the basis of an auxiliary selection equation. For example, we only observe wage rates for individuals that decided to work. Individuals that decided not to work are also in the data, but their wage rate is missing, or zero, or otherwise irrelevant. Formally, the model consists of a selection equation:

$$z^* = \alpha' x + u, \; z = \begin{cases} 0 & \text{if } z^* \le 0; \\ 1 & \text{if } z^* > 0, \end{cases}$$

and an equation for the outcome of substantive interest:

$$y = \beta' x + v \,,$$

where y is observed if and only if $z = 1$. If all selection operates through observed covariates, the second equation may be estimated using only the complete data. However, if there is correlation between u and v, estimation using only the complete data yields biased coefficient estimates. We assume

$$\begin{pmatrix} u \\ v \end{pmatrix} \sim N\left(0, \begin{pmatrix} 1 & \\ \rho\sigma_v & \sigma_v^2 \end{pmatrix}\right).$$

The model may be consistently estimated by the two-stage procedure proposed by Heckman (1979) or by full information maximum likelihood. aML only supports the latter method.

We illustrate the selection model using "`Samples\Chapter2\work.dat`" data introduced in Section 2.9. The data contain labor force participation and wage information on 1,126 individuals: 342 who did not work (`hours=0`) and 784 who did work (`hours>0`). The participation decision is a function of education and the number of young children that the respondent has at home; the wage rate, conditional on participation, is determined by education, age, and tenure on the job. There is only one record per person.

The model may be specified as (`heckman.aml`):

```
option title = "Wages with selective labor force participation";

dsn = work;

define regressor set AlphaX;
   var = 1 (educ==1) (educ==3) children;

define regressor set BetaX;
   var = 1 (educ==1) (educ==3) spline(age, 30 50) tenure;

define normal distribution; dim=2;
   name=u;
   name=v;

```

```
15  probit model;
16     outcome = (hours>0);
17     model = regset AlphaX +
18             res(draw=1, ref=u);
19
20  continuous model;  keep if (hours>0);
21     outcome = wage;
22     model = regset BetaX +
23             res(draw=1, ref=v);
24
25  starting values;
26
27  Constant     T     1.5684158103
28  dropout      T    -1.1304467365
29  college      T     1.1375451757
30  children     T    -.60561246295
31  Constant     T      24494.40052
32  dropout      T    -17137.041024
33  college      T     27747.629488
34  age<30       T     350.13601824
35  age30-50     T     1724.8176594
36  age>50       T    -1318.4817886
37  tenure       T      1296.061371
38  SigmaU       F     1
39  SigmaV       T     22829.395063
40  Rho          T     0
41  ;
```

Regressor set `AlphaX` contains variables that may affect the decision to work, `BetaX` variables that affect the wage rate. Lines 11-13 define a bivariate distribution for residuals *u* and *v*. Lines 15-18 specify the selection equation; its outcome is whether the person worked (`hours>0`). Lines 20-23 the model for the selectively observed continuous outcome; its keep statement (`hours>0`) makes sure that we only include individuals who worked. The starting values are converged values of two separate runs, not shown here, for the selection equation (`work.aml`) and the wage equation (`wage0.aml`).

For multiprocess models that consist of two or more equations, first estimate those equations separately, i.e., without correlation across the equations. Initialize the parameters of the combined model to converged values of separate models, and start correlations at zero.

The selection equation is a probit equation which requires that its residual has unit variance. We tend to omit such a residual from probit equations, because aML adds it by default. However, in the current application we must explicitly specify it, because it needs to be correlated with *v* in the continuous outcome equation. In both the selection and the continuous equations we specified residuals with "`draw=1`". Residuals `u` and `v` were (1) defined as part of the same distribution and (2) used with the same draw, and are therefore correlated across the equations.

File "heckman.out" contains the results of estimation. The correlation between u and v is estimated at 0.4212 (not shown here) and strongly significant, indicating that respondents who work are a non-random subset of the population.

Further Considerations

We illustrated the Heckman selection model in its traditional specification as a single-level normal density (continuous) model with a probit selection equation. Naturally, it may be extended to multilevel settings, in which there are multiple selection switches and multiple continuous outcomes. The specification is analogous to the illustration above, but with residual draws that differ corresponding to lower-level independence of outcomes.

The model may furthermore be extended to other, non-continuous types of outcomes that are based on the normal distribution. For example, suppose the survey did not ask for respondents' exact wage rates, but instead asked for range responses only: "Is your wage below $1,000, between $1,000 and $2,500, between $2,500 and $5,000, or above $5,000 per month?" These interval responses may be analyzed using an ordered probit model with known thresholds (Section 5.2). Note the close association of ordered probit ("normal interval") models and continuous ("normal density") models: both consist of a regression and a normally distributed residual. The ordered probit residual may be correlated with the residual in a probit selection equation, just like illustrated above for the residual in a continuous outcome model. We are not aware of applications in the literature where the Heckman selection model is generalized to ordered probit (normal interval) outcomes. However, it illustrates neatly how basic constructs may be combined in aML to lead to powerful models.

5.5. Heteroskedasticity

Heteroskedasticity is present if the variance of residuals varies across observations or across outcomes within observations. Consider error term ε_i, where *i* denotes the observation number or (sub)branch within an observation. Its variance, $\mathrm{Var}(\varepsilon_i) = E(\varepsilon_i^2) = \sigma_i^2$, may not be constant, but related to characteristics of the observation or (sub)branch.

Heteroskedasticity may be captured by aML in several ways, depending on the assumed parameterization. We illustrate several cases using an example of wage earnings in two countries. In both countries, a 1997 survey asked about individual respondents' wage earnings in earlier years. Some respondents only provided earnings for 1996; others for multiple years, sometimes as far back as 1978. Consider an equation explaining the log-earnings of person *i* at time *t*:

$$\ln Y_{it} = \beta' X_{it} + \varepsilon_i + v_{it} .$$

The unit of observation is a person; there are up to 10 wage records per person. In deviation from the convention used throughout most of this manual, we explicitly write the person subscript *i*. Explanatory covariates X_{it} include a country indicator, education, and age. Residual ε_i reflects differences in earnings due to person-specific heterogeneity, which does not vary over time; v_{it} captures transitory variation. We would like to test for heteroskedasticity in ε_i across individuals in the two countries and for recall bias, i.e., for heteroskedasticity in v_{it} as a function of the recall period. The data may be found in "`Samples\Chapter5\hskedas.dat`". The documentation file contains (`hskedas.sum`):

```
 1  Documentation for 'hskedas.dat'
 2  Created on Sun Mar 12 14:50:36 2000 with raw2aml version 1.00.
 3  Ascii data set: 'hskedas.raw'
 4
 5  Number of observations:    1000
 6  Maximum number of level 2 branches in any observation:     10
 7
 8  ------------------------------------------------------------
 9
10  LEVEL 1 VARIABLES:
11  Variable     N       Mean    Std Dev        Min        Max
12  _id       1000      500.5   288.8194        1.0     1000.0
13  country   1000      1.347   .4762539        1.0        2.0
14  educ      1000      2.152   .6643322        1.0        3.0
15
16  LEVEL 2 VARIABLES:
17  Variable     N       Mean    Std Dev        Min        Max
18  year      5664   1989.921    4.88628     1978.0     1996.0
19  age       5664   47.89936   12.45268       16.0       73.0
20  income    5664   20003.35   353278.5        1.0   2.06E+07
21
22  ------------------------------------------------------------
```

```
23
24  NOTE: there is variation in all data variables.
```

Level 1 corresponds to a respondent, level 2 to years for which the respondent provided wage earnings. Variable `country` takes value 1 for the first country and 2 for the second; `educ` is 1 for high school drop-outs, 2 for high school graduates, and 3 for college graduates. Variable `year` indicates the year to which earnings relate; `age` is the age of the respondent at the end of that year; and `income` denotes wage earnings, converted into a common currency.

Heteroskedasticity across groups

We want to test for heteroskedasticity in ε_i across countries, and parameterize:

$$\mathrm{Var}(\varepsilon_i) = \sigma_i = \begin{cases} \sigma_1^2 & \text{if the respondent is from Country 1;} \\ \sigma_2^2 & \text{if the respondent is from Country 2.} \end{cases}$$

We approach this by defining separate distributions to capture heterogeneity for respondents in Country 1 and Country 2. The aML control file is "`hskedas1.aml`":

```
 1  option title = "Heteroscedasticity in heterogeneity";
 2  dsn = hskedas;
 3  option save step;  /* often a good idea with continuous models */
 4
 5  define regressor set XBeta;
 6     var = 1 (country==2) (educ==1) (educ==3) spline(age, 30 55);
 7
 8  define normal distribution; dim=1; name=eps1;  /* for country 1 */
 9  define normal distribution; dim=1; name=eps2;  /* for country 2 */
10  define normal distribution; dim=1; name=v;
11
12  continuous model;  keep if (country==1);
13     outcome = log(income);
14     model = regset XBeta +
15        res(draw=1, ref=eps1) +
16        res(draw=_iid, ref=v);
17
18  continuous model;  keep if (country==2);
19     outcome = log(income);
20     model = regset XBeta +
21        res(draw=1, ref=eps2) +
22        res(draw=_iid, ref=v);
23
24  starting values;
25
26  Constant    T     2.7981373906
27  country2    T     .41548399208
28  dropout     T     -0.390239703
29  college     T     .65124745201
30  age<30      T     .07153197354
31  age30-55    T     .03429744567
32  age55+      T    -.01784057663
```

```
33  sig_eps1    T     1.0666608487
34  sig_eps2    T     1.0666608487
35  sig_v       T     2.2369666396
36  ;
```

Lines 8 and 9 define two distributions, one for each country. Accordingly, the model specification is split up by country to select the appropriate ε_i. The starting values follow from a specification without heteroskedasticity (`hskedas0`, not shown here). We may alternatively (and equivalently) parameterize the heteroskedasticity structure as follows:

$$\text{Std}(\varepsilon_i) = \sigma_i = \begin{cases} \sigma_\varepsilon & \text{if the respondent is from Country 1;} \\ \lambda\sigma_\varepsilon & \text{if the respondent is from Country 2.} \end{cases}$$

In other words, residual ε_i is scaled by load factor λ, which may be specified as a parameter and estimated directly (`hskedas2.aml`):

```
 1  option title = "Heteroscedasticity in heterogeneity";
 2  dsn = hskedas;
 3
 4  define regressor set XBeta;
 5     var = 1 (country==2) (educ==1) (educ==3) spline(age, 30 55);
 6
 7  define parameter Lambda;   /* Load factor on eps for Country 2 */
 8  define normal distribution; dim=1; name=eps;
 9  define normal distribution; dim=1; name=v;
10
11  continuous model;  keep if (country==1);
12     outcome = log(income);
13     model = regset XBeta +
14        res(draw=1, ref=eps) +
15        res(draw=_iid, ref=v);
16
17  continuous model;  keep if (country==2);
18     outcome = log(income);
19     model = regset XBeta +
20        par Lambda * res(draw=1, ref=eps) +
21        res(draw=_iid, ref=v);
22
23  starting values;
24
25  Constant    T     2.7981373906
26  country2    T     .41548399208
27  dropout     T     -0.390239703
28  college     T     .65124745201
29  age<30      T     .07153197354
30  age30-55    T     .03429744567
31  age55+      T    -.01784057663
32  Lambda      T     1
33  sig_eps     T     1.0666608487
34  sig_v       T     2.2369666396
35  ;
```

Line 7 defines λ; line 20 interacts it with ε. Interactions between parameters (or regressor sets) and residuals offer a natural, flexible, and intuitive way of specifying heteroskedasticity. The residual structure estimates are:

	hskedas0	hskedas1	hskedas2
sig_eps	1.0667 ***		0.9217 ***
	(0.0400)		(0.0517)
sig_eps1		0.9216 ***	
		(0.0517)	
sig_eps2		1.3034 ***	
		(0.0730)	
Lambda			1.4174 ***
			(0.1133)
sig_v	2.2370 ***	2.2375 ***	2.2372 ***
	(0.0193)	(0.0193)	(0.0193)
ln-L	-12990.05	-12980.80	-12980.79

The first column ignored heteroskedasticity; the second and third are defined above and equivalent to each other. Note that, apart from minor rounding errors, the estimated $\sigma_2 = 1.3034$ in the second column is equal to $\lambda\sigma_\varepsilon = 1.4174 * 0.9217 = 1.3064$ in the third column. (Stricter convergence criteria would reduce the difference.) The null hypothesis of homoskedasticity is rejected by the fact that λ is significantly different from one, or by a likelihood ratio test. (A likelihood ratio test on the first two columns computes the probability that a χ^2 with one degree of freedom exceeds twice the difference in log-likelihoods. As readily found by auxiliary program amltest, that probability is .00001694, i.e., homoskedasticity is strongly rejected. See Section 15.3 for amltest.)

Indirect Referencing

The heteroskedasticity specification of "`hskedas1.aml`" may be stated more compactly using so-called indirect referencing. We only briefly illustrate the concept here. For more details see Sections 5.11 and 13.3.4.

aML control files contain definitions of model building blocks and model specifications which refer back to the previously defined building blocks. The building blocks are typically given a name (`XBeta`, `Lambda`, `eps1`, `eps2`), and the references are typically by name:

```
regset XBeta
par Lambda
res(draw=1, ref=eps1)
```

In aML terminology, this is "direct referencing." aML also offers "indirect referencing" through one or more variables in the data. First, you need to assign so-called reference numbers to a

building block definition. Second, in the model statement, a reference variable or expression is specified whose value is matched to a defined building block. For example, the model in "hskedas2.aml" could have been specified as follows:

```
define normal distribution; dim=1; name=eps1; ref=1;
define normal distribution; dim=1; name=eps2; ref=2;

continuous model;
   outcome = ...;
   model = res(draw=1, refvar=country) + ...;
```

We assigned reference numbers 1 to eps1 and 2 to eps2. The residuals may now be directly referenced, by name, or indirectly, by reference variable (refvar). The models for the two countries are combined into one statement, and variable country is used as the reference variable. If country=1, aML figures out that eps1 applies; if country=2, eps2 applies. For more details see Sections 5.11 and 13.3.4.

Heteroskedasticity as a function of covariates

In a second example of heteroskedasticity, we wish to test for heteroskedasticity in v_{it} as a function of covariates. We suspect that the reported wage earnings are subject to recall bias, i.e., less precise for years long before the survey than for more recent years. Variables survey and year indicate the survey year and the year to which reported earnings pertain, respectively, so we may compute the recall period τ as survey-year. We parameterize:

$$v_{it} = (1+\alpha\tau)u_{it}\,.$$

If $\alpha > 0$, the variance of the transitory residual increases with recall period.[24] Note that $(1+\alpha\tau)$ is readily defined as a regressor set. The following specifies this heteroskedasticity parameterization by interacting a regressor set and a residual (hskedas3.aml):

```
1  option title = "Heteroscedasticity due to recall error";
2  dsn = hskedas;
3
4  define regressor set XBeta;
5     var = 1 (country==2) (educ==1) (educ==3) spline(age, 30 55);
6
7  define normal distribution; dim=1; name=eps;
8
```

[24] Should α be negative and $(1+\alpha\tau) < 0$, the resulting residual v_{it} has the opposite sign from u_{it}. Since the residual is normally distributed with zero mean, and thus symmetric around zero, the sign of v_{it} is inconsequential. In other words, we effectively parameterize $v_{it} = |1+\alpha\tau|u_{it}$. If u_{it} were defined as part of a multivariate distribution and $(1+\alpha\tau) < 0$, its correlation coefficient(s) would switch signs.

```
 9  /* Survey was held in 1997 */
10  define regressor set Recall; var = 1 (1997-year);
11  define normal distribution; dim=1; name=u;
12
13  continuous model;
14     outcome = log(income);
15     model = regset XBeta +
16        res(draw=1, ref=eps) +
17        regset Recall * res(draw=_iid, ref=u);
18
19  starting values;
20
21  Constant    T     2.7981373906
22  country2    T     .41548399208
23  dropout     T     -0.390239703
24  college     T     .65124745201
25  age<30      T     .07153197354
26  age30-55    T     .03429744567
27  age55+      T    -.01784057663
28  sig_eps     T     1.0666608487
29  one         F     1
30  alpha       T     0
31  sig_u       T     2.2369666396
32  ;
```

Heteroskedasticity in ε_i is ignored here. Heteroskedasticity is implemented by interacting a regressor set and a residual. The coefficient on the constant regressor, "1", must be fixed at 1, because it is not separately identified from σ_u. The null hypothesis of homoskedasticity corresponds to a zero coefficient on regressor $\tau = 1997\text{-year}$. File "hskedas3.out" contains the results of estimation with evidence of significantly less precise income responses as the recall period lengthens: $\hat{\alpha} = 0.0868$ and very significant (not shown here).

5.6. Random Coefficients Models

So far we only illustrated heterogeneity in its most common form, namely additive in a regression equation. This captures heterogeneity (stochastic variation) in the intercepts of models. In addition, there may be heterogeneity in slope coefficients. This class of models is known in the literature as random coefficient models (Hildreth and Houck, 1968; De Leeuw and Kreft, 1986; Longford 1993). In aML, random coefficients are implemented through interactions between variables (or regressor sets) and residuals, much like the heteroskedasticity described in Section 5.5.

Suppose we wish to analyze the determinants of test scores of students that are nested in schools (Hox 1995, pp. 11-16):

$$Y_{ij} = \beta_{0j} + \beta_{1j} X_{ij} + \varepsilon_{ij},$$

where Y_{ij} is the score of student *i* in school *j*. The unit of observation (level 1) is a school; students are at level 2. There is only one test per student. In deviation from the convention used throughout most of this manual, we explicitly write the school subscript *j*. There is only one explanatory variable, X_{ij}, socioeconomic status. The residual is assumed to be distributed *iid* normally: $\varepsilon_{ij} \sim N(0, \sigma_\varepsilon^2)$. The intercept, β_{0j}, and the slope coefficient, β_{1j}, are both subscripted by school identifier *j*. In other words, students with identical socioeconomic backgrounds may score differently depending on the school they attend. For example, schools which admit students on the basis of proven ability may have a brighter mix of students and a higher β_{0j} than schools without an entry exam; schools that pay more attention to students with higher socioeconomic status may show larger differences (β_{1j}) across students with different backgrounds than egalitarian schools. Both the intercept and the slope coefficient are random functions of school characteristics. We assume that the only school characteristic that matters is class size, denoted by Z_j:

$$\beta_{0j} = \gamma_{00} + \gamma_{01} Z_j + u_{0j},$$
$$\beta_{1j} = \gamma_{10} + \gamma_{11} Z_j + u_{1j},$$

where u_{0j} and u_{1j} are assumed to be normally distributed and potentially correlated. aML requires that you specify the reduced form equation, while maintaining (and estimating) all structural building blocks. Substituting for β_{0j} and β_{1j}, the test score equation becomes:

$$\begin{aligned} Y_{ij} &= \gamma_{00} + \gamma_{01} Z_j + u_{0j} + X_{ij}\left(\gamma_{10} + \gamma_{11} Z_j + u_{1j}\right) + \varepsilon_{ij} \\ &= \left(\gamma_{00} + \gamma_{01} Z_j\right) + X_{ij}\left(\gamma_{10} + \gamma_{11} Z_j\right) + u_{0j} + X_{ij} u_{1j} + \varepsilon_{ij}. \end{aligned}$$

We separated regressor sets from residuals in all interactions. The resulting expression is entirely in terms of regressor sets, variables, and residuals, and may be specified as such.

The data (`Samples\Chapter5\re.dat`) contain level 1 variable `class` (Z_j, class size) and level 2 variables `ses` (X_{ij}, socioeconomic status) and `score` (Y_{ij}, the test score). The model may be specified as follows (`random1.aml`):

```
option title = "Random effects in intercept and slope";
dsn = random.dat;

define regressor set Beta0; var = 1 class;
define regressor set Beta1; var = 1 class;

define normal distribution; dim=2; name=u0; name=u1;
define normal distribution; dim=1; name=eps;

continuous model;
   outcome = score;
   model = regset Beta0 +
      ses * regset Beta1 +
      res(draw=1, ref=u0) +
      ses * res(draw=1, ref=u1) +
      res(draw=_iid, ref=eps);

starting values;

Const0      T     29.127499803
class0      T    -.06432157198
Const1      T     3.9282273125
class1      T    -.05700642148
sig_u0      T     .91627656034
sig_u1      T     .52421460015
rho         T      0
sig_eps     T     1.2160897807
;
```

Regressor set `Beta0` represents $\gamma_{00}+\gamma_{01}Z_j$, `Beta1` corresponds to $\gamma_{10}+\gamma_{11}Z_j$; variable `ses` is X_{ij}. The first distribution is the joint normal distribution of u_{0j} and u_{1j}; the second defines ε_{ij}. The model is specified using an interaction of a variable and a regressor sets to specify $X_{ij}\left(\gamma_{10}+\gamma_{11}Z_j\right)$, and an interaction of a variable and a residual to specify $X_{ij}u_{1j}$.

As most multi-equation models, random effects models are very sensitive to good starting values. We therefore built up the model in steps (not shown here). We first estimated an ordinary least squares model of SES on test scores. Using its estimates as starting values, we estimated the two-level model of interest, but without any heterogeneity (`random0a.aml`). Using its estimates as starting values, we added heterogeneity in the intercept only, u_{0j} (`random0b.aml`). Using its estimates as starting values, we added heterogeneity in the slope, u_{1j}, but without correlation between u_{0j} and u_{1j} (`random0c.aml`). Finally, we freed up the correlation between u_{0j} and u_{1j}, as shown above (`random1.aml`).

5.7. Errors in Variables

We tend to assume that all variables used in a regression model are measured without error. In practice, however, many variables are subject to measurement error. Under some circumstances, such measurement error can lead to biased estimates of the parameters and/or their standard errors (e.g., Pindyck and Rubinfeld, 1991; Judge et al., 1985; Judge et al., 1988).

The most innocent case is when the outcome is measured with error. The main implication is that the standard deviation of the usual (implicit) residual becomes greater, so that standard errors of parameter estimates become greater. The parameter estimates themselves, however, remain unbiased.

A more serious case is when one or more explanatory covariate is measured with error. Suppose the true model is:

$$y = \beta_0 + \beta_1 x + u \,,$$

but x is not observed. Instead, we observe $x^* = x + v$, so that

$$y = \beta_0 + \beta_1 \left(x^* - v \right) + u = \beta_0 + \beta_1 x^* + \left(u - \beta_1 v \right).$$

Even if the measurement error v is normally distributed with zero mean and no autocorrelation, problems arise because explanatory variable x^* is correlated with residual $u - \beta_1 v$:

$$Cov\left(x^*, u - \beta_1 v \right) = E\left(\left(x + v \right)\left(u - \beta_1 v \right) \right) = -\beta_1 \sigma_v^2 .$$

Ignoring this type of measurement error thus leads to biased and inconsistent parameter estimates. We may approach this issue along the following lines:

```
define parameter Beta0;
define parameter Beta1;

define normal distribution; dim=1; name=u;
define normal distribution; dim=1; name=v;

continuous model;
   outcome = y;
   model = par Beta0 + xstar * par Beta1 +
           res(draw=1, ref=u) - xstar*res(draw=1, ref=v);
```

where "`xstar`" is a data variable contained the measured x^*. In this formulation, the problem is not identified. Theory needs to provide guidance on the nature of the measurement error and on the empirical implementation.

In a third case, there may be (independent) measurement error in both the outcome variable and one or more explanatory variables. The implications are similar to those with measurement

error in an explanatory variable. The direction of the bias, however, cannot be stated unambiguously. Asymptotically, however, it can be shown that ignoring these types of measurement error biases parameter estimates to zero (Pindyck and Rubinfeld, 1991). In aML, the issue may be addressed in a similar manner as with measurement error in an explanatory variable only.

5.8. Seemingly Unrelated Regression (SUR)

A system of seemingly unrelated regression equations consists of two or more equations whose residuals are correlated but that are otherwise unrelated (e.g., Pindyck and Rubinfeld, 1991). In particular, none of the outcomes enters as an explanatory covariate in any other equation. The equations may have regressors in common. Formally:

$$\begin{aligned} y_1 &= \beta_1' x_1 + u_1 \\ y_2 &= \beta_2' x_2 + u_2 \\ &\vdots \\ y_k &= \beta_k' x_k + u_k, \end{aligned}$$

where the covariance matrix of the residuals is not diagonal. The specification in aML is straightforward. Suppose there are three seemingly unrelated regression equations:

```
define normal distribution; dim=3;
   name=u1;
   name=u2;
   name=u3;

continuous model;
   outcome = y1;
   model = ... + res(draw=1, ref=u1);

continuous model;
   outcome = y2;
   model = ... + res(draw=1, ref=u2);

continuous model;
   outcome = y3;
   model = ... + res(draw=1, ref=u3);
```

The important feature is that the residuals were defined as part of the same distribution, and that their draws are the same. Then, and only then, are they correlated. In particular, "`draw=_iid`" will lead to undesired results, because the corresponding residual will be uncorrelated with everything else.

This single-level example readily generalizes to multilevel seemingly unrelated regression equations. For example:

$$\begin{aligned} y_{1i} &= \beta_1 x_{1i} + \varepsilon + u_i \\ y_{2i} &= \beta_2 x_{2i} + \eta + v_i \end{aligned}$$

where subscript i indicates replication of the outcomes; $\text{cov}(\varepsilon, \eta) \neq 0$; and $\text{cov}(u_i, v_j) \neq 0$, $i = j$, and $\text{cov}(u_i, v_j) = 0$, $i \neq j$. Suppose data variable "year" represents the replication number, i. The model may be specified as:

```
define regressor set BetaX1; var = ...;
define regressor set BetaX2; var = ...;

define normal distribution; dim=2; name=eps; name=eta;
define normal distribution; dim=2; name=u;   name=v;

continuous model;
   outcome=y1;
   model = regset BetaX1 +
           res(draw=1, ref=eps) +
           res(draw=year, ref=u);

continuous model;
   outcome=y2;
   model = regset BetaX1 +
           res(draw=1, ref=eta) +
           res(draw=year, ref=v);
```

Residuals eps and eta are correlated because they are defined as part of the same distribution and because their draw variable is always the same. Residuals u and v are pairwise correlated because they are defined as part of the same distribution and because draw variable "year" ties the pairs together.

5.9. Overlapping Splines

The term "overlapping splines" refers to hazard models in which dependencies on multiple durations may combine to form the baseline hazard. The duration since the moment at which the hazard event became at risk of occurring plays a central role in almost all hazard models. In addition, aML offers the capability of allowing the hazard to be a function of other durations.

For example, in an analysis of fertility (hazard of a conception), the central duration is the time since the woman became at risk of becoming pregnant. For the first conception, this is the time since menarche (during puberty years), and for subsequent conceptions, it is the time since the previous birth (or, more precisely, since the return of menses after giving birth). However, there may be more time concepts that are relevant:

- The woman's age may matter because of biological factors. We could capture age effects through several time-varying indicator variables, but a continuously changing function of age may appeal more.
- Couples may postpone childbearing until after marrying. We could capture the role of marriage through simple time-varying indicator variables for marital status, but that may not capture all dynamics. If births are spaced differently in the early years of marriage than in later years, a continuous function of duration since the wedding may be more appropriate.
- Similarly, we could capture the effect of school enrollment through a simple dummy variable. However, if women decide to postpone childbearing until after graduation, there may be a surge in fertility hazard shortly after leaving school, probably tapering off after the "saved up" interest in children is satisfied. Again, a continuously changing function of duration since leaving school may capture more of the richness in the data than a simple indicator variable for school enrollment.
- Given long term variations in fertility rates, calendar time may play a role. We could capture the effect of calendar time through time-varying covariates, but a continuously changing time trend may appeal more than a stepwise-changing trend. Unlike dependencies on other durations, there is no natural moment at which the time clock starts ticking. You may select any origin; see below.

In suggesting alternative ways to capture the effects of various time concepts, we used time-varying covariates rather than covariates that are constant for the duration of the birth interval. For example, some models in the literature include woman's age at the beginning of the birth interval as a covariate. If the birth interval starts at age 25, and the woman has no more children, such birth intervals may last several decades, and the initial age is not a very good measure of biological effects any more. Time-varying covariates are thus far preferable. However, by definition, time-varying covariates change discretely from one sub-interval to the next, and their effect on a hazard thus consists of discrete jumps. We tend to opt for continuously changing (piecewise-linear) duration patterns instead. They have the additional advantage that there is no need to break up the spell into subintervals between which time-varying covariates change.

Hazard duration dependencies in aML are always piecewise-linear in the log-hazard. The user selects the nodes (bend points, knots), and aML estimates the slopes between nodes. aML cannot estimate the optimal number or location of nodes; see Section 6.6 for suggestions on selecting nodes.

To see how duration dependencies combine to form the total baseline duration dependency, consider the following model specification of the hazard of conceiving a child:

```
define spline SpellDur;  nodes = 2 5;    intercept;
define spline Age;       nodes = 18 30;
define spline MarDur;    nodes = 1 5 10;
define spline OutSchool; nodes = 2;      intercept; effect=right;
define spline CalTime;   nodes = ;

hazard model; keep if (married==1 and parity==2);
   censor=...;  duration=...;
   model = durspline(origin=0, ref=SpellDur) +
           durspline(origin=age, ref=Age) +
           durspline(origin=mardur, ref=MarDur) +
           durspline(origin=school, ref=OutSchool) +
           durspline(origin=time, ref=CalTime) +
           ...;
```

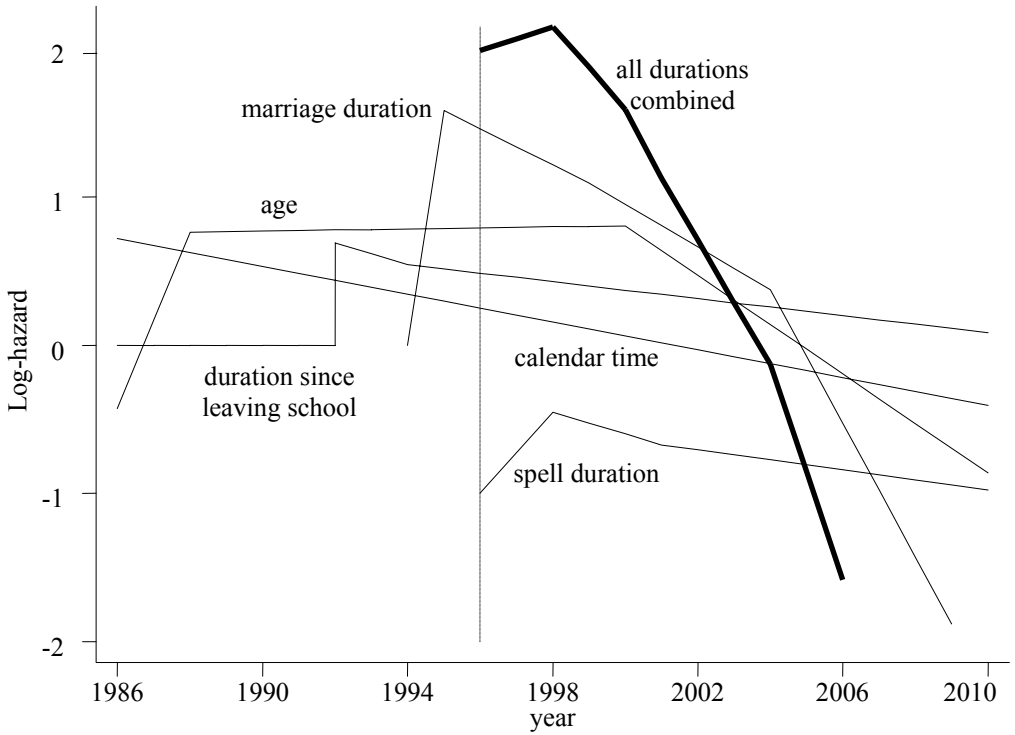
This model only applies to the second birth interval of married women. The hazard of conceiving a child is assumed to depend on:

- `SpellDur`: the duration of the spell itself, i.e., the duration since the birth of the previous child. It is the main duration and its pattern has nodes at 2 and 5 years. We tend to assign an intercept to the dependency on the main duration, but it may equivalently be done as part of a regressor set. The "`origin=0`" in the model specification makes the spell duration clock starts ticking at the beginning of the spell.

- `Age`: the age of the respondent, with nodes at 18 and 30 years of age. The age clock starts ticking at "`origin=age`", where age is a data variable representing the age at the beginning of the spell. It is at the same level as the censor and duration variables, and not time-varying. If a woman is, say, 26 at the beginning of the spell, the age clock has accumulated 26 years of age effect. The node at 18 years has thus long passed, and the node at 30 years comes four years into the spell.

- `MarDur`: the duration since the wedding, with nodes at 1, 5, and 10 years after the wedding. The marriage duration clock starts ticking at "`origin=mardur`", where `mardur` is a data variable representing how many years the woman has been married at the beginning of the spell. It is at the same level as the censor and duration variables, and not time-varying. If the woman married, say, two years before the beginning of the spell, the node at one year has already passed, and the effective marriage duration nodes will be three and eight years into the spell.

- `OutSchool`: the duration since leaving school, with a node at 2 years. Some women may leave school sometime during the spell. We wouldn't want the effect of leaving school to affect the period before leaving school, so we specify "`effect=right`". The "`intercept`" option makes the hazard of a conception jump up when the woman leaves school. The clock starts ticking at "`origin=school`", where `school` is a data variable representing how many years the woman has been out of school at the beginning of the spell. It is at the same level as the censor and duration variables, and not time-varying. If the woman graduated, say, four years before the beginning of the spell, the node at two years has already passed, and the effect of duration since leaving school is linear throughout the conception spell.
- `CalTime`: a dependency on calendar time, measured relative to an arbitrary point in time. The time clock starts ticking at "`origin=time`", where time is a data variable representing the number years since the arbitrarily chosen origin at the beginning of the spell. It is at the same level as the censor and duration variables, and not time-varying.



Overlapping Splines

Consider a woman who was born in 1970. She left school in 1992, married in 1994, and delivered her first baby at age 26 in 1996. Her second conception spell thus starts in 1996. We measure calendar time relative to the year 2000. At the beginning of the spell, her data variables are: `age=26`; `mardur=2`; `school=4`; `time=-4`. The figure above visualizes the five duration dependencies that combine to form her total baseline duration dependency.[25]

The dashed vertical line indicates 1996, when her conception spell starts. The main duration dependency, on spell duration, is pictured toward the bottom. The log-hazard of a conception increases at 0.27 per year for the first two years, decreases at –0.07 per year until the fifth year, and decreases at –0.03 per year thereafter. Note the nodes (bend points) in the figure at two and five years after the beginning of the spell, in 1998 and 2001. The dependency on age with nodes at 18 and 30 years translate into 1988 and 2000, as is clearly visible. (The portion of the age dependency before the beginning of the spell in 1996 is irrelevant to the conception interval. The figure only shows it to illustrate how the relevant portion of the age pattern is constructed, including the intercept shift that the age pattern contributes to the overall baseline hazard at the beginning of the hazard spell. The pre-spell portions of the effects of marriage duration, duration since leaving school, and calendar time are shown for the same reasons.) The calendar time trend is downward, reflecting decreasing fertility over the estimation sample period. The spell starts just two years after the wedding, so that marriage duration contributes substantially to the hazard of a conception. The marriage duration node at one year fell before the spell; the nodes at five and ten years translate into three and eight years into the spell, i.e., 1999 and 2004, respectively. (The node at 1999 is barely visible, because the slopes before and after five years into a marriage are almost identical.) The dependency on duration since school was without effect until 1992, when she left school. At that time, the log-hazard jumped up by 0.69. It then decreased for two years at –0.07 per year, and subsequently at –0.03 per year.

The thick line shows the aggregate duration dependency. It is simply the vertical aggregation of the five component dependencies. It may be clear now why aML requires duration dependencies to be piecewise-linear: the sum of any number of piecewise-linear duration dependencies is again piecewise-linear.

[25] The patterns were estimated on a sample of 5,825 women in the 1979-1991 National Longitudinal Study of Youth, and extrapolated in time for the simulation in the figure. With the exception of the age slope between 18 and 30 years and the spell duration slope after five years, all slopes were strongly significant with t-statistics that exceeded 2.5 in absolute value. The model was based on all birth intervals, both inside and outside of marriage. It controlled for many other covariates. It is a simplified version of the model described in Lillard, Panis, and Upchurch (1996).

All duration patterns are individually identified, because there is great variation in the timing of women's birth spells. In other words, women start conception spells at many different ages, which separately identifies the age pattern from the spell duration pattern. The spells also start at many different wedding durations, which identifies the marriage duration pattern. Et cetera. The location of nodes and the magnitude of slopes of the aggregate duration dependency are functions of the nodes and slopes of component dependencies, and the spline origins, i.e., moments at which their corresponding clocks started ticking.

5.10. Autoregressive and Moving Average Residuals

In addition to residuals from normal and finite mixture distributions, aML supports stationary autoregressive and moving average residuals. Autoregressive and moving average residuals v_t are defined by (e.g., Pindyck and Rubinfeld, 1991):

$$v_t = \begin{cases} \phi_1 v_{t-1} + e_t & \text{AR(1) model;} \\ \phi_1 v_{t-1} + \phi_2 v_{t-2} + e_t & \text{AR(2) model;} \\ e_t + \theta_1 e_{t-1} & \text{MA(1) model;} \\ e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} & \text{MA(2) model;} \\ \sum_{i=1}^{p} \phi_i v_{t-i} + e_t + \sum_{j=1}^{q} \theta_j e_{t-j} & \text{ARMA}(p,q) \text{ model,} \end{cases}$$

where subscript *t* denotes a time period and e_t is the innovation term, $e_t \sim N(0, \sigma_e^2)$. We suppressed the observation-level subscript. The time period must be integer-valued, except for AR(1) residuals:

$$v_t = \phi_1^{|t-s|} v_s + e_t ,$$

where *s* and *t* denote time periods. If $|t-s|$ is non-integer, ϕ_1 must be positive; see below.

Consider an AR(1) process with realizations that are one period apart. Its variance is $\sigma_v^2 = \sigma_e^2 \big/ \left(1 - \phi_1^2\right)$, often denoted by γ_0, and the autocovariance for a *k*-lag is $\gamma_k = \phi_1^k \gamma_0$. For example, the autocovariance matrix of an AR(1) process with eight realizations (outcomes) is:

$$\Sigma_{vv} = \frac{\sigma_e^2}{\left(1-\phi_1^2\right)} \begin{pmatrix} 1 & & & & & & & \\ \phi_1 & 1 & & & & & & \\ \phi_1^2 & \phi_1 & 1 & & & & & \\ \phi_1^3 & \phi_1^2 & \phi_1 & 1 & & & & \\ \phi_1^4 & \phi_1^3 & \phi_1^2 & \phi_1 & 1 & & & \\ \phi_1^5 & \phi_1^4 & \phi_1^3 & \phi_1^2 & \phi_1 & 1 & & \\ \phi_1^6 & \phi_1^5 & \phi_1^4 & \phi_1^3 & \phi_1^2 & \phi_1 & 1 & \\ \phi_1^7 & \phi_1^6 & \phi_1^5 & \phi_1^4 & \phi_1^3 & \phi_1^2 & \phi_1 & 1 \end{pmatrix}.$$

aML does not require (or assume) that the realizations are spaced one period apart; see below.

In its simplest form, an ARMA residual is defined as follows:

```
define arma(p,q) distribution;  dim=1;
   timevar = varname;
   name = resname;
```

These statements define a univariate ARMA(*p*,*q*) distribution. The autoregressive and moving average order are limited to $p \le 9$ and $q \le 9$. aML only supports univariate ARMA distributions. The "`dim=1`" statement is optional. You may define purely autoregressive distributions as "`ar(p)`" or "`arma(p,0)`", and purely moving average distributions as "`ma(q)`" or `arma(0,q)`".

ARMA residuals may only be used in continuous models. aML does not assume that all continuous outcomes are one time period apart; it requires that you specify a variable which contains the time that corresponds to an outcome. The time variable, specified in the mandatory "`timevar`" statement, must be at the same level as the outcome variable of the continuous model specification. Its absolute value is irrelevant; only the time difference between outcomes is relevant. For example, if your data contain annual outcomes between 1991 and 1996, you may create a time variable that is equal to 1991, 1992, ..., 1996. Equivalently, the time variable may be 1, 2, ..., 6, or any other set of equally spaced integer variables.

The distribution's residual must be named (`name=resname`), just like residuals of normal and finite mixture distributions.

As mentioned before, the time variable must be integer-valued, except for AR(1) distributions. The autocorrelation for an AR(1) residual with realizations at times *s* and *t* is equal to $\gamma_{|t-s|} = \phi_1^{|t-s|}\gamma_0$. If $|t-s|$ is non-integer, this requires that ϕ_1 is strictly positive, as indicated above. Stationarity further requires that ϕ_1 is less than one. aML therefore only accepts non-integer time variables for AR(1) distributions of which the autoregressive coefficient has been restricted to the (0,1) domain:

```
define ar(1) distribution;  dim=1;
   timevar = varname;
   ardomain=(0,1);
   name = resname;
```

You may always restrict autoregressive coefficients, even if the time variables are integer-valued. Similarly, you may restrict moving average coefficients. The following restrictions are supported:

```
ardomain = (-1,1);
ardomain =  (0,1);
madomain = (-1,1);
madomain =  (0,1);
madomain =  (0,Inf);
```

The restrictions apply to all autoregressive or moving average coefficients. In other words, you cannot restrict the first autoregressive coefficient of an AR(2) process to (0,1) and leave the other unrestricted.

Consider an example. Data file "Samples\Chapter5\arma.dat" contains variables that were generated with ARMA processes. Data structure 10 consists of two levels. The first level is, of course, the observation; the second level corresponds to points in time, as subscripted by *t* above. We estimate the following model with ARMA(2,2) residual:

$$y_t = \beta' x_t + v_t \text{ , where } v_t = \phi_1 v_{t-1} + \phi_2 v_{t-2} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2}$$

The outcome variable in the data is "y", a level 2 variable. The time variable is "time"; it is at the same level as the outcome, as it should be. Explanatory covariates are "var1" (at level 1), "var2" (at level 2), and "var3" (at level 3). The model may be specified as (arma1.aml):

```
 1  dsn = arma.dat;
 2
 3  define regset BetaX; var = 1 var1 var2 var3;
 4
 5  define arma(2,2) distribution;
 6     timevar = time;
 7     name = v;
 8
 9  continuous model; data structure = 10;
10     outcome = y;
11     model = regset BetaX +
12             res(draw=_iid, ref=v);
13
14  STARTING VALUES;
15
16  Constant    TTTTT    2.548
17  var1        TTTTT    0
18  var2        TTTTT    0
19  var3        TTTTT    0
20  rho1        FTTTT    0
21  rho2        FFFTT    0
22  SigmaV      TTTTT    1.83
23  theta1      FFTTT    0
24  theta2      FFFFT    0
25  ;
```

Lines 5-7 define the ARMA(2,2) distribution and its residual, "v". It is used in the model specification on line 12. Its draw is specified as "draw=_iid".

ARMA residuals must always be drawn independently at the outcome level with "draw=_iid".

Lines 16-24 initialize the parameters. An ARMA(p,q) distribution generates p+q+1 parameters which must be initialized in the following order: p autoregressive coefficients, one standard deviation of the innovation term, and q moving average coefficients.

The search for optimal ARMA parameters can be very sensitive to good starting values. We strongly recommend that you build up the model in small steps. The control file above first estimates the coefficients of the regressor set and the standard deviation of the innovation. The autoregressive and moving average coefficients are fixed at zero, i.e., this is a model with covariates and a normal (non-ARMA) residual only. The starting values of the constant and the standard deviation are the mean and standard deviation of the outcome, respectively. Upon convergence, we introduce first-order autoregression in the model by freeing up ϕ_1 (`phi1`). In the next step, we estimate an ARMA(1,1), then ARMA(2,1), and finally ARMA(2,2). The results are in "`arma1.out`" (not shown).

Non-Chronological Data

In our data, there were up to 12 outcomes per observation. They appeared in the data in chronological order. This will typically be the case, and aML checks by default that the time variables are indeed ordered. If they are not, it issues an error message. However, you may turn that check off:

```
timevar = varname;  increasing=no;
```

The default is "`increasing=yes`". Experience shows that proper creation of time variables can be tricky, and we recommend that you keep outcomes in chronological order and allow aML to perform this data integrity check.

ARMA Residuals in a Multilevel Model

The example above illustrated ARMA residuals in a two-level model. What if there are additional levels, and you have reason to believe that the autocorrelation is restricted to sub-units? Suppose we have a data set with wages of couples. The couple is level 1, the husband and wife are level 2 branches, and individual wage records are level 3 subbranches. We expect annual wages of a person to be autocorrelated, but don't expect that the wage of the husband in a year is correlated with the wage of the wife in the preceding year (other than perhaps through couple-level unobserved heterogeneity). Consider a couple for which we have three wage records of the husband and five records for the wife. We want the covariance matrix of the eight wages to be block diagonal:

$$\Sigma_{vv} = \frac{\sigma_e^2}{\left(1-\phi_1^2\right)} \left(\begin{array}{ccc:ccccc} 1 & & & & & & & \\ \phi_1 & 1 & & & & & & \\ \phi_1^2 & \phi_1 & 1 & & & & & \\ \hdashline 0 & 0 & 0 & 1 & & & & \\ 0 & 0 & 0 & \phi_1 & 1 & & & \\ 0 & 0 & 0 & \phi_1^2 & \phi_1 & 1 & & \\ 0 & 0 & 0 & \phi_1^3 & \phi_1^2 & \phi_1 & 1 & \\ 0 & 0 & 0 & \phi_1^4 & \phi_1^3 & \phi_1^2 & \phi_1 & 1 \end{array}\right)$$

In other words, we want autocorrelation within persons (level 2 branches), but not across persons. This is achieved in the definition of the ARMA residual:

```
define arma(p,q) distribution;  dim=1;
   timevar (within level 2) = varname;
   name = resname;
```

By default, aML assumes that all ARMA residuals are correlated, which may equivalently be stated by "`timevar (within level 1) = varname`". You may specify any level within which ARMA residuals are correlated, provided that it is at least one level above the outcome level. It would not make sense to specify that the residuals are correlated within level 3 if the outcome is at level 3, because the residuals would then always be independent.

As before, only absolute values of differences in time variables are relevant. With three wage outcomes for the husband and five for the wife, the differences between the time variables of the first three and the latter five outcomes are relevant; the difference between the third (husband's last) and fourth (wife's first) time variables is again irrelevant.

You may specify multiple ARMA processes that are autocorrelated within different levels. For example, if the couple wage data would include an additional layer for jobs at level 3, you could specify one ARMA residual that is correlated with all wage observations of a spouse (`within level 2`) and a second ARMA residual that is correlated within jobs only (`within level 3`).

Data structure 20 in "`Samples\Chapter5\arma.dat`" contains variables that are autocorrelated only within level 2. It consists of three levels. The first level is the observation; the second level corresponds to individuals (or other mid-level units); and the third level corresponds to points in time. We estimate the following model with AR(1) residual:

$$y_{it} = \beta' x_{it} + v_{it}, \text{ where } v_{it} = \phi_1 v_{it-1} + e_{it},$$

and $\text{cov}\left(v_{it}, v_{js}\right) = 0$ for all $i \neq j$.

The outcome variable in the data is "`y`", a level 3 variable. The time variable is "`time`"; it is at the same level as the outcome, as it should be. Unlike in the preceding example, it is not

necessarily integer-valued. Explanatory covariates are "var1" (at level 1), "var2" (at level 2), and "var4" (at level 3). There are up to three level 2 branches per observation, and up to twelve level 3 subbranches in any one level 2 branch. The model may be specified as (arma2.aml):

```
 1  dsn = arma.dat;
 2
 3  define regset BetaX; var = 1 var1 var2 var4;
 4
 5  define ar(1) distribution;
 6     timevar (within level 2) = time;
 7     ardomain=(0,1);
 8     name = v;
 9
10  continuous model; data structure = 20;
11     outcome = y;
12     model = regset BetaX +
13             res(draw=_iid, ref=v);
14
15  starting values;
16
17  Constant    TT    -1.60674
18  var1        TT    0
19  var2        TT    0
20  var4        TT    0
21  phi1        FT    0.3
22  SigmaV      TT    1.423
23  ;
```

Line 6 specifies that the residual is correlated with other residuals within level 2, but not across level 2 branches. Line 7 restricts the autoregressive coefficient to be between zero and one, as is required with data that are not integer-spaced. The remaining is similar to the example above. The results are in "arma2.out" (not shown).

Cumulative Autoregressive Residuals

aML supports a variation of first order autoregressive residuals, the cumulative first order autoregressive residual, or CAR(1). Unlike regular AR(1) residuals, the CAR(1) does not assume that the residual series has infinite history. It assumes that the series started at period 1 with just an innovation term:

$$v_1 = e_1$$
$$v_t = \phi_1 v_{t-1} + e_t \quad \text{for } i > 1.$$

As a result, the variance of a CAR(1) is not constant but increases over time from σ_e^2 at time t=1 to $\sigma_e^2 / \left(1-\phi_1^2\right)$. The definition of a CAR(1) residual is

```
define car(1) distribution;
```

with otherwise identical further specifications and options. Its use in model statements is identical to the use of AR(1) residuals.

In addition to the increasing variance, there is one further important distinction between CAR(1) and AR(1) residuals. The absolute value of an AR(1) time variable is irrelevant, because the variance and autocovariances only depend on the differences between time variables. By contrast, the absolute value of the CAR(1) time variable does matter. The series is assumed to start at time t=1.

5.11. Indirect Referencing and Conditional Building Blocks

aML models are specified in terms of previously defined building blocks such as regressor sets, residuals, splines, parameters, et cetera. For example:

```
define regset BetaX; var = ...;
define normal distribution; dim=1; name=eps;
model = regset BetaX + res(draw=_iid, ref=eps);
```

We first defined a regressor set and gave it a name, `BetaX`. Similarly, we defined a distribution and named its residual, `eps`. These building blocks were referenced by name in the model statement. In aML terminology, they were "directly referenced." aML supports an alternative way to refer to building blocks, "indirect referencing."

Indirect referencing requires that building blocks are assigned one or more numbers, so-called reference numbers. Think of these reference numbers as an alternative to names. Indeed, building blocks do not need to have a name; a reference number is sufficient. For example:

```
define regset; ref=3; var = ...;
define normal distribution; dim=1; name=eps; ref=44;
```

The regressor set no longer has a name. It cannot be referenced directly anymore. However, it has a number, 3, and may be identified by that number. The residual has both a name and a number; it may be directly or indirectly referenced. Indirect referencing is done with a reference variable or expression:

```
model = regset(refvar=educ) +
        res(draw=1, refvar=category);
```

where `educ` and `category` are data variables. If `educ` equals 3, the regressor set is included in the equation; if `category=44`, the residual enters.

Indirect referencing allows the selection of building blocks on the basis of data variables, i.e., it allows different model specifications for different observations. Suppose we want to estimate separate equations for three education categories, identified by `educ`=1, 2, and 3:

```
define regset Dropouts;  ref=1; var=...;
define regset Graduates; ref=2; var=...;
define regset College;   ref=3; var=...;

model = regset(refvar=educ) + ...;
```

Depending on the value of `educ`, one of three different regressor sets is included in the equation. We did not have to assign names to the regressor sets, but did so anyway to make it easier to understand the control file.

You may use expressions to indirectly reference building blocks:

```
regset(refvar=100*sex+educ)
regset(refvar=12*(educ==2)+max(age,18))
```

Et cetera. All usual expression operators are allowed (Section 13.17).

While you may only assign one name to any one building block, you may assign as many reference numbers as you like. This offers a powerful way to quickly explore interactions. For example, to combine high school drop-outs (`educ=1`) and high school graduates (`educ=2`) but distinguish them from college graduates (`educ=3`), you may specify:

```
define regset NoCollege; ref=1 2; var=...;
define regset College;   ref=3;   var=...;

model = regset(refvar=educ) + ...;
```

What if the reference variable or expression evaluates to a number which was not found among any definition? If the reference variable is nonzero, this will result in an error message. Zero, however, is a value with special meaning. If the reference variable evaluates to zero, aML omits the building block from the equation. For example, suppose a model includes some person's characteristics, plus the characteristics of his/her spouse, if the person is married:

```
define regset Everyone;             var=...;
define regset MarriedOnly;  ref=1;  var=...;

model = regset Everyone +
        regset(refvar=married) + ...;
```

If `married=1`, both the first and the second regressor set enter the model; if `married=0`, the second term drops out.

Reference numbers must be strictly positive integers. They may not be zero, because zero has the special interpretation that the building block should not be in the equation.

Reference variables (and variables involved in a reference expression) may be at any level that is equal to or higher than the outcome variable's level. It would not make sense to determine the model specification of a level 3 variable based on a level 4 variable. However, it is fine to make the selection based on a level 1, 2, or 3 variable.

Just like you may not assign the same name to two or more regressor sets (or other building blocks), you may not assign duplicate reference numbers. More precisely, reference numbers must be unique within building block types, but may be re-used for building blocks of another type. It is, for example, permissible to assign a regressor set and a spline the same reference number. There are four types of building blocks: regressor sets, splines, distributions (residuals), and parameters/vectors. (aML treats parameters and vectors as the same type, so all parameter and vector reference numbers must be unique.)

In summary, the following rules apply to indirect referencing:

- Reference numbers are strictly positive integer-valued numbers by which building blocks may be identified. They must be unique within building block types, but the same number may be used in different building block types.
- Model statements may specify building blocks directly, by name, or indirectly, through reference variables or expressions. For every outcome (every equation), aML evaluates the reference variable and attempts to match its value to defined building blocks of the appropriate type.
- If the reference variable evaluates to zero, the building block does not enter the equation. If the building block was interacted with other building blocks, the entire interaction term is excluded. If the reference variable evaluates to a nonzero number which was not assigned to any building block, aML aborts with an error message.
- Reference variables may be at any level that is at or above the level of the outcome variable.

In most cases, indirect referencing is merely a convenient and concise alternative to multiple model statements that apply to a subset of cases and that reference building blocks by name. For example, the following model specification

```
define regset Dropouts;  ref=1; var=...;
define regset Graduates; ref=2; var=...;
define regset College;   ref=3; var=...;

probit model;
   outcome=varname;
   model = regset(refvar=educ) + ...;
```

may equivalently be written as:

```
probit model; keep if educ==1;
   outcome=varname;
   model = regset Dropouts + ...;

probit model; keep if educ==2;
   outcome=varname;
   model = regset Graduates + ...;

probit model; keep if educ==3;
   outcome=varname;
   model = regset College + ...;
```

In other cases, indirect referencing enables models to be specified which would otherwise be very tedious. For example, we model the stability of a marriage as a function of the number and age composition of the couple's children. We want to allow for the possibility that the effect of a child on the hazard of divorce varies with the child's age. The data contain couples with zero to,

say, 16 children. An efficient way to capture the effects of children on the hazard of marital disruption is:

```
define spline ChildEffect; ref=1;
   intercept; effect=right; nodes=...;
hazard model;
   censor=...;  duration=...;
   model = ... +
           durspline(origin=kiddur1, refvar=kid1) +
           durspline(origin=kiddur2, refvar=kid2) +
           ... +
           durspline(origin=kiddur16, refvar=kid16);
```

where `kiddur1` through `kiddur16` are variables measuring the time from the wedding to the birth of a child (if any). For children born before the wedding, this variable is positive; for children born during the marriage, it is negative. Most couples have fewer than 16 children; their corresponding `kiddur` variables are irrelevant and missing in the (SAS, Stata, SPSS) data. For conversion into aML-formatted data, we recommend that you set them equal to 99999, for a reason that will become clear shortly. Variables `kid1` through `kid16` equal one if the corresponding child is ever born and zero otherwise. Consider a couple with two children. Its `kid1=kid2=1`, and `kid3` through `kid16` are all zero. The `ChildEffect` spline thus enters twice into their divorce hazard equation. They make the hazard jump (up or down, depending on the sign of the intercept that is part of the spline). Their effect is only felt after the child's birth, not throughout the spell (`effect=right`). The other 14 potential entries are omitted, because their reference variables `kid3-kid16` evaluate to zero.

One could specify this model by interacting each duration spline by a regressor set with one variable that evaluates to one if the child is relevant and zero if it is not. You would need sixteen of those regressor sets, each consisting of one variable with a coefficient fixed to one. There would thus be sixteen redundant parameters in the model. Computationally, it would be grossly inefficient, because all sixteen interactions need to be evaluated and accumulated for every couple, including those with no or few children. In short, indirect referencing with reference variables that may become zero offers a superior alternative.

One issue remains: what to do with origin variables `kiddur1` through `kiddur16` that are irrelevant because there was no corresponding child? They may be set equal to any value. Experience shows that the data preparation for problems like the one described here can be tricky. By default, aML therefore verifies that you really intended a duration spline to drop out of the equation by checking that its matching origin variable is equal to 99999. You may turn off that check by specifying "`option check99999=no` (page 281). There is no such check for other types of building blocks.

5.12. Interactions of Building Blocks

Section 5.5 and 5.6 showed examples where parameters and variables were interacted with regressor sets and residuals in order to capture heteroskedasticity or random coefficients. More generally, parameters and regressor sets may be interacted with other parameters and regressor sets, as well as with duration splines and (integrated) residuals. The rules for regressor sets apply equally to simple variables and regressor splines.

This allows for nonlinearities of the following types:

$\lambda_1\lambda_2$	two parameters;
$\lambda(\beta'X)$	parameter and regressor set;
$(\beta_1'X_1)(\beta_2'X_2)$	two regressor sets;
λx	parameter and variable;
$x(\beta'X)$	variable and regressor set;
$\lambda(\gamma'T)$	parameter and duration spline;
$(\beta'X)(\gamma'T)$	regressor set and duration spline;
$\lambda\varepsilon$	parameter and (integrated) residual;
$x\varepsilon$	variable and (integrated) residual;
$(\beta'X)\varepsilon$	regressor set and (integrated) residual.

Regressor splines, which are internally treated as a special case of regressor sets, may be interacted in the same way as regressor sets. There is no limit to the number of factors in interactions, so that, for example, $\lambda_1(\beta_1'X_1)(\beta_2'X_2)\lambda_2(\beta_3'X_3)X_4\varepsilon$ may be specified, should that make sense.

The ability to interact regressor sets extends the class of models that aML supports beyond just linear models. Furthermore, regressor set interactions are allowed in any type of model, not just continuous outcome models.

While multiple parameters and regressor sets/splines may appear in interactions, only one duration spline or one residual may be specified. It is not clear what the interaction of two duration splines or two residuals would mean, and aML does not allow them. Furthermore, interactions between duration splines and (integrated) residuals are not allowed.

Interactions are specified in the model statement. For example:

```
model = par Lambda1 * par Lambda2 + ...;
model = par Lambda * regset Xbeta + ...;
model = regset XBeta1 * regset Xbeta2 + ...;
```

```
model = par Lambda * varname + ...;
model = varname * regset BetaX + ...;
model = par Lambda * durspline(origin=..., ref=...) + ...;
model = regset XBeta * durspline(origin=..., ref=...) + ...;
model = par lambda * res(draw=..., ref=eps) + ...;
model = par lambda * intres(draw=..., ref=eps) + ...;
model = varname * res(draw=..., ref=eps) + ...;
model = regset XBeta * res(draw=..., ref=eps) + ...;
model = regset XBeta * intres(draw=..., ref=eps) + ...;
```

Parameters, regressor sets, and regressor splines may be interacted in any order. However, if an interaction involves a duration spline, then all other building blocks must be listed first. Similarly, if the interaction involves a residual or integrated residual, then this residual must be listed as the last factor.

For example, the following specifications are illegal:

```
model = durspline(origin=..., ref=...) * par Lambda + ...;
model = intres(draw=..., ref=eps) * regset XBeta + ...;
```

Indirectly Referenced Interaction Terms

Building blocks that are interacted with other building blocks may be directly or indirectly referenced, just like standalone building blocks. If a standalone building block is indirectly referenced, and its reference variable evaluates to zero, it drops out of the equation (Section 5.11). If the reference variable of an interacted building block evaluates to zero, the entire interaction drops out of the equation.

Interacting Parameters, Regressor Sets, and/or Regressor Splines

There are very many models which require interactions of parameters, regressor sets, and/or regressor splines. Random coefficient models typically require the interaction of two regressor sets (Section 5.6). For example, suppose that an individual's rate of wage growth depends on his intelligence level. In a model explaining individuals' log-wages, we may interact a measure of intelligence with some age pattern. Suppose we capture the age pattern by a spline with nodes at 30 and 55 years of age. A simple model (without random coefficients) may be:

$$\ln W = \beta_0 + \beta_1 IQ + (\beta_2 + \beta_3 IQ)\gamma' A + \varepsilon,$$

where $\ln W$ denotes the log-wage of the individual whose subscript we suppressed; IQ is Intelligence Quotient; A is a spline transformation of age, and $\varepsilon \sim N(0, \sigma_\varepsilon^2)$. The model may be specified as:

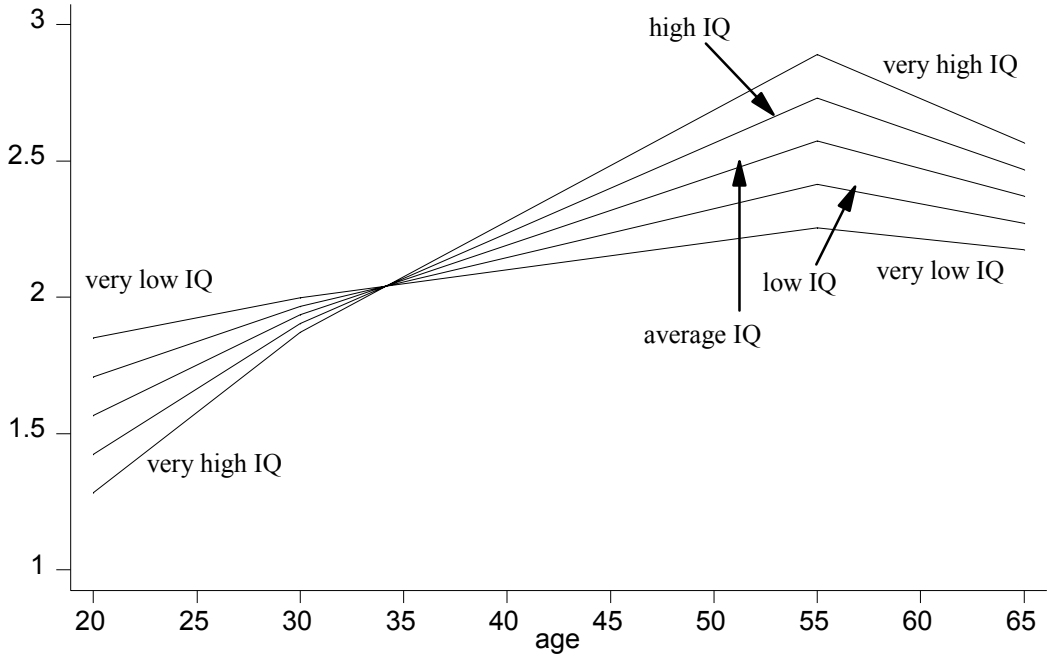
```
define regressor set XBeta0;     var = 1 IQ;
```

```
define regressor set Interact;   var = 1 IQ;
define regressor set AgeProfile; var = spline(age, 30 55);

define normal distribution; dim=1; name=eps;

continuous model;
   outcome = log(wage);
   model = regset XBeta0 +
      regset Interact * regset AgeProfile +
      res(draw=1, ref=eps);
```

The results are perhaps best interpreted with a picture. The figure below shows the simulated wage paths for individuals with mean IQs and for those one and two standard deviations below and above the mean. Note that the profile tilts around a certain age, depending on the values of the estimated parameters. (We included `IQ` in the equation both directly and through the age interaction. If the direct term were omitted, the pattern would be forced to tilt around zero, which is generally undesirable.) In the simulation, that age is in the displayed range, but it could be anywhere. A potential interpretation for the observed pattern is that highly intelligent individuals tend to stay in school longer and/or need more time to find a good job match. Their intercepts are lower, but their wages grow faster than of less intelligent persons.



Simulated Wage Profile, Interacted With IQ

Interacting a Duration Spline

The interaction of a parameter or regressor set with a duration spline, in hazard models, gives similar results: the duration dependence is tilted. Duration spline interactions commonly appear in multiprocess models. For example, suppose that the desire of couples to conceive a child is in part dependent on the stability of their marriage (Lillard, 1993). The model of fertility timing should thus be estimated jointly with a model of divorce risk:

$$\begin{aligned} \text{divorce:}\quad & \ln h^d(t) = \gamma_1' T_1(t) + \beta_1' X_1 + \varepsilon; \\ \text{conception:}\quad & \ln h_i^c(t) = \lambda \ln h^d(t) + \gamma_2' T_2(t) + \beta_2' X_2 + \delta \\ & = \lambda\gamma_1' T_1(t) + \gamma_2' T_2(t) + \lambda\beta_1' X_1 + \beta_2' X_2 + \lambda\varepsilon + \delta, \end{aligned}$$

where $\ln h^d(t)$ is the log-hazard that a couple whose subscript we suppressed gets divorced at time t, and $\ln h_i^c(t)$ is the log-hazard that this couple conceives its i-th child. The term $\lambda\gamma_1' T_1(t)$ interacts a parameter and a duration spline. It contributes a tilted divorce hazard duration pattern to the fertility baseline hazard.

Duration splines may also be interacted with regressor sets, so that the degree of tilting varies across observations. Hazard models allow regressor sets with time-varying variables that are one level below the outcome level, provided that those regressor sets enter additively in the log-hazard equation. If they are interacted with a duration spline (or integrated residual), they may only contain variable at or above the outcome level.

The effect of a covariate in a regressor set that is interacted with a duration spline does not shift the baseline duration pattern proportionally, but tilts it. The ability to interact regressor sets with duration spline thus extends the class of hazard models that aML supports beyond proportional hazard models.

Interacting a Residual or Integrated Residual

Residuals and integrated residuals may be interacted with parameters, regressor sets, and regressor splines. As illustrated above, this allows for the specification of random coefficients models and (other) forms of heteroskedasticity. It also commonly appears in multiprocess models, such as in the joint divorce-fertility model above, $\lambda\varepsilon$.

The variance of an interacted residual $\lambda\varepsilon$ is $\text{var}(\lambda\varepsilon) = \lambda^2 \text{var}(\sigma_\varepsilon^2)$, i.e., the same for positive and negative λ's of the same magnitude. However, the implied correlation(s) with other residuals may have switched signs. For example, if a two-equation model with ε in one equation and $\lambda\varepsilon$ in the other, the correlation between ε and $\lambda\varepsilon$ is equal to 1 if $\lambda > 0$ and –1 if $\lambda < 0$. Similarly, if δ and ε are defined as part of the same distribution with correlation ρ, the correlation between δ and $\lambda\varepsilon$ is equal to ρ if $\lambda > 0$ and $-\rho$ if $\lambda < 0$.

6. Starting Values

6.1. General Overview

The importance of very carefully specifying starting values can hardly be overstated. If initial parameter values are far removed from their optimal values, the search process may take a long time, and in many cases optimization fails altogether. Furthermore, the likelihood function need not be concave when equations are combined, so that poor starting values may lead to a local likelihood maximum. Always specify starting values as close to optimal values as possible, using all information available. This chapter summarizes some rules of thumb for selecting good starting values. Parameters are initialized in the "`starting values`" statement (Section 13.16).

The more complicated the model, the more urgent the need for selecting good starting values. Unfortunately, finding good starting values is particularly difficult in complicated models. The solution is to strip the model down to its basics, estimate those first individually, and build up gradually to complicated models. aML's capabilities for searching in multiple rounds over increasingly large numbers of parameters tend to be very helpful in this building process, as shown below. Also, auxiliary program `update` (explained in Sections 2.1.7 and 15.1) is a helpful utility in this process. In general, building up models is best done as follows.

1. Start with a single equation model and keep the residual structure as simple as possible (i.e., without heterogeneity). First find a good intercept value. Sometimes the mean outcome, as available from the data summary (`.sum`) file, may be of guidance, as explained in the sections below for individual models. If no good intercept is available, start with zero and estimate the equation with an intercept only.
2. Once a good intercept is found, free up some explanatory covariates.
3. Add residuals (heterogeneity) to the model, one at a time. Do not start standard deviations of residuals out at (close to) zero, as the search process tends to have trouble departing from zero. Instead, start standard deviations roughly at the standard deviation of the outcome. For heterogeneity in qualitative outcome models, standard deviation values in the 0.5 to 1.0 range tend to work well. It is often a good idea to first estimate the standard deviation while only allowing the intercept to vary.
4. When including additional regressors, start them at zero.
5. Once the single equation model with regressors and residual structure has converged, it may be joined with other single equation models to form a system of simultaneous equations. Specify the joint model such that its initial value is equivalent to the underlying single equation models which were independently estimated. For example, if the joint model introduces a correlation coefficient, initialize it at zero. If a latent outcome enters as explanatory factor in another equation, start its coefficient at zero. First estimate only the residual structure and all intercepts; free up more parameters in subsequent rounds.

6.1.1. Failure to Converge

It is not uncommon (and very frustrating) for complex multiprocess and multilevel models to not converge. This may be due to a variety of causes.

Some models do not converge because they are underidentified. For example, an attempt to estimate heterogeneity where no repeated outcomes are available will typically result in a standard deviation approaching zero; similarly, an attempt to estimate both an intercept and an exhaustive set of indicator variables (e.g., dummy variables for both male and female) will fail because of multicollinearity. Generally speaking, if the smallest eigenvalue of the matrix of second derivatives (which is written out every iteration) is close to zero, a likely cause is that the model is underidentified. Tinkering with starting values will not help in these cases; some restriction must be imposed, mostly dictated by the theoretical underpinnings of the model.

If an identified model fails to converge, try some other starting values and carefully follow the directions above and below.

If an identified model with good starting values does not converge, the culprit may be in a poor search direction. The search direction is based, in part, on the matrix of second derivatives of the log-likelihood with respect to model parameters (Hessian matrix). By default, this matrix is approximated using the BHHH method. For small samples, this approximation can be poor. Try specifying "`option numerical search`", which calculates the Hessian matrix more accurately. See page 270 for details.

If all else fails, try a grid search (Sections 6.2 and 13.16).

6.1.2. Order of Starting Values

All model parameters need to be initialized in the "`starting values`" statement. Each model parameter is part of a model building block: a regressor set, distribution, parameter, spline, et cetera. These building blocks are defined prior to their use in model statements. For example:

```
define regressor set BetaX; ...
define normal distribution; ...
define parameter Lambda;
define spline AgePattern; ...
```

Each definition introduces model parameters. These parameters need to be initialized in the order in which their corresponding building blocks were defined. In the above example, the regression parameters of regressor set `BetaX` need to be initialized first, followed by the standard deviations and correlations of the distribution, followed by the `Lambda` parameter, and finally followed by the slope parameters of the `AgePattern` spline.

Building blocks may imply multiple parameters. For example, a regressor set with four variables introduces four parameters, a bivariate distribution introduces two standard deviations

and one correlation, etc. Refer to the exact descriptions of building block definitions (Section 13.2) for the order in which these parameters need to be initialized:

Building block:	Page:
Parameter	284
Regressor set	286
Spline	290
Vector	295
Matrix	299
Normal distribution	301
ARMA(p,q) distribution	308
Cumulative AR(1) distribution	313
Finite mixture distribution	315

6.2. Grid Searches

Complex multiprocess and multilevel models may have non-concave likelihood surfaces and even local maxima. A grid search can help explore the likelihood surface and avoid convergence on a local maximum.

Grid searches may be specified as part of the "`starting values`" statement. Instead of initializing a parameter to a certain value, you may specify a lower bound, and upper bound, and the number of values to be tried. For example:

```
starting values;

<...>
Beta1     T     grid(-2, 10, 11)
<...>
;
```

aML will first evaluate the log-likelihood for 11 values of `Beta1`, equally spaced between –2 to 10 (namely –2, -.8, .4, 1.6, 2.8, 4, 5.2, 6.4, 7.6, 8.8, and 10). It will select the value with the highest log-likelihood and proceed with its usual Gauss-Newton search algorithm. More generally, the syntax is:

```
<parameter_name>     {T|F}     grid(a, b, n)
```

where `a` and `b` and lower and upper bound, respectively, and `n` is the number of grid points.

Grid searches may be conducted over multiple parameters, i.e., in a multidimensional space. For example,

```
starting values;

<...>
Beta1     T     grid(-2, 10, 11)
```

```
Beta2     T     grid(0, 5, 6)
<...>
;
```

results in a grid search over the (Beta1, Beta2) space between coordinates (-2, 0), (-2,5), (10,0), and (10, 5). There will be 11*6=66 function evaluations. There is no limit to the number of dimensions or the number of grid points. Naturally, the time required to conduct the grid search is linear in the number of points, which can increase rapidly with the number of dimensions.

Consider sample file Chapter6\grid.aml, based on the very simple probit model of high school attainment that was illustrated in Section 2.1. (Obviously, conducting a grid search on a simple probit model is unnecessary, but it will do to illustrate various features.) The control file is:

```
 1  dsn = ..\Chapter2\education.dat;
 2
 3  define regressor set BetaX;
 4     var = 1 female birth18 dadltHS dadcoll momltHS momcoll poorkid;
 5
 6  probit model;
 7     outcome = HSgrad;
 8     model = regset BetaX;
 9
10  starting values;
11
12  Constant    T    grid(-4, 4, 9)
13  female      T    0
14  birth18     T    0
15  dadltHS     T    0
16  dadcoll     T    0
17  momltHS     T    0
18  momcoll     T    0
19  poorkid     T    0
20  ;
```

aML will evaluate the log-likelihood for nine values of the Constant parameter, equally spaced between –4 and 4. By default, aML writes out grid search results to both the screen and the output file, in slightly different manners. In screen output, it adds asterisks for parameter values that improve the likelihood:

```
  Constant        ln-L
----------------------
      -4.0  -4061.1623 *
      -3.0  -2590.3354 *
      -2.0  -1484.8263 *
      -1.0  -735.32803 *
       0.0  -326.47232 *
       1.0  -213.16019 *
       2.0  -307.89262
       3.0  -522.53989
```

```
          4.0  -818.46043

Optimal grid point (ln-L = -213.1601913):

Constant         1.0
```

The best value is the one with the last asterisk, Constant=1. This value is used as starting value in subsequent model estimation. By default, output file `grid.out` contains similar information without improvement asterisks. Optionally, you may direct grid search results to another file:

```
option gridfile = filename;
```

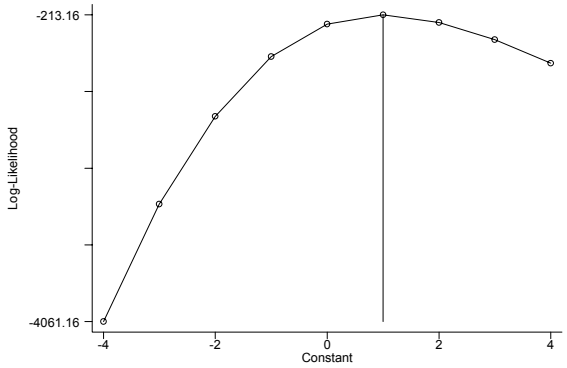
This may be useful for further analysis of the likelihood surface using a third-party software package. The file would contain the following:

```
   Constant        ln-L
 ----------------------
       -4.0  -4061.1623
       -3.0  -2590.3354
       -2.0  -1484.8263
       -1.0  -735.32803
        0.0  -326.47232
        1.0  -213.16019
        2.0  -307.89262
        3.0  -522.53989
        4.0  -818.46043
```

Users of Stata may find it convenient to write grid search results to a Stata dictionary file. If the optional gridfile name ends in `.dct`, aML writes the results in Stata dictionary format. For example, "`option gridfile=grid.dct`" results in the following `grid.dct` file:

```
dictionary {
   Constant
   lnl       "Log-Likelihood"
}
       -4.0  -4061.1623
       -3.0  -2590.3354
       -2.0  -1484.8263
       -1.0  -735.32803
        0.0  -326.47232
        1.0  -213.16019
        2.0  -307.89262
        3.0  -522.53989
        4.0  -818.46043
```

The results may then be conveniently read in and plotted:[26]



6.3. Probit Model

Without any regressors, the optimal intercept value is the inverse cumulative normal of the fraction successes:

$$\hat{\beta}_0 = \Phi^{-1}\left(\overline{Y}\right) \quad \text{where} \quad \overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i \,.$$

The fraction successes, $\overline{Y}$, is directly available from the data documentation (`.sum`) file. The inverse cumulative normal function, $\Phi^{-1}(\cdot)$, is built-in in most statistical packages. Start all regressors at zero and estimate the intercept and regressors before adding variance components (heterogeneity).

[26] This graph was produced by the following Stata commands:

```
infile using grid.dct
summarize lnl
summarize Constant if lnl==r(max)
global max=r(mean)
graph lnl Constant, xlabel xline($max) c(l)
```

6.4. Logit Model

Without any regressors, the optimal intercept is the logarithm of the odds ratio:

$$\hat{\beta}_0 = \log\left(\frac{\overline{Y}}{1-\overline{Y}}\right) \quad \text{where} \quad \overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i \,.$$

The fraction successes, $\overline{Y}$, is directly available from the data documentation (`.sum`) file. Start all regressors at zero and estimate the intercept and regressors before adding variance components (heterogeneity).

6.5. Continuous Model

The search procedure that aML applies is not particularly efficient at finding maximum likelihood parameter estimates of continuous models. The number of iterations required to achieve convergence tends to be larger than the number required for qualitative outcome models, and the search algorithm tends to step out relatively many times the search direction (which is based on a quadratic approximation). For example, it is not uncommon to achieve better likelihoods as far as eight or sixteen times the search direction. You may gain some efficiency by specifying "`option save step`" (page 276).

The best way to find good starting values is to run an Ordinary Least Squares (OLS) regression in your (SAS, Stata, SPSS) data preparation package. OLS yields the same coefficients on explanatory covariates as maximum likelihood estimation of a model with only one independently and identically distributed normal residual, but does so far more efficiently. Its estimate of the mean squared error is almost identical to the maximum likelihood estimate of the residual's variance. In the case of a multilevel continuous outcome model, use the OLS estimates as starting values for covariates, and distribute the mean squared error over the residuals in your model. For example, consider a continuous outcome model:

$$y_i = \beta' x_i + \varepsilon + u_i \,,$$

where *i* denotes a repeated outcome within an observation; the observation subscript has been suppressed. This is a two-level model. First estimate a simpler single-level model, without heterogeneity, using ordinary least squares:

$$y_i = \beta' x_i + v_i \,,$$

where all outcomes are treated independently. The estimated $\hat{\beta}$ provides good starting values for the multilevel model of interest. The estimated mean squared error provides the OLS estimate of the residual variance, σ_v^2. Distribute this variance over ε and u_i to get starting values for σ_ε^2 and σ_u^2. For example, without guidance from theory on their relative magnitudes, you could set:

$$\sigma_{\varepsilon}^2 = \tfrac{1}{2}\sigma_v^2 \text{, or } \sigma_{\varepsilon} = \tfrac{1}{2}\sigma_v\sqrt{2} \text{, and similarly } \sigma_u = \tfrac{1}{2}\sigma_v\sqrt{2} .$$

If there are more than two variance components, we recommend that you first estimate two, then three, et cetera. In the example of student test scores in Section 4.1.2 above, we distributed the OLS variance over four residuals and estimated their variances one-by-one in multiple rounds. Although only one variance was freed up from round to round, the model always had four residuals. If there would have been convergence problems, we would have first estimated a model with just two residuals; then included a third residual and let that model converge; and only then add a fourth residual.

Failure to converge is often caused by poor starting values and/or by too many parameters that need to settle in at the same time. A good approach is to return to a previous specification that did converge successfully, and free up only one or a small number of parameters.

If OLS starting values are not available, initialize the intercept to the mean of the outcome variable. This mean is directly available from the data documentation (`.sum`) file. (If the outcome is the logarithm of a data variable, `outcome=log(`*`varname`*`)`, you need to compute the average of the logarithms separately, because the mean logarithm is not equal to the logarithm of the mean.) The initial model specification should only include one residual that is drawn independently (`draw=_iid`) for each outcome. Its standard deviation should be initialized to the standard deviation of the outcome variable, also directly available in the data documentation file. Start all regressors at zero and estimate the intercept, regressors, and standard deviation of the residual. Then proceed with adding variance components (heterogeneity), as described above.

6.6. Hazard Model

As with all other model types, hazard models should be built up gradually. First, estimate a Gompertz hazard without covariates. The Gompertz hazard is linear in the log-hazard:

$$\ln h(t) = \gamma_0 + \gamma_1 t .$$

Baseline duration patterns in aML are always piecewise-linear (piecewise Gompertz); the Gompertz itself is a special case using a duration spline without nodes. Initialize γ_1 to zero; intercept γ_0 follows from

$$\gamma_0 = -\ln\left\{\frac{1}{N_{nc}}\sum_{i=1}^{N} t_i\right\},$$

where N is the number of spells, N_{nc} the number of noncensored spells, and t_i the duration of spell i. For censored spells, t_i is simply the duration from the beginning of the spell to the last

known end-point. For censored spells, however, aML requires that the user specify two duration variables which mark the lower and upper bound of the event window. For the purpose of initializing the intercept, such bounds very much complicate matters. We suggest that you simply set t_i equal to the midpoint of the event window. The durations should follow readily from your data preparation package. Alternatively, note that aML's output files provide summary statistics on spell durations, as shown below.

Once a Gompertz baseline hazard function has been estimated, specify about four nodes, spread out roughly evenly over the relevant spells range. Initialize the intercept at the first-stage estimated intercept, and all slopes at the first-stage estimate slope. Inspect the resulting pattern and remove nodes of which the surrounding slopes are roughly equal. Typically, two or three nodes are sufficient to adequately capture the baseline duration pattern. Experiment by shifting individual nodes to find a pattern that captures the essence of the pattern in the data. Finally, add key regressors and additional duration patterns, if any. Heterogeneity is best added last.

Consider an example of marriage durations, i.e., the hazard of divorce (Sections 2.5 and 3.5). Our data preparation package indicates (not shown) that the average duration of noncensored spells is 10.106 years, of censored spells 19.438 years. This may also be derived from summary statistics that aML writes out:

censor	#	Mean	Std Dev	Min	Max
/ lower	1243	8.862188	7.526053	0.06	50.075
0 - upper	1243	11.34998	9.796729	0.142	70.439
\ window	1243	2.487735	6.34943	0.006	46.954
1 - spell	2995	19.4383	15.48876	0.063	73.068

Note that the average of the event windows' lower and upper bounds is (8.862188+11.34998)/2 = 10.106. A good starting value for the intercept is therefore

$$-\ln\left\{\frac{1243*10.106+2995*19.438}{1243}\right\}=-4.042,$$

and the first step is to estimate a Gompertz duration dependency:

```
define spline DurMar; nodes = ;

define regressor set Getdiv; var = 1;

hazard model;
   censor=censor; duration=lower upper;
   model = durspline(origin=0, ref=DurMar) +
      regset Getdiv;

starting values;
```

```
durslope    T      0
Constant    T     -4.042
;
```

We update the starting values with converged values and add four nodes at locations that we guess may be relevant:

```
define spline DurMar; nodes = 2 6 10 20;

define regressor set Getdiv; var = 1;

hazard model;
   censor=censor; duration=lower upper;
   model = durspline(origin=0, ref=DurMar) +
      regset Getdiv;

starting values;

dur0-2      T   -.0427148839
dur2-6      T   -.0427148839
dur6-10     T   -.0427148839
dur10-20    T   -.0427148839
dur20+      T   -.0427148839
Constant    T   -3.6518258127
;
```

We find that the hazard of divorce increases steeply during the first two years of marriage, is almost flat for the next four years, declines for the next four years, remains flat for another ten years, and then declines. If two slopes on contiguous segments had been about the same, we would have eliminated the node separating those segments. Here, we decide to explore the steep increase in the first two years by shifting the first node to 1 and moving the second to 4 years. The increase appears mostly due to the first year. We experiment some with the outer nodes and settle on nodes at 1, 4, 15, and 25 years.[27] Updating the slopes with converged values, we add covariates of interest:

```
define spline DurMar; nodes = 1 4 15 25;

define regressor set Getdiv;
   var = 1 (marnum==2) (marnum>=3) heblack (hiseduc<12)
         (hiseduc>=16) (agediff>10) (agediff<-10)
         (heblack!=sheblack) numkids;
```

[27] There are no hard rules on choosing nodes. We could have left the nodes at 2, 6, 10, and 20 years with virtually the same estimates of covariate parameters of interest. Try it out for yourself using `divorce3.dat` in the `Samples\Section3` subdirectory.

```
hazard model;
   censor=censor; duration=lower upper; timemarks=time;
   model = durspline(origin=0, ref=DurMar) +
      regset Getdiv;

starting values;

dur0-1      T    1.7704887806
dur1-4      T     .0734946166
dur4-15     T    -.0492449675
dur15-25    T     -.028029827
dur25+      T    -.1320915794
Constant    T   -5.6549655723
mar2        T    0
mar3+       T    0
heblack     T    0
dropout     T    0
college     T    0
heolder     T    0
sheolder    T    0
mixrace     T    0
numkids     T    0
;
```

As the last step, we add heterogeneity. Note that we initialize the standard deviation of the heterogeneity component to 0.6 (which tends to be in the right ballpark), and that we first allow all other coefficients to settle in before freeing up this standard deviation:

```
define spline DurMar; nodes = 1 4 15 25;

define regressor set Getdiv;
   var = 1 (marnum==2) (marnum>=3) heblack (hiseduc<12)
         (hiseduc>=16) (agediff>10) (agediff<-10)
         (heblack!=sheblack) numkids;

define normal distribution; dim=1;
   number of integration points = 4;
   name= delta;

hazard model;
   censor=censor; duration=lower upper; timemarks=time;
   model = durspline(origin=0, ref=DurMar) +
      regset Getdiv +
      intres(draw=1, ref=delta);
```

```
starting values;

dur0-1      TT    1.8140846191
dur1-4      TT     .0923564046
dur4-15     TT    -.0388807972
dur15-25    TT    -.0229186888
dur25+      TT    -.1289813362
Constant    TT   -5.6044539064
mar2        TT     .2384592885
mar3+       TT      .603922662
heblack     TT      .010035551
dropout     TT    -.3175157405
college     TT    -.3050647156
heolder     TT    -.5203170435
sheolder    TT     .1993272706
mixrace     TT     .3614972652
numkids     TT    -.0791978408
SigDelta    FT     .6;
```

While the steps leading up to this final specification may appear laborious, you will find that they actually save time (not to mention frustration!) Good starting values are critical to the successful estimation of complicated models.

6.7. Binomial Model

The implementation of binomial models in aML offers three alternatives to specify the probability of a success. The simplest form is a direct specification. Its syntax is:

```
prob = linear(...);
```

where the argument to `linear(...)` may be any linear combination of parameters, regressors sets, and integrated residuals. (It is the user's responsibility to ensure that the probability is between zero and one.) The intercept follows directly from the data summary (`.sum`) file: it is the ratio between the means of the outcome and exposure variables. Start regressors at zero, and add residuals last.

More commonly, the probability is a logistic or probit (cumulative normal) transformation of parameters, regressor sets, and integrated residuals. The syntax for the logistic transformation is:

```
prob = logistic(...);
```

where the argument to `logistic(...)` may be any linear combination of parameters, regressors sets, and integrated residuals. Denote the argument by $\beta' X + \varepsilon$; the transformation is given by:

$$p = \frac{1}{1+\exp(-\beta' X - \varepsilon)}.$$

Note that this transformation guarantees $0 < p < 1$. In the absence of regressors and integrated residuals, the intercept is analogous to the log-odds ratio in logit models:

$$\hat{\beta}_0 = \log\left(\frac{f}{1-f}\right) \text{ where } f = \frac{\overline{Y}}{\overline{E}}$$

and $\overline{Y}$ and $\overline{E}$ denote the mean outcome and exposure, as reported in the data documentation (`.sum`) file. Add regressors and subsequently integrated residuals, if any. Similarly, the syntax for the probit (or cumulative normal) transformation is:

```
prob = normprob(...);
```

where the argument to `normprob(...)` may be any linear combination of parameters, regressors sets, and integrated residuals. Denote the argument by $\beta' X + \varepsilon$; the transformation is given by:

$$p = \Phi\left(\beta' X + \varepsilon\right).$$

Note that this transformation guarantees $0 < p < 1$. In the absence of regressors and integrated residuals, the intercept is

$$\hat{\beta}_0 = \Phi^{-1}(f),$$

with *f* defined as the ratio of the mean outcome and exposure, as above. Add regressors and subsequently integrated residuals, if any.

6.8. Poisson Model

Poisson models tend to be sensitive to good starting values. The likelihood surface is very flat around poor starting values and the search algorithm may not find its way out of the flat region.

In the absence of varying exposure across observations, the optimal intercept of the Poisson incidence is simply the mean of the count outcome variable Y_i, as found in the data documentation (`.sum`) file. aML requires specifying the incidence as:

```
incidence = exp(...);
```

so that the optimal intercept is actually the logarithm of the mean of the outcome, $\beta_0 = \log\left(\overline{Y}\right)$.

If an exposure variable E_i is included in the model specification, the optimal intercept of the log-incidence is the difference between the logarithms of the outcome and the exposure variables: $\beta_0 = \log\left(\overline{Y}\right) - \log\left(\overline{E}\right)$.

With this intercept, first estimate regressors (initialized at zero), then add heterogeneity, if any.

The default search direction, based on the BHHH approximation to the matrix of second derivatives, is sometimes fairly poor in Poisson models, and particularly so with small sample sizes. Consider the "`option numerical search`" (pages 64 and 202; Section 13.1.4).

6.9. Negative Binomial Model

Negative binomial models are somewhat more complicated, because there are two equations: one for the dispersion or log-dispersion and one for the log-incidence. We suggest initializing the dispersion to one (or the intercept of the log-dispersion to zero). Similarly, initialize the intercept of the log-incidence to zero. Let the dispersion and incidence intercepts settle in before adding covariates. Heterogeneity in the incidence equation, if any, should be added last.

The default search direction, based on the BHHH approximation to the matrix of second derivatives, is sometimes fairly poor in negative binomial models, and particularly so with small sample sizes. Consider the "`option numerical search`" (pages 64 and 202; Section 13.1.4).

6.10. Multinomial Logit Model

A multinomial logit model with n potential outcomes requires n-1 model equations. The intercepts in those equations are simply the logarithm of odds ratios (relative frequencies of the outcomes).

Consider the occupational choice model of Section 2.10. The occupational choice outcome is distributed as follows:

```
        occ |      Freq.     Percent
------------+-----------------------
          1 |       569       28.45
          2 |       531       26.55
          3 |       344       17.20
          4 |       556       27.80
------------+-----------------------
      Total |      2000      100.00
```

Suppose we specify model equations for choices 1, 2, and 3, all relative to omitted category 4. The intercepts of those models are $\ln(569/556)$, $\ln(531/556)$, and $\ln(344/556)$, respectively.

The multinomial logit model tends to be very well-behaved and robust to poor starting values. Except with exceptionally large data sets, it is probably not worth your while to figure out the optimal intercepts. You could initialize intercepts to zero and first let aML estimate them, then add regressors, and finally heterogeneity, if any.

6.11. Multinomial Probit Model

A multinomial probit model with *n* potential outcomes requires *n*-1 model equations. Each model equation may have an intercept and each must have a residual from a normal distribution with dimension *n*-1.

The standard deviations of the n-1 residuals are not identified. Initialize them to one and do not estimate them. Initialize the $(n-1)(n-2)/2$ correlations to zero and the *n*-1 intercepts also to zero. First estimate the intercepts only; then free up the regressors; then free up correlations; and introduce heterogeneity, if any, last.

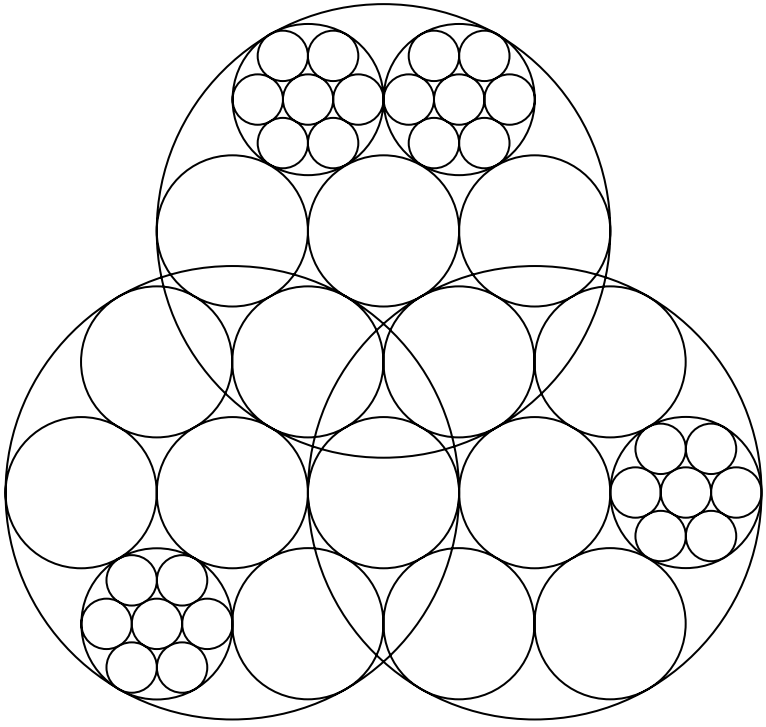
6.12. Multiprocess Models

Equations in multiprocess models should always first be estimated individually following the steps outlined above for single equation models. Once the intercept, key covariates, and (univariate) heterogeneity have been estimated, the equations may be combined one-by-one. This typically requires that correlation between residuals in multiple equations is introduced, i.e., multiple univariate distributions are combined into multivariate distributions. Initialize the standard deviations at their converged values and correlation coefficients at zero. A good check on the correct specification of a joint model is that the log-likelihood of the joint model in the first iteration is equal to the sum of the converged log-likelihoods of single equation models.

Similarly, if the (latent) outcome of one equation enters the equation of another model as explanatory covariate, initialize its coefficient to zero, so that the initial model is identical to the combination of two single equation models.

At first, only estimate the residual structure and the intercepts; free up other covariates later. If the model fails to converge, back up and free up alternative subsets of parameters before attempting the fully joint model. Pay particular attention to potentially endogenous variables, as their coefficients are most likely to change substantially. After the intercepts and residual structure have settled in, you may want to also free up potentially endogenous variables before estimating all parameters.

Multilevel Multiprocess Modeling



aML

REFERENCE MANUAL

7. Roadmap

As explained in the User's Guide, the main stages of using the aML package are data preprocessing, model specification and estimation, and results interpretation. This Reference Manual is organized in roughly the same order. It contains the following chapters.

This chapter documents four little programs which make the aML user's life a little easier: update, mktab, amltest, and points.

16. Miscellaneous Features 435

This section documents miscellaneous features in the aML package, including its macro language capabilities.

8. Raw2aml Command Line Options

Raw2aml features many options. Some may be specified on the command line, i.e., in the DOS or UNIX window, when you invoke the raw2aml executable program; some may be specified in the raw2aml control file, i.e., the `.r2a` file; and some may be specified either way. This chapter documents command line options; Chapter 9 deals with control file options.

Raw2aml may be invoked with the following command line options:

`raw2aml [-h] [-r] [-o file.dat] [-m macrofile] file.r2a`

Command line options may appear in any order. The remainder of this chapter explains how each option alters raw2aml's default behavior.

Option -h

Option `-h` ("help") provides limited on-line help. It lists all supported command line options and concisely explains their use.

Option -r

By default, raw2aml checks whether the specified output data file already exists on disk. If it indeed already exists, it asks the user for permission to overwrite. Option `-r` ("replace") causes raw2aml to overwrite the output data file without confirming that this is OK.

Option -o file.dat

Option `-o` ("output") definitively specifies the output data file. There are three ways to specify the name of the output data file. Raw2aml processes them in the following order:

- Command line option `-o` *`file.dat`*. If this command line option is specified, raw2aml will create the specified data file. (If no extension is specified, raw2aml will add `.dat`; see below.)
- If command line option `-o` is not specified, raw2aml parses the raw2aml control file for "`option output data file =` *`file.dat`*". If that control file option is specified, raw2aml will create the specified data file. (If no extension is specified, raw2aml will add `.dat`; see below.)
- If the output data set name is neither specified on the command line nor in the control file, raw2aml applies its default naming convention based on the name of the control file. By default, the output data file name is equal to the control file name,

where extension `.r2a` is replaced by `.dat`. If the control file name does not contain extension `.r2a`, extension `.dat` is appended. For example, if the control file is "`abcdef.r2a`", the output data file will be "`abcdef.dat`"; if the control file is "`abcdef`", the output data file will be "`abcdef.dat`"; if the control file is "`abcdef.ctl`", the output data file will be "`abcdef.ctl.dat`".

Raw2aml likes output data file names to have the `.dat` extension. If you specify an output data file name, whether on the command line or in the control file, which does not contain an extension, raw2aml will append `.dat` to that name. However, if you do specify an extension—any extension—, raw2aml will respect that choice and name the output data file accordingly. For example, if you specify "`abcdef`", raw2aml will create output data file "`abcdef.dat`"; if you specify "`abcdef.data`", raw2aml will create output data file "`abcdef.data`"; if you specify "`abcdef.`" (with an explicit dot but no extension), raw2aml will create output data file "`abcdef`". And, of course, "`abcdef.dat`" is taken literally.

In addition to converting ASCII input data into an output data file, raw2aml creates a file with summary statistics of the data. This summary file may be read using any text editor. Its name is derived from the name of the output data file. If the output data file has the conventional `.dat` extension, the summary file name will be identical but with extension `.sum` instead of `.dat`. If the output data file name does not have a `.dat` extension, `.sum` is appended to its name. For example, the summary file for "`abcdef.dat`" will be "`abcdef.sum`"; for "`abcdef`" it will be "`abcdef.sum`"; and for "`abcdef.data`" it will be "`abcdef.data.sum`".

We strongly recommend that you follow our convention of naming output data files and summary files with `.dat` and `.sum` extensions, respectively. It greatly helps in keeping your files organized.

Option -m *macrofile*

Raw2aml control files may contain user comments which raw2aml should ignore (Section 16.1). This is implemented through a control file pre-processor which strips out all user comments. By default, the stripped-down control file is saved only temporarily to disk, and is removed as soon as raw2aml has finished parsed the control file statements. Option `-m` causes raw2aml to save the stripped-down control file permanently to disk, so that you may look at its contents. Its name is the user-specified *`macrofile`*.

There is no reason why you would ever want to look at a version of your control file without comments. A more useful purpose of the `-m` option is to debug macros that you may embed in raw2aml control files. The raw2aml control file pre-processor namely supports a simple macro language (Section 16.2). The stripped-down control file has all macros resolved; it is what raw2aml actually parses. Should raw2aml protest at your use

of macros, then this will provide a way to determine how your macros were resolved. It is a particularly useful option when using nested macros, which may get tedious.

Argument *file.r2a*

The last command line argument, *file.r2a*, is not optional. It specifies the name of the raw2aml control file. We wrote `file.r2a`, with extension `.r2a`, to remind you of the convention to name control files with a `.r2a` extension. However, you may specify any name.

If you do not specify an extension, raw2aml will first look for a control file with extension `.r2a`. If such a file does not exist on disk, it will look for the file as you specified it. For example, if you specify argument "`abcdef`", raw2aml will first attempt to open "`abcdef.r2a`"; if such a file does not exist, it will look for "`abcdef`". If you specified "`abcdef.ctl`", raw2aml will first look for "`abcdef.ctl.r2a`", then for "`abcdef.ctl`".

9. Raw2aml Control File Statements

The raw2aml control file consists of two parts. The first part, described in Section 9.1, specifies global control features such as conversion options and input and output file names. The second part, documented in Section 9.2, specifies the structure of the data that need to be converted. This includes labels of data structures, their level structure, and variable names.

9.1. Global Control

The general part of the `raw2aml` control file specifies general options and the names and locations of input and output files. The syntax is:

```
[option observations = {n|all};]
[option irrelevance check ={yes|no|n};]
[option deviations;]
[option crossmeans = no;]

ascii data files = filename [filename ... filename];
[id file = filename;]
[option output data file = filename [(replace)];]
```

We now describe each statement in detail.

`option observations = {n|all};`

This option restricts the number of observations that raw2aml will convert. It is sometimes useful for testing purposes. The default is `all`, which is understood to be all observations. If you specify a numeric value, say, "`option obs=100`", only the first 100 observations in the ASCII input file(s) will be converted and written out to the output data file. If the data contain fewer observations than the number specified for conversion, then all observations are converted and the option has no effect.

`option irrelevance check = {yes|no|n};`

As explained in Section 3.3, ASCII data may be written in either 'compressed' or 'rectangular' form. Suppose the maximum number of level 3 branches is 10; many records will have fewer. In SAS it is possible to output only relevant branches by looping over only those branches. In Stata, however, there is no possibility to loop over an observation-dependent number. All 10 subbranches must be output, even though some of them are irrelevant.

The "`irrelevance check`" option provides a data integrity check for rectangular data forms. The user should set irrelevant variables to a reserved integer number such as

-99. The program will then check that irrelevant variables are equal to -99. If any irrelevant variable is not -99, the program aborts with an error message. If any *relevant* variable is equal to -99, raw2aml issues a warning.

The default is to check that irrelevant variables have been set to -99. Instead, "`irrelevance check=no`" does not check anything, and "`irrelevance check=`*n*" checks that irrelevant variables have been set to a user-specified integer, *n*. If you like, you may explicitly state the default as "`option irrelevance check=-99`" or "`option irrelevance check=yes`" or just "`option irrelevance check`".

The irrelevance check only applies to rectangular data. In data with three levels, only level 3 variables will be checked; in data with four levels, both level 3 and 4 variables are checked; et cetera. The special value to which irrelevant variables are set must be the same for all data structures.

We strongly recommend that you allow raw2aml to check the values of irrelevant data items. It provides good protection against unintended misalignment of variables. Also, we recommend that you scan the summary statistics that raw2aml generates in the summary (.sum) file. A minimum or maximum that is equal to, say, -99, may be indicative of incorrect ordering of variables.

`option deviations;`

By default, the output data will be in absolute value, but optionally, some or all variables may be converted into deviations from their means. This may under some circumstances improve search stability, especially when new regressors are added to a model specification.

If "`option deviations`" is specified, all data variables will in principle be converted into deviations from their mean. Control variables such as IDs and data structure numbers always remain in absolute value. However, variables that are listed in any variable list (see below) may remain in absolute value by attaching "`(a)`" to their names. This will be essential for variables that are used as weights, outcomes, duration spline origins, et cetera. Conversely, any variable with "`(d)`" attached to its name will be converted into deviations, regardless of the deviations option. In other words, the deviations option sets the default for all variables. This default is overridden by attaching either `(a)` ("absolute") or `(d)` ("deviations") to variable names.

Consider the following excerpt of a raw2aml control file:

```
option deviations;
<other statements...>
var = weight(a) censor(a) age income educ;
```

All data variables in the ASCII input file will be converted into deviations from their means, except variables `weight` and `censor` which remain untransformed.

When a variable appears in multiple data structures, the user may want the mean to be computed over all data structures, or he may want the mean to be specific to the current data structure. See "`option crossmeans`".

You may transform variables into deviations from any real number. For example, suppose your data contain variable `year` which represents the calendar year (1980, 1981, et cetera) to which the record refers. You decide that you would like, say, 1990, to be the reference year in your models. This is facilitated by writing:

```
var = ... time(d=1990) ...;
```

The output data set will contain `time` in deviations from 1990, rather than in absolute value or in deviations from its mean. Note that deducting zero is equivalent to keeping a variable in absolute value, so that, for example, "`censor(d=0)`" yields the same result as "`censor(a)`".

`option crossmeans = no;`

It is often convenient to split data into separate data structures. If deviations from means are taken, either because of "`option deviations`" or because "`(d)`" is attached to a variable name, the means may be computed within data structure or over all data structures in which the variable appears. The default is to compute the mean of a variable over all data structures in which that variable appears; "`option crossmeans=no`" computes means separately by data structure.

For example, suppose data for males and females are split into separate data structures. If the analysis will be conducted entirely separately for males and females, you may want raw2aml to compute means separately for males and females. However, if males and females are pooled, the overall mean should be computed. Separate means would namely absorb some of the sex intercept difference. This is most clearly illustrated with the variable for sex, say, `male`, which is always zero in the female data structure and one in the male data structure. If deviations from means were taken, where the means are computed within each data structure, then the `male` variable would always end up being zero. For females, `male` is always zero, so its mean is zero, so `male` in deviations from its mean is always zero; for males, `male` is always one, so its mean is one, so `male` in deviations from one is also always zero. In other words, deviations from means within data structures reduces some (and here all) of the variation in the variable across data structures.

It is not possible to use overall means for some variables and within means for others: the crossmeans option applies to all variables. Should this feature be undesired, then the user can compute means in whichever way he wants, and attach that mean to certain variable names, e.g., `age(d=14)`. See the description of "`option deviations`".

ASCII data files = *filename* [*filename* ... *filename*];

The "ASCII data file" statement is not optional. It specifies the name and location of the ASCII input file(s). Single or double quotes are optional. However, if the pathname or filename contains blank spaces, the entire name must be enclosed by quotes.

Input data may be spread over multiple ASCII files. The "ASCII data files" statement lists these files. Their order is irrelevant, unless no ID file is specified. In that case, the first ASCII data file must contain all unique IDs. Also see the optional "id file" statement, next.

id file = *filename*;

There may be multiple ASCII input data files, each providing information on a piece of the data or model. For example, suppose you are interested in jointly modeling mortality, divorce, and income. You could put all corresponding data into one ASCII file, or you could organize your data such that one ASCII file contains data on mortality spells, another on divorce spells, and yet another on income records. Raw2aml needs to merge data from the three files so that all information related to one observation (person, in this case) is stored together. Each data structure record therefore needs to have an ID that is common to all records relating to one observation. IDs may be any integer value, including negative and zero.

The ID file is a master file that contains all IDs, and nothing else. Each ID needs to be on a separate line. Raw2aml reads in an ID from this ID file. It then reads in a record from the first ASCII file and checks whether the ID matches. If it does, this record is stored and becomes part of the current observation; raw2aml will then attempt the next record of the same ASCII file, until an ID does not match anymore. It then moves on to the second ASCII file and attempts to read all consecutive records that match the current ID. It then goes to the third ASCII file, et cetera. When all ASCII files have been read, the observation is complete and the next ID will be read from the ID file. It may be that an ID from the ID file cannot be matched to any ASCII file; it is then simply ignored. However, if an ID from an ASCII file does not appear in the ID file, an error message is generated. The number of observations in the output file is equal to the number of IDs that could be matched to at least one data structured.

If one of the ASCII files contains a record with an ID that does not appear in the ID file (or first ASCII file, if no ID file is specified), then this record cannot be matched to any observation. When raw2aml has completed reading the ID file (or first ASCII file), it checks that all other ASCII files have been read entirely. If this is not the case, raw2aml will abort with an error message.

Consider again the example of three ASCII files with data on individuals' mortality spells, divorce spells and income records. If the three ASCII files are specified in that order (`ascii data files = filename ... filename`), there will be no problem, because every person must have mortality spell data. However, if the divorce spell file is listed first, raw2aml will not find the IDs of individuals that were never married. It will thus not read in those individuals' mortality or income data, and issue an error message.

If there is only one ASCII file, there is no need for an ID file. Even if there are multiple ASCII files, no ID file may be needed, provided that all IDs appear in the ASCII file that is listed first. In other words, if the control file does not specify an ID file, the first ASCII file will serve as ID file. The program will read in all consecutive records from the first ASCII file with the same ID, and then reads as many records from other ASCII files that can be matched to the same ID. This requires that the first ASCII file contains all IDs, just like the ID file must contain all IDs.

Strictly speaking, ID files and ASCII data files need not be sorted by increasing ID. However, the ID file and ASCII files must be sorted in the same ID order. Also, all identical IDs must appear in consecutive records. For example, the IDs of records may appear as 2 5 5 5 8 8 3 3, in which case there are four observations. The sequence 2 3 5 8 8 5 3 5 would be interpreted as seven observations. Raw2aml issues a warning if IDs are not sorted in increasing order (which is OK), but does not keep track of non-contiguous duplicate IDs (which is not OK). We recommend that you sort your data by ID, so that the ID and ASCII files are also sorted.

`option output data file = filename [(replace)];`

This statement optionally specifies the output data set. By default, raw2aml derives the name of the output data file from the control file by substituting extension "`.dat`" for "`.r2a`". The user may override this default by specifying the "`option output data file`" option in the control file or by the "`-o`" option on the command line. The latter has highest priority; see Chapter 8.

If the output data file already exists, it will not be overwritten unless "`(replace)`" is also specified. The default may be explicitly stated, if you like, by "`(replace=no)`".

If `filename` is specified without extension, raw2aml adds extension "`.dat`". If the user specifies any extension, the name is taken as-is. (aML also assumes default extension "`.dat`"; it need not be specified in aML control files.)

Raw2aml will create a data summary file along with the output data set. Raw2aml derives its name from the output data file name by substituting extension "`.sum`" for "`.dat`". If the output data file name does not have extension "`.dat`", extension "`.sum`" is appended to its name. Summary files provide information on the maximum number of data structure records (level 2 branches) in any one observation, the maximum number of level 3 subbranches in any one level 2 branch, et cetera, as well as summary statistics

(mean, standard deviation, minimum, maximum) on all variables. Always check that the number of observations is equal to what was expected and check the summary statistics on variables.

9.2. Control over Data and Data Structures

The second part of the raw2aml control file specifies how the data are organized, including the number of levels and variable names and whether they are distinguished into so-called data structures. The syntax depends on the complexity of the data, particularly the number of levels and the organization of data into data structures. aML supports multilevel models, and the second part of the raw2aml control file lists the variables at all levels. aML also supports multiprocess models. It is often convenient to organize one's data in so-called data structures, particularly when different processes involve different variable lists.

Definition: Data Structure

A data structure is a subset of variables that are or need to be distinguished from other subsets of variables.

The concept of a data structure may be best illustrated by some examples. Suppose one is interested in analyzing children's educational attainment, joint with parental wage income. In the data, level 1 may correspond to a family and level 2 to a person (father, mother, children). (Note that level 1, the highest level, corresponds to the greatest level of aggregation, with lower levels corresponding to disaggregated, repeating data.) We may model children's educational attainment as a schooling progression model, with a probit for the completion f each grade. Level 3 for children may then denote grade levels. Level 3 for parents may correspond to jobs, with level 4 containing annual wage reports. With such asymmetric data, it makes no sense to include the same variables at all levels. Instead, one would define one 4-level data structure with parents, their jobs, and wages, and another 3-level data structure with children and their grade levels. The data structures are different in their number of levels and their variable lists.

You may also define data structures for no more compelling reason than convenience. For example, suppose you wish to estimate a model separately for men and women. The data fit perfectly well into a single data structure, because there are no differences in levels or variable lists. (The model statements will need a keep or drop statement so that they apply to only men or only women.) However, if you find it convenient, you may assign men to one and women to another data structure, and specify models in aML accordingly.

The general syntax for the second part of the raw2aml control file is:

```
[level 1] variables = varlist;

[data structure = n;]
   [level 2 variables = varlist;]
   [level 3 variables [(nb=n)] = varlist;]
   [level 4 variables [(nb=n)] = varlist;]
```

<et cetera>

Indeed, most statements are optional in the sense that they need not always appear. Depending on the complexity of the data, however, some statements are mandatory. We illustrate the syntax by increasing data complexity: (1) single level, single data structure; (2) multiple levels, one data structure; and (3) multiple levels, multiple data structures. We start with the simplest case.

9.2.1. Single Level, Single Data Structure

[level 1] variables = *varlist;*

The only statement that is needed provides the list of variable names. You may explicitly state that this is a list of level 1 variables (`level 1 var = varlist`), or, since there is no room for confusion, omit the level 1 specification (`var = varlist`). The variable list, *varlist*, is simply a list of variable names.

Variable names may be up to eight characters in length, must begin with an alpha character (a-z, A-Z), and may only contain alpha characters, numeric characters, and underscores ("_"). Variable names are case-sensitive, just like all other user-defined names.

You may assign any names to your variables. Indeed, there need not be any relationship to the names of the variables in your data processing package (SAS, Stata, SPSS, or other). However, we recommend that you assign intuitive names that are similar to the names in your data processing package, so that the potential for confusion is minimal.

If desired, raw2aml transforms one or more variables into deviations from their mean or from any other value. There are four ways to achieve this (also see "`option deviations`", page 225.)

- If "`option deviations=yes`" is specified (page 225), all data variables are transformed in deviations from their mean. By default, the mean over all data structures in which a variable appears is deducted. "`Option crossmeans=no`" (page 226), causes raw2aml to deduct the variable's mean as calculated within data structures.
- *varname*`(d)` deducts the mean of *varname* from the variable before writing it out to the aML-formatted output file, regardless of the "`deviations`" option. The mean over all data structures is deducted, unless "`option crossmeans=no`" is specified.
- *varname*`(a)` retains the absolute value of *varname*, regardless of the "`deviations`" option.

- *varname*(d=*x*), where *x* is a real number, causes raw2aml to deduct *x* from *varname*. For example, age(d=14) will be converted into age minus 14, regardless of whether the "deviations" option is specified.

Transforming variables into deviations from their mean is sometimes helpful to ensure smooth search paths during maximum likelihood model optimization in aML. This mostly applies to explanatory covariates and continuous outcome variables. *Do not take deviations of qualitative outcome variables, as their transformed values would be meaningless!* More generally, refrain from transforming variables into deviations from their mean unless you are very experienced in maximum likelihood search procedures.

9.2.2. Multiple Levels, Single Data Structure

```
level 1 variables = varlist;
level 2 variables = varlist;
level 3 variables [(nb=n)] = varlist;
level 4 variables [(nb=n)] = varlist;
<et cetera>
```

The syntax for data with multiple levels tells raw2aml how many levels to expect, and what the names of variables at each level are. In addition, the optional [nb=*n*] specifications tell raw2aml how many subbranches to expect at levels 3 and lower. This option is only relevant for ASCII data that are written out in "rectangular" format, as explained in Chapter 10.

It is worthwhile restating aML's numbering of levels here:

aML uses the convention that data at the observation level, i.e., at the highest level of aggregation, are level 1 data. Levels 2 and lower are for disaggregated data for which multiple measures are available.

For example, in an analysis of student test scores, school-level variables (location, private/public) may be level 1 variables; student-level variables (sex, birth year) are level 2 variables; academic year-level variables (grade, pass/retain) are level 3 variables; and test-level variables (subject area, score) are level 4 variables. Note that some other multilevel software packages use the reverse numbering scheme. The choice has no substantive consequences. In fact, you never use level numbers in the model specification stage, just variable names.

9.2.3. Multiple Levels, Multiple Data Structures

```
level 1 variables = varlist;

data structure = n;
   level 2 variables = varlist;
   level 3 variables [(nb=n)] = varlist;
   level 4 variables [(nb=n)] = varlist;
   <et cetera>

data structure = n;
   level 2 variables = varlist;
   level 3 variables [(nb=n)] = varlist;
   level 4 variables [(nb=n)] = varlist;
   <et cetera>

data structure = n;
<et cetera>
```

For data with multiple data structures, you need to tell raw2aml how to distinguish records from the various data structures. This is done by assigning a unique identifier to each data structure (`data structure=`*n*). These data structure numbers must be strictly positive integers. For specification of variable lists at each level, the syntax is very similar to the case with just a single data structure. Note that level 1 variables are shared by all data structures. This is only logical: level 1 is the highest level and contains observation-specific information. By definition, there is only one such set of information. One might thus say that data structures are defined at level 2, i.e., all data structures share the same level 1 variables and may differ only at level 2 and lower.

10. ASCII Input Data

The data that raw2aml converts are stored in ASCII input files. You create these files using your favorite data management package (SAS, Stata, SPSS, or other). This chapter documents how the ASCII input data files need to be structured. In other words, what is the sequence in which IDs, data structure numbers, other control variables, and data variables are written out to ASCII data files?

The ASCII data format depends on the number of levels in the data, on whether subsets of variables are distinguished into multiple data structures, and on whether the data are written in "rectangular" or "compressed" format. The data must be structured as follows:

```
id [struc] [nb]
    x1
   [x2]
   [x3]
   [x4]
   <et cetera>
```

where `id` is the observation's ID; `struc` is an optional data structure number; `nb` denotes one or more numbers of subbranches; `x1` is a vector of level 1 variables; `x2` is a vector of level 2 variables; `x3` denotes one or more vectors of level 3 variables; `x4` denotes one or more vectors of level 4 variables; et cetera. The following sections document the sequence of ASCII input files in increasing order of data complexity:

10.1. Single Level, Single Data Structure

Data with a single level and no distinct data structures are very straightforward. Just write out an observation ID and the list of variables. None of the optional items applies, so that the data must be structured as follows:

```
id x1
```

Note that `x1` is shorthand for the vector of level 1 variables, i.e., it represents potentially many variables. The only thing that requires a little attention is the ID variable. With single-level data, you might think that raw2aml does not need to pull records together into observations, and thus does not need an ID. To keep the requirements consistent with more complex data, though, raw2aml does require an ID preceding every record. If your data do not already contain an ID, you will need to create one. Setting the ID equal to the observation number will do just fine.

The data may span multiple lines. From the control file, raw2aml knows how many variables to read for each observation, and it will continue reading as many lines per observation as needed. However, each observation must start on a new line.

Data from multiple observations must be on separate physical lines in ASCII input data. In other words, IDs must always be the first variable on a line.

To further clarify this, suppose your data contain just three variables, `age`, `sex`, and `income`, plus an ID variable, `id`. The first three observations in your data are:

Observation	id	age	sex	income
1	1	25	1	20000
2	2	54	2	45000
3	3	37	2	50000

The raw2aml control file reads as follows:

```
ASCII input file = filename;
var = age sex income;
```

In the ASCII file that you create, the ID variable must be the first variable on a line. For example:

```
1 25 1 20000
2 54 2 45000
3 37 2 50000
```

In the raw2aml control file you must *not* list the ID variable. Only data variables are listed in the raw2aml control file. Control variables, such as ID variables, data structure numbers, and numbers of subbranches, discussed below, are not specified. Raw2aml knows they should be there.

The variables pertaining to a particular ID may wrap over multiple lines. Furthermore, variables may be separated by one or more spaces, tabs, or commas.[28] For example:

```
1   25
1 20000
2
54
2
45000
   3,37,2
    50000
```

It is not immediately obvious to the human eye which numbers belong to which ID, but raw2aml knows from the control file to expect an ID plus three variables.

Data fields in ASCII files may be delimited by spaces, commas, or tab characters.

Data fields that belong to one logical record may wrap over multiple lines (multiple physical records), provided that the data structure number (if any) is on the same line as the ID.

A new logical record must always start on a new line, i.e., IDs must always be the first number on a line.

[28] If you are using Stata to prepare data and create ASCII files, you may want to specify the "`comma`" option to the "`outfile`" command. This results in smaller-sized ASCII data. You may also want to specify the "`wide`" option which prevents wrapping over multiple lines. The latter is for convenience only, in case you wish to visually inspect the ASCII file.

10.2. Two Levels, Single Data Structure

The general order for data with two levels without distinct data structures is:

```
id x1 x2
```

Note that x1 and x2 represent vectors of level 1 and level 2 variables, respectively. Each level 2 "branch" is written out to a separate ASCII record. For example, suppose your data contain individuals' annual earnings for multiple years. You have data on the person's sex and education and, or multiple years, earnings amount, age, and 5-category health indicator. The person-specific (level 1) variables are `sex` and `educ`; year-specific (level 2) variables are `earnings`, `age`, and `health`. For the first person you have three years of data, for the second person one year, and for the third person four years. Each year represents a level 2 "branch." Their values are as follows.

id	sex	educ	earnings	age	health
1	2	12	21000	32	1
			22000	33	1
			25000	34	2
2	1	16	43000	55	3
3	1	10	14000	23	3
			14500	24	2
			15200	25	4
			16000	26	5

The raw2aml control file should read as follows:

```
ASCII input file = filename;
level 1 var = sex educ;
level 2 var = earnings age health;
```

Your ASCII data should look as follows:

```
1 2 12 21000 32 1
1 2 12 22000 33 1
1 2 12 25000 34 2
2 1 16 43000 55 3
3 1 10 14000 23 3
3 1 10 14500 24 2
3 1 10 15200 25 4
3 1 10 16000 26 5
```

Note that there is a record for every level 2 branch. IDs and level 1 variables are repeated as many times as there are level 2 branches. The first three records have the same ID, so raw2aml groups them into one observation. In the process, it checks that the level 1 variables have the same values in each record and generates an error message if this is not the case. The output (`.dat`) data set removes the duplication and stores only one set of level 1 variables.

The above translates into the following rule.

ASCII data must contain one level 2 branch per record.

In following sections, the fuller meaning of this rule will become clear.

10.3. Three Levels, Single Data Structure

This section documents the order in which you need to write out data with three levels and a single data structure. You should read and understand the preceding sections first. We assume in this section that the data are "compressed," as opposed to "rectangular." Don't worry if you don't know the difference, as the compressed format is the most intuitive and commonly used format. Rectangular data are discussed starting on page 249.

The general order for data with three levels without distinct data structures is:

```
id nb
    x1
    x2
    x3
```

Before illustrating this data order, consider an alternative. For data with three levels, one could write an ASCII record for every level 3 branch, thereby repeating level 1 and level 2 variables as often as needed. Essentially, this would collapse level 3 onto level 2. Put differently, data with three conceptual levels would be written out with two technical levels. This is perfectly fine and often more convenient than writing out data with three technical levels. See Sections 3.4 and 10.5.

Raw2aml requires each ASCII record to correspond to a level 2 branch, and contain all lower level variables. Trouble is that some level 2 branches have more level 3 branches than others, i.e., that the data may be unbalanced. For example, if level 1 corresponds to a school, level 2 to a student, and level 3 to a grade level, then some students may attend more years at the school than others. You need to tell raw2aml how many level 3 branches (grades) to read in for every level 2 branch (student). This is done by creating a variable representing the number of level 3 branches. That variable, a raw2aml "control variable," is represented above as "`nb`". For data with three levels, it is a scalar variable; for data with more levels, `nb` is a vector, as explained in the next subsection.

Consider the example of a school progression model. We have data on schools (level 1), multiple students (level 2) in schools, and grade levels (level 3) for each student. The unit of observation in our (SAS, Stata, SPSS) data set is a student. Level 1 variables are `id` (school ID), `urban` (located in urban area?) and `private` (private school?); level 2 variables are students' `sex` and `byear` (year of birth) ; level 3 variables `grade1-grade4` (grade level) and `pass1-pass4` (did student pass this year?). Note that we list four grade level and pass variables. This is because the unit of observation in our data is the student (level 2 branch), and there are as many as four grade levels (level 3 branches) per student. However, not all students were in school all four years. Some dropped out or transferred, while others transferred in at an advanced level. We derive another level 2 variable, `numyears`, equal to the number of years that the student spent at this school. The raw2aml control file is:

```
ASCII input file = filename;
level 1 var = urban private;
level 2 var = sex age;
level 3 var = grade pass;
```

Note that there is only one `grade` and one `pass` variable, rather than (up to) four in your data set. Raw2aml and aML are aware of the nested level structure of the data, and need only one name for up to four variables. Suppose the first two schools in your data have the following values:

id	urban	private	sex	byear	numyears	grade				pass			
1	0	1	1	1980	4	9	10	11	12	1	1	1	1
			0	1979	3	9	10	11	.	1	1	0	.
			0	1981	3	10	11	12	.	1	1	1	.
2	1	0	1	1978	2	11	12	.	.	1	0	.	.
			1	1976	4	9	10	11	12	1	1	1	0
			1	1978	1	9	.	.	.	0	.	.	.
			0	1980	3	9	10	11	.	1	1	1	.

Note that variable `numyears` indicates how many years the student attended the school, and thus the number of relevant level 3 variables. It is the "`nb`" in the required variable order. The ASCII data should be as follows:

```
1 0 1 1 1980 4 9 1 10 1 11 1 12 1
1 0 1 0 1979 3 9 1 10 1 11 0
1 0 1 0 1981 3 10 1 11 1 12 1
2 1 0 1 1978 2 11 1 12 0
2 1 0 1 1976 4 9 1 10 1 11 1 12 0
2 1 0 1 1978 1 9 0
2 1 0 0 1980 3 9 1 10 1 11 1
```

Variables `id` and `numyears` are so-called control variables. The control variables are written out first, followed by all data variables. Do not specify control variables in the raw2aml control file! Raw2aml expects them to be in the data.

Note carefully that level 3 variables are in the following order: `grade1 pass1 grade2 pass2 grade3 pass3 grade4 pass4`, not `grade1-grade4 pass1-pass4`!

ASCII records that contain multiple sets of repeating variables, such as `numyears` sets of level 3 variables `grade` and `pass`, should first list all variables of the first branch, then all variables of the second branch, et cetera.

Also note that the ASCII records only contain the relevant level 3 variables, not the variables beyond `numyears` branches, i.e., not the irrelevant (missing) values. In SAS this is easily accomplished by looping over relevant branches:[29]

```
 1  data _null_;
 2     set yourdata;
 3     array grade(4)  grade1-grade4;  /* level 3 variable */
 4     array pass(4)   pass1-pass4;    /* level 3 variable */
 5     file 'yourname.raw';  /* ASCII data file */
 6     put id numyears;
 7     put urban private;    /* level 1 variables */
 8     put sex byear;        /* level 2 variables */
 9     do i=1 to numyears;
10        put grade(i) pass(i);  /* level 3 variables */
11     end;
```

Other data preparation packages, such as Stata, do not allow observation-specific loops. It may thus be very tedious to write out "compressed" ASCII data as above, with often fewer than four sets of level 3 variables. Instead, you may always write out four sets of level 3 variables; see the section on "rectangular" data on page 249.

ASCII data must contain one level 2 branch per record.

We repeat this rule, because its meaning may have become clearer now. Each ASCII record contains all information pertaining to a level 2 branch, including all lower level variables.

[29] The resulting ASCII data differ from those illustrated above in that each ASCII record wraps over multiple lines: one for `id` and `numyears`; one for `urban` and `private`; one for `sex` and `byear`; and `numyears` lines for `grade` and `pass`. As explained earlier, raw2aml reads as many lines as needed.

10.4. Four or More Levels, Single Data Structure

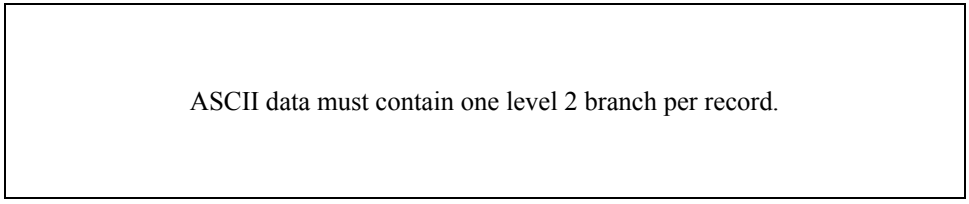
This section documents the order in which you need to write out data with four or more levels and a single data structure. You should read and understand the preceding sections first.

As stated before, it is rarely necessary to create data with four or more levels. It is typically more convenient to collapse levels such that the lowest (most disaggregated) level becomes level 2. Variables at all levels in between are then duplicated in the resulting data, but the resulting savings in programmer effort probably far outweigh additional computing costs. As a general rule, convert your data such that the unit of observation in your (SAS, Stata, SPSS) data set becomes level 2. See Sections 3.4 and 10.5.

There is no hard limit on the number of levels in aML. The general order for data with four or more levels without distinct data structures is:

```
id nb
    x1
    x2
    x3
    x4
   [x5]
   <et cetera>
```

As before, `id` and `nb` are control variables; all others are data variables. Variable id is a scalar. In data with three levels, nb is also a scalar (indicating the number of level 3 branches for each ASCII record). The situation becomes more complicated with four or more levels. In addition to telling raw2aml how many level 3 branches each level 2 branch (or ASCII record) contains, you also need to tell it, for every level 3 branch, how many level 4 branches there are. The "`nb`" above represents all such control variables jointly, and is thus a vector of non-negative integers. With five levels, you need to additionally specify the number of level 5 branches within each level 4 branch, et cetera. This information needs to be in the data, not the control file, because the data may be unbalanced, i.e., some branches contain more subbranches than others.



We repeat this rule, as it applies throughout. Consider again the schooling example, and suppose the data contain data on up to twelve tests (level 4) for every year in school (level 3) of every student (level 2). The variables reflect, say, `area` (subject area) and `score` (test score). There are up to four years in school for every student, so four sets of level 3 variables (`grade1`-

grade4 and pass1-pass4). Similarly, there are up to twelve tests per school year, so 4*12=48 sets of level 4 variables (area01-area48 and score01-score48). ASCII data must contain one level 2 branch per record, so all information pertaining to one student (level 2 branch) is written out in one sweep, i.e., all level 1, 2, 3, and 4 variables. (Yes, including the student's level 1 school variables, which is duplicative since they are written out on the records of all students of a school. It is a small price to pay for simplicity, particularly in writing your SAS/Stata/SPSS code.)

The raw2aml control file should contain:

```
ASCII input file = filename;
level 1 var = urban private;
level 2 var = sex byear;
level 3 var = grade pass;
level 4 var = area score;
```

As before, the unit of observation in your (SAS, Stata, SPSS) data is a student. Variable numyears indicates the number of years this student attended the school (0<=numyears<=4). For each year in school, the number of tests are denoted by ntest1 through ntest4, where 0<=ntest(i)<=12. In SAS, the data may be written out as follows:

```
 1  data _null_;
 2     set yourdata;
 3     array grade(4)    grade1-grade4;    /* level 3 variable */
 4     array pass(4)     pass1-pass4;      /* level 3 variable */
 5     array area(4,12)  area01-area48;    /* level 4 variable */
 6     array score(4,12) score01-score48;  /* level 4 variable */
 7     file 'yourname.raw';  /* ASCII data file */
 8     put id numyears;
 9     do i=1,numyears;
10        put ntest(i);   /* number of level 4 branches */
11     end;
12     put urban private;    /* level 1 variables */
13     put sex byear;        /* level 2 variables */
14     do i=1 to numyears;
15        put grade(i) pass(i);  /* level 3 variables */
16     end;
17     do i=1 to numyears;
18        do j=1 to ntest(i);
19           put area(i,j) score(i,j);  /* level 4 variables */
20        end;
21     end;
```

Note lines 8-11 in which the control variables (ID and the numbers of branches) are written out. Level 1 variables are written out on line 12, level 2 variables on line 13. Lines 14-16 show numyears sets of level 3 variables, and lines 17-21 write out numyears sets of ntest(i) sets of level 4 variables. The ID is only written out once, but recall that each SAS observation corresponds to a person, so that the same ID is written out multiple times for students of the same school. In other words, all data are written out at level 2.

Note the numbers of subbranches which appear after the ID. If there are three levels in your data, one integer is written out, namely the number of level 3 branches in the current (level 2) record. Let's denote the number of level 3 branches by `n3`. If there are four levels in your data, `1+n3` integers are written out: the first is the number of level 3 branches, and for every level 3 branch, an additional integer is written for the number of level 4 branches in the corresponding level 3 branch. Denote the number of level 4 branches in the `i`-th level 3 branch by $n4_i$. If there are five levels, $1+n3+\sum_{i=1}^{n3} n4_i$ integers are written: the number of level 3 branches, the numbers of level 4 branches in each level 3 branch, and the numbers of level 5 branches in each level 4 branch. Et cetera—raw2aml supports arbitrarily many levels.

ASCII data must contain one level 2 branch per record.

We repeat this rule again, because it is so central to your data organization. Make sure you fully understand its meaning.

10.5. An Common Alternative: Collapse All Levels to Level 2

Raw2aml requires that ASCII input data are written out one level 2 branch per record. This should pose no problem if the unit of observation in your (SAS, Stata, SPSS) data corresponds to a level 2 branch. In the examples above, we therefore assumed that the unit of observation in your data was a student, rather than a school, year in school, or test. What to do when the unit of observation in your data does not correspond to a conceptual level 2 branch?

One approach is to convert your data. In Stata, this may be done conveniently using the "`reshape`" command, and similar conversions are feasible in other data management packages. An attractive alternative is to leave the data as they are, and separate conceptual levels from technical levels. For example, suppose each observation in your school test data set corresponds to a test. Conceptually, the test is a level 4 branch. However, it is perfectly fine to act as-if the test is a level 2 branch. Technically, the test becomes a level 2 branch. The school remains at level 1, but all other conceptual levels collapse into technical level 2. The number of level 2 branches will be the same as all conceptual level 4 branches (tests per school) combined.

The raw2aml control file would be:

```
ASCII input file = filename;
level 1 var = urban private;
level 2 var = sex byear grade pass area score;
```

If the unit of observation in your data is a test, your data would contain fewer variables and more observations than if the unit of observation were a student. Instead of variables `grade1`-`grade4`, `pass1`-`pass4`, `area01`-`area48`, and `score01`-`score48`, your data would contain only four variables (`grade`, `pass`, `area`, and `score`). In SAS, you would simply write out ASCII data as follows.

```
1  data _null_;
2     set yourdata;
3     file 'yourname.raw';  /* ASCII data file */
4     put id urban private sex byear grade pass area score;
```

You may wonder how raw2aml and aML will know which variable is at what level. The answer is that they will not know this. The software will only be aware of the two levels that you specify. However, there will be no difference whatsoever in the aML control file. Models are specified using variable names only, without mention of level. There are two important exceptions; see below.

But then how will aML know which residuals' dependencies? For example, if your analysis of test scores includes school-specific, student-specific, year-specific, and test-specific residuals, how will aML know which tests belong to the same school, student, or year? The answer follows from aML's use of so-called draw variables (see Section 13.3.6). Residual draws are controlled by integer-valued draw variables, not by level structures. For example, the use of a student ID as

draw variable for the student-specific residual will tie all tests of a student together through their common student ID value.

There are two cases in which aML does care about the levels at which variables are stored.

1. In hazard models with time-varying covariates. Time marks and time-varying covariates must be one level below the outcome variables (censor and duration variables). For example, if the censor and duration variables are at level 2, time marks and time-varying covariates must be at level 3. If you were to collapse conceptual level 3 to technical level 2, you would be repeating the censor and duration variables as often as there are intervals (level 3 branches) in hazard spells (level 2 branches). aML would then think that the data contain many more outcomes than there really are.[30]

2. In continuous models with ARMA(*p*,*q*) or CAR(1) residuals. Both ARMA(*p*,*q*) and CAR(1) distributions are defined with time variables which specify how outcomes are spaced in time: "`timevar [(within level n)] = varname`" (Sections 13.2.7 and 13.2.8, respectively). The residuals are autocorrelated across outcomes within the specified level, and independent across branches of the specified level. It is thus important to maintain conceptual levels in the data.

The first exception draws attention to another case in which you should not collapse levels. Suppose you study grade promotion and test scores with data on schools (level 1), students (level 2), years in school (level 3), and tests (level 4). Grade promotions are outcomes at level 3; test scores at level 4. If you were to collapse all tests to level 2, all level 2 and level 3 variables are duplicated. In an analysis of grade promotion, aML would generate an equation for every outcome variable in the data, i.e., as many equations as there are tests in each year. These equations are not independent, and you would be led to believe that the model parameters are estimated with more precision than they should be. In short, duplicating explanatory variables is harmless, but don't collapse levels if that implies that you are duplicating outcome variables.

Collapsing data levels implies that variables above the lowest (most disaggregated) level are duplicated. For explanatory variables, this is harmless. However, don't collapse levels if there are outcome variables above the lowest level, as this would imply that more equations enter your model than there are independent outcomes.

[30] If you absolutely must collapse time-varying covariates to level 2, you can do so and create a hazard spell for every interval. Suppose there are five intervals in a non-censored hazard spell. You may break this up into five short hazard spells: four censored and one non-censored spell. This implies that you must replace the censor variable by one for all spells except the last, and replace all duration variables to reflect the short spells. Also, you need to add the cumulative duration in prior spells to duration spline origin variables, so that the origin keeps referring to the correct point in absolute time. Technically, there are no more time-varying variables in this set-up.

Apart from these cases, why bother with complicated level constructs? Well, there is indeed no compelling technical reason to do so. Keeping conceptual levels consistent with technical levels may help you stay organized and keep an overview of your data (especially useful when repeating the analysis a few years later). Another disadvantage of data with collapsed levels is that they take up more disk space. In the example, all level 2 and level 3 variables are duplicated in both the ASCII and aML (`.dat`) data files. However, the penalty in raw2aml and aML runtimes will typically be very small.

As a general rule, you may be best off assigning (technical) level 2 to whatever (conceptual) level your unit of observation corresponds. The time you will spend converting your data to preserve conceptual levels probably far exceeds the time lost due to longer runtimes. Whichever way you structure your data will make no difference for model specification and estimation in aML; all is defined and specified using variable names, and aML doesn't care at what level outcomes are repeated. Also see Section 3.4.

10.6. Multiple Levels, Multiple Data Structures

If your data are organized in data structures, you must indicate their numbers in the ASCII records. The data structure number is inserted immediately following the ID. It must be on the same line as the ID. (All other variables may be separated by line feed and/or carriage return characters.) All other variables are in the same order as in the single data structure case:

```
id struc [nb]
    x1
    x2
   [x3]
   [x4]
   <et cetera>
```

Data structure numbers must be strictly positive integers.

You may wonder why raw2aml requires that ASCII data must contain one level 2 branch per record. Why not one level 1 branch per record (i.e., write out an entire observation to a single record), or one level 3 branch per record? Recall that data structures apply to level 2 and lower only. There can only be one set of values for variables at the most aggregate (observation) level, but at level 2 and lower different variable lists may apply. It is only at level 2 and lower than data structures may be distinct. This underlies the requirement of one level 2 branch per record. It also tends to make the life of the programmer simpler. If you like, however, you may write out one ASCII record for each level 3 (or higher) branch; see the previous section.

10.7. Rectangular Data

The above discussion pertains to the so-called "compressed" data format. However, some data preparation packages do not support the creation of ASCII files as described above. In our sample data, there were up to four years in school per student and up to twelve tests per year. For many students, however, fewer than four years of data may be available, and some grade levels may have fewer than twelve tests. The remaining years and/or tests are irrelevant and will be missing (".") in the data. Raw2aml never accepts missing values in the form of dots ("."); see the next subsection. Irrelevant values should therefore preferably not be written out to the ASCII data. Our SAS sample code, above, looped over the actual number of years in school and the actual number of tests, and wrote out only relevant values. We call such data "compressed," because they exclude irrelevant values. Writing out compressed data is not always easy to do in some other data preparation packages, however. Also see Section 3.3.

Raw2aml offers the option of writing out some maximum number of branches at every level. This type of data are called "rectangular." You would write out relevant variables first, and pad the record with irrelevant variables such that the total number of branches equals the maximum number of relevant branches in the data. In the ASCII data file, the number of *relevant* branches for each record is written out immediately following the ID and the optional data structure number, just like in the examples above. The maximum number of level 3 and lower branches is specified in the control file. Raw2aml knows how many branches are relevant from the ASCII record, and knows how many branches to read in total from the raw2aml control file. It is a very good idea to assign a special number (preferably -99) to all irrelevant data values. Raw2aml reads the relevant data and issues a warning if any value is ever equal to the special number; it also reads the irrelevant data and aborts with an error message if any of the irrelevant values is ever not equal to the special number. The special number may be defined by

```
option irrelevance check = {yes | no | x};
```

By default (`yes`), the special number is `-99`; another number may be selected by specifying it on the right-hand-side; specifying "`no`" turns off the relevance/irrelevance checks. Also see page 224. Consider again the schooling example with up to four years in school (level 3 branches) per student and up to twelve tests (level 4 branches) per year in school. (The maximum number of students per school does not matter, because each ASCII record corresponds to one student; the distinction between compressed and rectangular data matters only at level 3 and lower.) Suppose the unit of observation in your (Stata) data set is a person. Ignore for now everything at level 4; we first illustrate how to create a rectangular data file with three levels. The raw2aml control file contains the following variable specification:

```
ASCII input file = filename;
level 1 var = urban private;
level 2 var = sex byear;
level 3 var (nb=4) = grade pass;
```

Note the "nb=4" which specifies that the number of level 3 branches under each level 2 branch (i.e., on every ASCII record) is always four. The data may be written out as follows.[31]

```
/* All irrelevant values are missing; set them to -99 */
mvencode _all, mv(-99)
#delimit ;
outfile id numyears urban private sex byear
   grade1 pass1
   grade2 pass2
   grade3 pass3
   grade4 pass4
   using yourname.raw, comma wide;
```

Note that the second variable, numyears, tells raw2aml how many relevant years in school (level 3 branches) to read; from the control file it knows that the total number to read is always four. Based on the data shown on page 240, the ASCII file will contain the following:

```
1,0,1,1,1980,4,9,1,10,1,11,1,12,1
1,0,1,0,1979,3,9,1,10,1,11,0,-99,-99
1,0,1,0,1981,3,10,1,11,1,12,1,-99,-99
2,1,0,1,1978,2,11,1,12,0,-99,-99,-99,-99
2,1,0,1,1976,4,9,1,10,1,11,1,12,0
2,1,0,1,1978,1,9,0,-99,-99,-99,-99,-99,-99
2,1,0,0,1980,3,9,1,10,1,11,1,-99,-99
```

Now introduce the fourth level with tests. The raw2aml control file contains the following:

```
ASCII input file = filename;
level 1 var = urban private;
level 2 var = sex byear;
level 3 var (nb=4) = grade pass;
level 4 var (nb=12) = area score;
```

Note again the "nb=12" on the last line: there are always 12 sets of level 4 variables in the ASCII data, even though fewer may be relevant. The data may be written as follows:

```
/* First set all missing values to -99 */
mvencode _all, mv(-99)
#delimit ;
outfile id numyears urban private sex byear
   grade1 pass1
```

[31] Stata's "#delimit ;" command sets the command delimiter to a semicolon, so that commands may wrap over multiple lines. The "comma" option of the outfile statement causes the ASCII data to be comma-delimited, resulting in far smaller file sizes than the default of multiple spaces. The "wide" option prevents Stata from wrapping the lines; it sometimes helps when looking at ASCII data to find the cause of an error. Neither the "comma" nor the "wide" option is required.

```
grade2 pass2
grade3 pass3
grade4 pass4
area01 score01
area02 score02
      :
area48 score48
using yourname.raw, comma wide;
```

Note that we write out 4*12=48 sets of level 4 variables. Each of these sets may have both relevant and irrelevant (-99) variables. Consider a student who attended the school for three years and took 10, 12, and 8 tests. His area01-area10 are relevant and contain actual values; area11 and area12 are equal to –99; area13-area24 are all relevant; area25-area32 are also relevant; area33-area36 are equal to –99; and area37-area48 are all equal to –99. Similar for score01-score48.

In summary, the table below defines the exact ASCII variable sequence for compressed data.

Compressed Sequence of Variables in an ASCII Record

description	data type	Definition
id	1 integer	Identification number for the record
struc	1 integer	Structure number (optional)
n3	1 integer	Number of level 3 branches (only for level 3 and lower data structures).
$(\text{n4}(i),$ $i = 1,\ldots,\text{n3})$	n3 integers	Numbers of level 4 branches in each of the level 3 branches (only for level 4 and lower data structures).
$((\text{n5}(i,j),$ $j = 1,\ldots,\text{n4}(i)),$ $i = 1,\ldots,\text{n3})$	$\sum_{i=1}^{\text{n3}} \text{n4}(i)$ integers	Number of level 5 branches in each of the level 4 branches for each level 3 branch (only for level 5 and lower data structures).
Etc, for level 6 and lower		
$(\text{x1}(i),$ $i = 1,\ldots,\text{nx1})$	nx1 reals	All (nx1) level 1 variables.
$(\text{x2}(i),$ $i = 1,\ldots,\text{nx2})$	nx2 reals	All (nx2) level 2 variables
$((\text{x3}(i,j),$ $j = 1,\ldots,\text{nx3}),$ $i = 1,\ldots,\text{n3})$	nx3 × n3 reals	All (nx3) level 3 variables in each of the n3 level 3 branches, i.e., n3 series of nx3 variables. First record all nx3 variables of the first level 3 branch, then all nx3 variables of the second level 3 branch, etc. (Only for level 3 and lower data structures.).

description	data type	Definition
$(((\text{x4}(i,j,k),$ $k=1,...,\text{nx4}),$ $j=1,...,\text{n4}(i)),$ $i=1,...,\text{n3})$	$\text{nx4}\times\sum_{i=1}^{n3}\text{n4}(i)$ reals	All (nx4) level 4 variables in each of the level 4 branches for each level 3 branch, i.e., $\sum_{i=1}^{\text{n3}}\text{n4}(i)$ series of nx4 variables. First record all nx4 variables for the first level 4 branch of the first level 3 branch, then all nx4 variables for the second level 4 branch of the first level 3 branch, etc. (Only for level 4 and lower data structures.)
$((((\text{x5}(i,j,k,l),$ $l=1,...,\text{nx5}),$ $k=1,...,\text{n5}(i,j)),$ $j=1,...,\text{n4}(i)),$ $i=1,...,\text{n3})$	$\text{nx5}\times$ $\sum_{i=1}^{\text{n3}}\sum_{j=1}^{\text{n4}(i)}\text{n5}(i,j)$ reals	All (nx5) level 5 variables in each of the level 5 branches for each level 4-branch and each level 3 branch, i.e., $\sum_{i=1}^{\text{n3}}\sum_{j=1}^{\text{n4}(i)}\text{n5}(i,j)$ series of nx5 variables. The loop over index l is executed most; the loop over i only once. (Only for level 5 and lower data structures.)
Etc, for level 6 and lower		

In the rectangular form, all records must have the same number of variables. The structure and definitions are the same as for the compressed form, above. The only difference is that records are padded with irrelevant values that are set to a special value, preferably -99. The table below defines the structure of each ASCII record in rectangular form.

Rectangular Sequence of Variables in an ASCII Record

description	data type	Definition
id	1 integer	Identification number for the record
struc	1 integer	Structure number (optional)
n3	1 integer	Number of level 3 branches (only for level 3 and lower data structures).
$(\text{n4}(i),$ $i=1,...,\text{maxn3})$	maxn3 integers	Numbers of level 4 branches in all level 3 branches, whether relevant or irrelevant (only for level 4 and lower data structures). n3 relevant values; maxn3−n3 irrelevant values, set to -99.
$((\text{n5}(i,j),$ $j=1,...,\text{maxn4}),$ $i=1,...,\text{maxn3})$	maxn3 × maxn4 integers	Number of level 5 branches in all level 4 branches in all level 3 branches, whether relevant or irrelevant. (only for level 5 and lower data structures).
Etc, for level 6 and lower		
$(\text{x1}(i),$ $i=1,...,\text{nx1})$	nx1 reals	All (nx1) level 1 variables.
$(\text{x2}(i),$ $i=1,...,\text{nx2})$	nx2 reals	All (nx2) level 2 variables

description	data type	Definition
$((\text{x3}(i,j),$ $j=1,...,\text{nx3}),$ $i=1,...,\text{maxn3})$	nx3 × maxn3 reals	All (nx3) level 3 variables in each of the maxn3 level 3 branches, i.e., maxn3 series of nx3 variables. First record all nx3 variables of the first level 3-branch, then all nx3 variables of the second level 3-branch, etc. (only for level 3 and lower data structures). n3 sets of relevant variables and maxn3−n3 sets of irrelevant values, set to -99.
$(((\text{x4}(i,j,k),$ $k=1,...,\text{nx4}),$ $j=1,...,\text{maxn4}),$ $i=1,...,\text{max3})$	nx4 × maxn4 × maxn3 reals	All (nx4) level 4 variables in each of the level 4 branches for each level 3 branch, i.e., maxn4 × maxn3 series of nx4 variables. First record all nx4 variables for the first level 4 branch of the first level 3 branch, then all nx4 variables for the second level 4 branch of the first level 3 branch, etc. Pad with -99 for irrelevant level 4 branches of the first level 3 branch. The all nx4 variables of the first level 4 branch of the second level 3 branch, etc. (only for level 4 and lower data structures).
$((((\text{x5}(i,j,k,l),$ $l=1,...,\text{nx5}),$ $k=1,...,\text{maxn5}),$ $j=1,...,\text{maxn4}),$ $i=1,...,\text{maxn3})$	nx5 × maxn5 × maxn4 × maxn3 reals	All (nx5) level 5 variables in each of the level 5-branches for each level 4-branch and each level 3-branch, i.e., maxn5 × maxn4 × maxn3 sets of nx5 variables. The loop over index l is executed most; the loop over i only once. (only for level 5 and lower data structures).
Etc, for level 6 and lower		

Numeric values in the ASCII files must be separated by blanks, commas, or tab characters. The id must always be the first number on a line and the data structure number must appear on that same line. All other numbers may span multiple lines. In other words, all variables may also be separated by line feeds and/or carriage returns, except for the ID and data structure number, which must appear on the same line.

10.8. Missing Values and Character Variables

Most third-party statistical packages internally reserve one or more special numerical values to represent the "missing value." Upon input and output, such missing values are typically represented by a period (".."). Raw2aml and aML do *not* support such missing values. All numerical values must be legitimate numbers on the real line.

This does not imply that you may not use aML if your data contain missing values. It only implies that you need to resolve missing values before transforming your data into aML-format using raw2aml. One method for resolving missing values is to impute them. Another common method is to generate a separate indicator variable which flags whether the original variable is missing. If the original variable is missing, the indicator variable is one; else, it is zero. Now that there is a flag for missing values, missing values of the original variable may be set to some legitimate numerical value, such as zero or the mean over nonmissing values. (The latter provides a test for whether the variable is missing randomly.)

Before writing out ASCII data, make sure that all missing values are resolved. If periods (".") inadvertently enter ASCII data, raw2aml will abort with an error message. It will attempt to diagnose the problem and communicate its findings.

In rectangular data, it is important to distinguish between missing and irrelevant values. Irrelevant values occur when a level 2 or lower branch has fewer subbranches than the maximum number in the data. Irrelevant should be set to -99 before creating ASCII data files. Raw2aml will recognize that they are irrelevant and discard them. The aML-formatted data will only contain relevant values. By contrast, missing values occur, for example, when a respondent is not able or willing to respond to a question. These values must be resolved before creating ASCII data files.

Finally, aML does not support character string variables with values such as "`abc`" or "`yourname`". Only numerical variables are supported. If a character string inadvertently enters ASCII data, raw2aml will abort with an error message. Note that values like "`1.23E+02`" (without the double quotes) are interpreted correctly as numerical, not string values.

11. Raw2aml Data Documentation File

Raw2aml produces two output files: a data file (with extension `.dat`) and a data documentation file (with extension `.sum`). The data file is in binary format and should only be read by aML. The data documentation file is a text file and you may read it using any text editor. This chapter interprets the raw2aml output file with summary statistics of the data set. Raw2aml wrote its contents also to standard output, i.e., to the window from which raw2aml was run.

11.1. General Output

The data documentation file, also known as summary file, contains the following information.

- Name of data file to which the documentation pertains, date of creation, and names of ASCII input files.
- Number of observations, maximum numbers of branches per observation, maximum numbers of level *n* branches per level *m* branch, where $m<n$.
- A frequency table of data structures.
- Variable names, means, standard deviations, minima, and maxima, for all variables and for all data structures and levels. In addition, if one or more variable names appear in multiple data structures, raw2aml reports summary statistics as computed across all data structures in which the variable(s) appear.
- Notes and warnings, if any.

Consider the following summary file:

```
 1  Documentation for 'path\filename.dat'
 2  Created on Sun Jul 25 21:43:04 1999 with raw2aml version n.
 3  ID file: 'path\id.raw'
 4  Ascii data sets: 'path\file1.raw'
 5                   'path\file2.raw'
 6                   'path\file3.raw'
 7                   'path\file4.raw'
 8
 9  Number of observations:    5825
10  Maximum number of level 2 branches in any observation:     25
11  Maximum number of level 3 branches in any observation:     96
12  Maximum number of level 3 branches in any level 2 branch:  42
13
14       STRUCTURE |      Freq.     Percent
15    -------------+------------------------
16             100 |       7202       11.96
17             200 |       4860        8.07
18             300 |      10243       17.02
19             400 |       9426       15.66
```

```
 20             510 |      21326       35.43
 21             520 |       7141       11.86
 22    -------------+-----------------------
 23           Total |      60198      100.00
 24
 25  ------------------------------------------------------------
 26
 27  LEVEL 1 VARIABLES:
 28  Variable      N       Mean    Std Dev        Min        Max
 29  _id        5825   5798.123   3423.358        1.0    12558.0
 30  var1       5825   .9781321   .7240252     0.1121     3.6698
 31  var2       5825   .1460944   .4421697        0.0        7.0 (dev=mean)
     et cetera
 43
 44  ------------------ DATA STRUCTURE 100 ----------------------
 45
 46  LEVEL 2 VARIABLES:
 47  Variable      N       Mean    Std Dev        Min        Max
 48  var20      7202   0.324632   .4682697        0.0        1.0 (dev=.5388737)
 49  var21      7202   2.869869   6.121001        0.0       24.4 (dev=12)
     et cetera
 74  var36      7202        0.0        0.0        0.0        0.0
 75  var37      7202    99999.0        0.0    99999.0    99999.0
 76
 77  LEVEL 3 VARIABLES:
 78  Variable      N       Mean    Std Dev        Min        Max
 79  var40     88235   10.08367   4.763343      0.003     24.791
 80  var41     88235   .0730662   .2602467        0.0        1.0
     et cetera
 88
 89  ------------------ DATA STRUCTURE 200 ----------------------
 90
     et cetera
324
325  ------------------------------------------------------------
326
327  NOTE: there are 11 data variables without any variation.
328
329  NOTE: the following 64 variables appear in multiple data structures.
330  Summary statistics across all data structures in which they appear:
331
332  Variable      N       Mean    Std Dev        Min        Max
333  var20     31731   .5388737   .4984944        0.0        1.0
334  var21     31731   8.327178   7.113525        0.0       24.4
     et cetera
397
398  NOTE: there is variation in all data variables.
399
400  WARNING: there is at least one variable whose name appears at varying
401  levels in multiple data structures.  One such variable is 'var45'.
```

The first section (lines 1-7) lists the name of the data file to which the summary file pertains, the date and time at which it was created, the ID file, and the ASCII input file(s). To be precise: the date and time indicate the moment raw2aml started execution, not when it terminated.

```
Number of observations:    5825
Maximum number of level 2 branches in any observation:     25
Maximum number of level 3 branches in any observation:     96
Maximum number of level 3 branches in any level 2 branch:  42
```

The second section (lines 9-12) reports the number of observations, the numbers of branches within any one observation, and the number of branches within any one other branch. In the example, there were 5,825 observations, i.e., 5,825 distinct IDs. There are as many as 25 level 2 branches per observation, as many as 96 level 3 branches per observation, and as many as 42 level 3 branches per level 2 branch. This enables you to check the integrity of the data. If observations correspond to schools, level 2 to students, and level 3 to grade levels, would these figures make sense? Are there indeed 5,825 schools in your sample? Does one or more schools contribute 25 students to the data? Are there as many as 96 years enrolled per school? Can it be that one or more student was enrolled for 42 years? Probably not, and the information provided here should thus prompt you to revisit the way in which data are written out, and/or the raw2aml control file specification.

If the data contain more than three levels, this section will report additional maximum numbers of branches per observation and per higher-level branch. Note the symmetry of the information that is reported. By definition, an observation is a level 1 branch, so raw2aml reports the maximum number of level *n* branches per level *m* branch, for all *n>m*.

The third section (lines 14-23) tabulates the number of data structures in your data. Recall that data structures are defined or distinguished at level 2, so the table also tells you the total number of level 2 branches in the data (here, 60,198). Does this correspond to the number you expect?

The fourth section (lines 44-325) reports summary statistics of all data variables, by data structure and level. All data variables are summarized. In addition, line 29 summarizes the ID variable. You did not specify this variable in the raw2aml control file, but raw2aml makes it available in the data as-if it were a data variable. It names the variable `_id`. If you like, you may use it in aML model specifications.

The summary statistics include the number of times the variable appears in the data (within data structure), its mean, standard deviation, minimum, and maximum value. Note that the numbers of times level 2 variables appear correspond to the data structure frequency on lines 14-23. Level 3 variables appear more often, as there may be multiple level 3 branches for each level 2 branch.

The data documentation file indicates on the far right which, if any, variables are in deviations from their mean or from some other value. (This information is not provided among the summary statistics as computed across all data structures combined; see below.) Note line 31, which states "`(dev=mean)`". This indicates that the (`.dat`) data file contains variable `var2` in deviations from its mean, i.e., in deviations from 0.324632. This was achieved by specifying "`var2(d)`" in the raw2aml control file (page 225). Line 48 states "`(dev=.5388737)`". Variable `var20` was apparently transformed into deviations from 0.5388737. This was achieved by specifying

"`var20(d)`" in the raw2aml control file. However, `var20`'s mean as reported on line 48 is 0.324632, not 0.5388737; by default, raw2aml deducts the cross-data structure mean. As explained below, line 333 indicates that that mean is indeed 0.5388737. Line 49 states "`(dev=12)`" indicating that the `.dat` file contains `var21-12`, rather than just `var21`. This was achieved by specifying "`var21(d=12)`" in the raw2aml control file.

11.2. Notes, Warnings, and Error Messages

Raw2aml issues notes to help the user determine whether the data have been converted correctly, and warnings when it finds patterns in the data that may be indicative of a user error. In addition, it issues an error message when something is definitely wrong. Notes and warnings do not affect execution of the program; error messages always abort execution. This section documents notes and warnings; the very many error messages that are built into raw2aml are designed to be fully informative and self-evident.

NOTE: there are *n* data variables without any variation

Raw2aml computes summary statistics of all data variables. If there are multiple data structures, these statistics are at first computed within data structure. In other words, even if a variable appears in multiple data structures, summary statistics are at first computed for each data structure without regard to other data structures. (Statistics pooling data structures are also computed and reported; see below.) After raw2aml writes out summary statistics (such as on lines 44-325 in the sample documentation file), it checks whether all variables have variation. If there are variables without variation (i.e., variables which always have the same value within a data structure), raw2aml writes out a note.

Line 327 in the sample file draws attention to the fact that there are 11 variables without any variation. Two of those variables appear on lines 74 and 75. Note that their standard deviation is zero. Data variables without variation do not have much informational content. If there were only one data structure, the above note would thus be replaced by a warning. Similarly, if there is no variation across all data structures combined, a warning is written out; see below.

NOTE: there is variation in all data variables

Complementary to the preceding note, this message confirms that there is variation in all data variables. In the sample documentation file, this statement appears on line 398, i.e., when all data structures are considered jointly. The 11 variables that had been identified without any variation (line 327) apparently did take on multiple values across data structures.

NOTE: the following *n* variables appear in multiple data structures

Line 329 draws attention to the fact that 64 variables appear in multiple data structures, i.e., that the user assigned the same name at least twice. It is not allowed to give the same name to two or more variables in the same data structure, but it is perfectly fine to use names multiple times across data structures. Lines 44-325 reported summary statistics within data structures; lines 332-396 report summary statistics as computed

across all data structures in which a variable appears. The latter section does not indicate whether any variables are in deviations from their mean or other value.

NOTE: all variable names are unique

The names of variables in any one data structure must be unique, but it is perfectly fine to assign the same name to variables in different data structures. This note indicates that all names are unique, even across data structures. It is the complement of the preceding note.

NOTE: all irrelevant level 3+ variables are equal to -99

This message applies to conversions of rectangular data only. As discussed on pages 224 and 249, it is a very good idea to set irrelevant variables equal to some special number, such as -99. Raw2aml checks that all irrelevant values are indeed equal to -99 (or whatever special number you chose in "`option irrelevance check`"). This note indicates that all appears to be OK. It supports that you correctly lined up your relevant and irrelevant values.

NOTE: relevant level 3+ variables are never equal to -99

This message applies to conversions of rectangular data only. Complementary to the preceding note, it states that relevant data variables are never equal to -99 (or whatever special number you chose in "`option irrelevance check`"). It supports that you correctly lined up your relevant and irrelevant values.

Relevant variables may take on any value, including -99, but raw2aml's check on what values are relevant and irrelevant is more powerful if the special value is never a legitimate value. If relevant variables are ever equal to the special value, raw2aml issues a warning; see below.

WARNING: there are *n* data variables without any variation

This message draws attention to the fact that there are variables in your data without any variation, i.e., variables that always take on the same value. Their standard deviation is zero. Such variables have little or no informational content. If they are used in regressions, they will be collinear with the intercept term, and their coefficient cannot be estimated.

The warning should prompt you to check whether you wrote out your ASCII data correctly. An alternative explanation is that the variables really do not have any variation, in which case you may want to check your (SAS, Stata, SPSS) data preparation logic.

WARNING: after computing summary statistics across all data structures in which each variable appears, there are still *n* data variables without any variation

This message is similar to the preceding one, but is issued if there are multiple data structures, and some variables lack variation even across data structures.

WARNING: there is at least one variable whose name appears at unequal levels in multiple data structures. One such variable is '*varname*'.

The names of variables in any one data structure must be unique, but it is perfectly fine to assign the same name to variables in different data structures. It is also perfectly fine to use the same name for variables that are at, say, level 2 in one data structure and level 3 in another. However, if data levels in raw2aml correspond to conceptual data levels, it does not often make sense to use the same name for variables at different levels. This message draws attention to duplicate names at unequal levels.

WARNING: your very first variable is 'varname'. Note that the observation ID is assumed to be on every record in the first position; it should not be listed among the data variables, unless it really appears twice in the data.

This warning hopes to prevent a very common error. The very first variable on ASCII records is always the observation's ID variable. This is a control variable, not a data variable with substantive meaning. Since it is a control variable, it should not be listed among (level 1) variables in your raw2aml control file. However, many users explicitly list the ID variable. Raw2aml will think that the number of level 1 variables is one more than it really is, and attempt to read in an extra variable for each record. The result is that all variables are offset, i.e., different in your (SAS, Stata, SPP) data set and the `.dat` file. In addition, raw2aml counts too few observations, because it reads one line too many for each record.

To help prevent this error, raw2aml checks whether the first variable in your list of level 1 variables is '`id`' or '`ID`'. If that is the case, it ventures the above warning. If you listed the ID variable under a different name, raw2aml may be unable to catch the error.

WARNING: the IDs are not sorted in increasing order. There is no reason why they should be, but it may be indicative of a failure to properly match various records into one observation. Please check that the number of observations (*n*) is correct.

This message is self-explanatory. A very common cause of this message is a misspecification of the number of variables in the raw2aml control file. If you list more variables than there really are in the data, raw2aml will read additional lines in an attempt to read the additional variables. It will then interpret the first value on the next line as an

ID, whereas that variable may in fact be a data variable. Similarly, if you list too few variables, raw2aml may not read enough lines per ASCII record. The resulting number of observations will then be too low or too high, hence the last sentence of the warning.

It is a good idea to always sort your IDs (in SAS, Stata, SPSS) in increasing order. If this warning is issued, check that the number of observations corresponds to your expectation, and that the number of variables in your (SAS, Stata, SPSS) output statement corresponds to the number listed in the raw2aml control file.

WARNING: *n* level 3+ variables are relevant but equal to -99.

This message applies to conversions of rectangular data only. As discussed on pages 224 and 249, it is a very good idea to set irrelevant variables equal to some special number, such as -99. In addition to checking that all irrelevant values are indeed equal to -99 (or whatever special number you chose in "`option irrelevance check`"), raw2aml checks whether any relevant value is ever equal to the special number. While there is no formal objection against this, it weakens the data integrity test that raw2aml carries out. If your data are rectangular, and some relevant values are truly equal to -99, we suggest that you set irrelevant values to some other special number (say, -99999) to which relevant variables are never equal. Tell raw2aml to check that irrelevant values are always equal to that special number, and that relevant values are never equal to that special number, by specifying "`option irrelevance check = -99999`". Also see page 224.

12. aML Command Line Options

aML features many options. Some may be specified on the command line, i.e., in the DOS or UNIX window, when you invoke the aML executable program; some may be specified in the aML control file, i.e., the `.aml` file; and some may be specified either way. This chapter documents command line options; Chapter 13 deals with control file options.

aML may be invoked with the following command line options:

`aml [-h] [-l] [-c] [-r] [-o file.dat] [-m macrofile] file.aml`

Command line options may appear in any order. The remainder of this chapter explains how each option alters aML's default behavior.

Option -h

Option `-h` ("help") provides limited on-line help. It lists all supported command line options and concisely explains their use.

Option -l

Option `-l` (letter "l", as in "license," not number "1" as in "one") is used for displaying and updating license information. The very first time you run aML, you should specify this option. aML will ask for your serial number and authorization code and create a license file. Each subsequent time that you run aML, it will check that you have a valid license file. For more information see page 5.

Option -c

Option `-c` ("course") makes aML behave as-if you have a course license, regardless of your actual license. A course license is intended for use by students who take a statistics course which requires hands-on experience with aML. It powers down aML's capabilities. Specifically, it limits the number of data observations, the number of model parameters, and the number of covariates. The `-c` option is intended for course instructors who themselves have an unrestricted aML license. The option enables the course instructor to verify that students will be able to carry out the assignments by temporarily powering down his or her own version of aML.

Option -r

By default, aML checks whether the specified output file with estimation results (`.out` file) already exists on disk. If it indeed already exists, it asks the user for

permission to overwrite. With option `-r` ("replace"), aML will overwrite the output data file without confirming that this is OK.

Option -o *file.out*

Option `-o` ("output") specifies the output file with estimation results. By default, output is written to a disk file with the same name as the control file, but with extension "`.out`" instead of "`.aml`". For example, if the control file is "`abcdef.aml`", the output file will be "`abcdef.out`". (If the control file does not have extension "`.aml`", the output file name is derived by appending "`.out`" to its name. For example, control file "`abcdef.ctl`" results in "`abcdef.ctl.out`"; control file "`abcdef`" results in "`abcdef.out`".) This default behavior may be changed by using the "`-o`" option and explicitly specifying the name of the output file.

We strongly recommend that you follow our convention of naming output data files with extension `.out`. It greatly helps in keeping your files organized.

Option -m *macrofile*

aML control files may contain user comments which aML should ignore (Section 16.1). This is implemented through a control file pre-processor which strips out all user comments. By default, the stripped-down control file is saved only temporarily to disk, and is removed as soon as aML has finished parsed the control file statements. Option `-m` causes aML to save the stripped-down control file permanently to disk, so that you may look at its contents. Its name is the user-specified *`macrofile`*.

There is no reason why you would ever want to look at a version of your control file without comments. A more useful purpose of the `-m` option is to debug macros that you may embed in aML control files. The raw2aml control file pre-processor namely supports a simple macro language (Section 16.2). The stripped-down control file has all macros resolved; it is what aML actually parses. Should aML protest at your use of macros, then this will provide a way to determine how your macros were resolved. It is a particularly useful option when using nested macros, which may get tedious.

Argument *file.aml*

The last command line argument, *`file.aml`*, is not optional. It specifies the name of the aML control file. We wrote *`file.aml`*, with extension `.aml`, to remind you of the convention to name control files with a `.aml` extension. However, you may specify any name.

If you do not specify an extension, aML will first look for a control file with extension `.aml`. If such a file does not exist on disk, it will look for the file as you specified it. For example, if you specify argument "`abcdef`", aML will first attempt to open "`abcdef.aml`"; if such a file does

not exist, it will look for "`abcdef`". If you specified "`abcdef.ctl`", aML will first look for "`abcdef.ctl.r2a`", then for "`abcdef.ctl`".

13. aML Control File Statements

aML commands are run in batch mode, i.e., you issue a complete series of commands at once. Those commands specify the input data set, specify models, initiate parameter values, et cetera. aML reads your commands from a control file and writes out the estimation results to both standard output (the computer window) and an output file. This chapter documents the syntax of all control file statements.

An aML control file consists of four parts: global control and options, model element ("building block") definitions, model specifications, and starting values. Each is documented in a subsection below. In addition, two subsections document indirect referencing and expressions.

Options, definitions, model specifications, and starting values are all given in statements. These statements may wrap over multiple lines. The end of a statement is delineated by a semicolon (";"). Comments may be inserted freely; they must be opened by a `/*` forward slash-asterisk combination and closed by an asterisk-slash `*/`; see Section 16.1. Repetitive statements or other repetitive blocks of text may be replaced by user-defined macros; see Section 16.2. The maximum line width in the control file is 80 characters.

13.1. Global Control and Options

The first section of the aML control file specifies the name of the input data set, the desired detail of reported estimation results, settings of the maximum likelihood search algorithm, dimensions of scratch arrays, and several miscellaneous settings. The name of the input data set must be specified in every aML control file; all other settings are optional. The table below lists the available options, their default values, and the page on which they are described.

Description	Default Value	Page
Input Data Set Control		
dsn = *filename*;		269
Output Control		
option title = '*string*';		269
option screen info level = *n*;	3	270
option file info level = *n*;	5	270
option numerical search;		270
option numerical standard errors;		270
option huber;		272
option variance-covariance matrix;		272
option correlation matrix;		273
option Hessian matrix;		273
option table format;		273
option starting value format;		273
Search/Optimization Control		
option observations = *n*;	0	274
option weight = *varname*;	none	274
option normalized weight = *varname*;	none	274
option iterations = *n*;	40	274
option step range = *n* [to *n*];	-10 to 1	275
option save step;		276
option converge = {wgn|rfi|gn|rpc}<*x* [or ...];	wgn<0.1	276
Scratch Array Control		
option maximum specification space = *n*;	1000	278
option maximum model space = *n*;	5000	278
option maximum hazard baseline space = *n*;	4000	278
option maximum scratch data space = *n*;	≥5000	278
option maximum number of frequency categories = *n*;	20	279
option maximum number of residual draws = *n*;	100	279
option maximum number of hazard baseline nodes = *n*;	100	279
option maximum number of reference numbers = *n*;	300	279
option maximum number of user-defined constants = *n*;	500	280

Description	Default Value	Page
option maximum number of hazard model splines = *n*;	20	280
Miscellaneous		
option scratchdsn = *filename*;	none	280
option gridfile = *filename[.dct]*;	none	280
option ensure positive definite = {yes|no};	yes	280
option check99999 = {yes|no};	yes	280
option version = *n*;	2	281

13.1.1. dsn = filename;

Each run must include the name of the data file to be used in estimation. For example:

```
dsn = mydata;
dsn = mydata.dat;
dsn = ..\Data\mydata;
dsn = "C:\My Documents\Project\mydata";
dsn = /max/a/Data/aml/mydata.dat;
```

By default, raw2aml creates data files with extension ".dat". Similarly, aML expects to read data from ".dat" files. It first tries to open the filename that you specified, with ".dat" appended. If this fails, it drops the ".dat" extension and tries again.

You may specify a relative or absolute path, as shown in the third, fourth, and fifth examples. If no pathname is specified, as in the first and second examples, aML assumes that the file is located in the current working directory, i.e., the directory from which the program is run. PC-based systems tend to use backslashes ("\") in paths, whereas UNIX systems use forward slashes ("/"). aML accepts either convention on both platforms.

Enclosing the filename in quotes is optional. Both single and double quotes are accepted, as long as you use two single or two double quotes. If the filename includes a space (as in the fourth example), you must use single or double quotes.

13.1.2. option title = "string";

A title for the control file may be specified and will be printed at the top of the first page of the output file. The title must fit on one line and may thus be no more than 80 characters in length. It must be delimited by either single or double quotes. (Single quotes is suggested; use double quotes if the title itself includes single quotes.) By default, there is no title. Examples include:

```
option title = 'Title here.';
option title = "Model for 'low' and 'high' earners";
```

13.1.3. option screen info level = n; option file info level = n;

These options control the amount and detail of information that aML writes to standard output and output file, respectively. There are five levels of output detail:

Level	Description
0	No output
1	Only estimation results including convergence criteria, parameter estimates, standard errors and asymptotic t-statistics.
2	Adds for each iteration: log-likelihood at the end of each iteration.
3	Adds for each iteration: beginning parameter value, gradient, search direction at stepsize 1, smallest 5 eigenvalues of the information matrix, current numerical values of all criteria used for convergence.
4	Adds information about the input and output files, input data, date and time, convergence criteria, defined building blocks, and model statements.
5	Adds summary statistics of outcomes, covariates, and other variables such as reference variables and spline origins.

By default, aML writes the highest level of detail (5) to the output file, and an intermediate level (3) to the screen. We recommend that you maintain all detail in the output file, and study the presented information carefully.

13.1.4. option numerical search;

This option instructs aML to compute the matrix of second derivatives (Hessian matrix) numerically, i.e., as the numerical derivative of analytically computed first derivatives. This numerical Hessian is used in determining the search direction. It also forms the basis for calculating standard errors of parameter estimates, so that "`option numerical search`" implies "`option numerical standard errors`" (see below).

aML's search algorithm is Gauss-Newton (Judge et al., 1985), so that the search direction d is minus the product of the inverse Hessian matrix and the gradient vector:

$$d = -\left(\frac{\partial^2 \ln L}{\partial\theta\partial\theta'}\right)^{-1}\left(\frac{\partial \ln L}{\partial\theta}\right).$$

By default, the Hessian matrix is computed using the BHHH approximation (Berndt, Hall, Hall, and Hausman, 1974):

$$\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \approx -\sum_{i=1}^{N}\left(\frac{\partial \ln L_i}{\partial\theta}\right)\left(\frac{\partial \ln L_i}{\partial\theta'}\right),$$

where $\ln L_i$ is the log-likelihood for the *i*-th observation and *N* is the number of observations. This approximation tends to be good for large sample sizes, but can be poor for smaller samples. For the purpose of searching, this is typically harmless. The search direction may not be optimal, but the maximum likelihood is typically reached eventually.

If the program has trouble converging, consider searching on the basis of a numerically computed Hessian matrix that is not subject to the BHHH approximation. This option can be very time-consuming: with *k* free parameters, a numerical Hessian requires *k* gradient evaluations. Also see "`option numerical standard errors`" and "`option huber`".

13.1.5. option numerical standard errors;

In order to report standard errors of parameter estimates, aML computes the covariance matrix of estimated model parameters. This covariance matrix is asymptotically given by the inverse of minus the Hessian matrix (matrix of second derivatives of the log-likelihood with respect to all model parameters):

$$\hat{\Sigma}_{\hat{\theta}\hat{\theta}'} \approx -\left(\frac{\partial^2 \ln L}{\partial\theta\partial\theta'}\right)^{-1}$$

where θ represents a vector of all model parameters and *L* the aggregate likelihood. By default, aML approximates the Hessian matrix as minus the sum over observations of the outerproduct of first derivatives:

$$\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \approx -\sum_{i=1}^{N}\left(\frac{\partial \ln L_i}{\partial\theta}\right)\left(\frac{\partial \ln L_i}{\partial\theta'}\right)$$

where L_i is the likelihood of observation *i* and *N* is the number of observations. This approximation, known as the BHHH approximation (Berndt, Hall, Hall, and Hausman, 1974), tends to be good for large sample sizes, but can be quite misleading for smaller samples. For smaller samples, and for final runs based on any sample, it is better to compute standard errors based on actual second derivatives. Because of the potentially enormously complex models that aML supports, it does not compute analytical second derivatives (but see "`option numerical search`"). Instead, you may optionally specify that you wish standard errors be based on numerical second derivatives, i.e., numerical first derivatives of analytically computed first derivatives: "`option numerical standard errors;`". Unless "option numerical search" was also specified, the BHHH approximation of the Hessian matrix is computed during the search

procedure. In that case, aML reports two sets of standard errors, one based on the "numerical Hessian" matrix and one based on the BHHH approximation.

Computing numerical second derivatives can be very time-consuming: with *k* free parameters, a numerical Hessian requires *k* gradient evaluations. Also see "`option numerical search`" and "`option huber`".

13.1.6. option huber;

As explained under "`option numerical standard errors`", aML by default approximates standard errors of parameter estimates based on the BHHH approximation of the Hessian matrix. "`Option numerical standard errors`" computes a more accurate Hessian matrix, but the resulting standard errors remain an approximation. With "`option huber`", aML will compute an estimator of the covariance matrix of parameter estimates that is robust to some types of model misspecification, such as heteroskedasticity (Huber 1967; White 1980, 1982). This estimator is also known as the robust, Huber, White, or sandwich estimator, or an estimator using a first-order Taylor series expansion.

Optionally, upon convergence, aML calculates Huber-corrected standard errors and prints them along with parameter estimates. Huber-corrected standard errors are obtained by pre- and post- multiplying the covariance matrix based on second derivatives (computed as numerical derivatives of analytic first derivatives) by the inverse of the BHHH estimate of the matrix variance-covariance matrix based on the outerproduct of (observation level) first derivatives:

$$\hat{\Sigma}_{\hat{\theta}\hat{\theta}'} \approx \left(-\frac{\partial^2 \ln L}{\partial \theta \partial \theta'} \right)^{-1} \left(\sum_{i=1}^{N} \left(\frac{\partial \ln L_i}{\partial \theta} \right) \left(\frac{\partial \ln L_i}{\partial \theta'} \right) \right) \left(-\frac{\partial^2 \ln L}{\partial \theta \partial \theta'} \right)^{-1}$$

The computation of Huber-corrected standard errors involves the computation of numerical second derivatives, and can be quite time-consuming: with *k* free parameters, a numerical Hessian requires *k* gradient evaluations. Also see "`option numerical search`" and "`option numerical standard errors`".

13.1.7. option variance-covariance matrix;

By default, aML reports standard errors of parameter estimates, but not the full variance-covariance matrix. This option allows for printing of the covariance matrix of estimated parameters.

13.1.8. option correlation matrix;

By default, aML reports standard errors of parameter estimates, but not the full correlation matrix. This option allows for printing of the correlation matrix of estimated parameters.

13.1.9. option Hessian matrix;

This option allows for printing of the Hessian matrix (matrix of second derivatives) at convergence or the end of estimation. If this option is combined with "`option huber`" or "`option numerical standard errors`", the reported Hessian matrix is computed numerically, i.e., as the numerical first derivative of analytical first derivatives of the log-likelihood. If neither of those options is specified, "`option Hessian matrix`" reports the BHHH-approximation to the Hessian matrix (see "`option numerical standard errors`").

13.1.10. option table format;

Upon convergence, aML always reports a table of parameter estimates, standard errors, and t-statistics on one line per parameter. Optionally, aML will report an additional table of parameter estimates with a standard error in parentheses under each parameter and between zero and three asterisks to indicate the level of significance in a two-sided test from zero. Three asterisks ("`***`") indicate significant at 1 percent; two asterisks ("`**`") indicate significance at 5 percent, and one asterisk ("`*`") indicates significance at 10 percent.

If combined with "`option numerical standard errors`" and/or "`option huber`", additional lines with those standard errors are added in {curly} or [square] parentheses.

Note that aML comes bundled with a `mktab` utility. The mktab command creates tables in the same was as "`option table format`", but has additional functionality for multiple models in multiple columns and for a convenient way to import tables into standard spreadsheet or word processor packages. You'll probably like it better than the table format option. See Section 15.2.

13.1.11. option starting value format;

This option causes aML to print end-of-run estimates in a format that is readily copied into a control file to begin a next model estimation.

Note that aML comes bundled with an `update` utility. The `update` command reads an output file and updates the starting values in the corresponding control file with converged estimates. You'll probably like it better than the starting value format option. See Section 15.1.

13.1.12. option observations= n;

By default, aML estimates models based on all observations in the data. You may limit the number aML uses by specifying "`option observations = n`". Only the first *n* observations will be used. The number *n* may be any non-negative integer; the default, "`0`", is interpreted as "all observations."

13.1.13. option weight = varname;

By default, aML assigns equal unit weight to all observations. This option results in weighted likelihood maximization. Variable *varname* must be a level 1 variable. Optimization is based on the weighted log-likelihood and weighted first and second derivatives. Standard errors and the variance-covariance matrix of the estimated parameters are based on the weighted second derivatives (BHHH, numerical second derivatives, or Huber).

The weight variable may take any real value, even negative ones. However, negative weights rarely make sense. Should you want to account for conditioning probabilities, use a denominator statement in your model specification (page 320). The use of negative weights to deal with conditioning probabilities will invalidate the BHHH approximation to second derivatives.

13.1.14. option normweight = varname;

Similar to "`option weight`", this option also results in weighted likelihood maximization. The difference from "`option weight`" described above is that weights are normalized to sum to the number of observations (sample size) so that the average weight is one. By default, all observations are weighted equally with unit weight.

13.1.15. option iterations = n;

This option allows control of the maximum number of iterations to try for maximum likelihood optimization. By default, aML iterates 40 times. If the convergence criteria are not met before the final iteration, the search will fail. A warning message will be

printed in the output, but parameter estimates and all requested output information will be computed even though it may be inappropriate.

13.1.16. option step range = n [to n];

The aML search procedure uses the Gauss-Newton algorithm (see pages 25, 270, and Judge et al., 1985). This procedure first checks whether the log-likelihood improves when the search direction is added to current parameter values. If the log-likelihood is improved, it attempts to step out twice the search direction; if it is improved further, it doubles the stepsize again to four times the search direction. The steps thus start with $1=2^0$ times the search direction and double to $2=2^1$, $4=2^2$, $8=2^3$, et cetera. It keeps doubling the stepsize until the log-likelihood worsens or until the maximum stepsize is reached. "`Option step range`" controls the maximum stepsize. Conversely, if the log-likelihood is worse at stepsize = 1, the algorithm tries stepsize $^1/_2$ ($=2^{-1}$); if it is still not improved, it tries $^1/_4$ ($=2^{-2}$), and so forth until the log likelihood improves or the minimum stepsize is reached. If the log-likelihood is still improved at the maximum stepsize, aML starts a new iteration; if it is still not improved at the minimum stepsize, the search stops with a warning that convergence was not achieved. The requested output is computed even though it may be inappropriate.

The minimum and maximum stepsizes may be specified as powers of 2. For example, the default for most models is:

```
option step range = -10 to 1;
```

which means that stepsizes may increase to $2^1=2$ times the search direction before starting a new iteration and may be halved to $2^{-10}=^1/_{1024}$ before terminating with a warning of "failure to improve the likelihood". The lower limit must be negative and the upper limit positive, so the range must include zero. The range may also be defined by a single integer value,

```
option step range = n;
```

in which case the range is set to "`-n to n`". For example, "`option step range=5`" is equivalent to "`option step range = -5 to 5`".

Stepping out by many search directions may improve the likelihood, but sometimes brings one or two parameters far away from their optimal values. This sometimes makes it difficult to eventually reach the optimum parameter values. We therefore recommend an upward maximum stepsize of $2^1=2$ for most models. However, searches of continuous and tobit models tend to benefit from large steps, especially when the parameters are far from the optimum. The default step range for continuous and tobit models is therefore:

```
option step range = -10 to 4;
```

so that the maximum stepsize is $2^4=16$ times the search direction.

Failure to improve the log-likelihood after stepping as far back as $2^{-10}=1/1024$ times the search direction may result from very poor starting values or underidentification of the model. If the smallest eigenvalues of the Hessian matrix that aML reports are very close to zero, the Hessian matrix is near-singular, which is indicative of an underidentified model. Failure to improve the log-likelihood may also be the result of a poor search direction. The search direction involves the matrix of second derivatives, which is by default approximated by the BHHH algorithm. This approximation can be poor in small samples. If the program reports failure to improve the likelihood, consider the "`option numerical search`", which computes the Hessian matrix more accurately (see page 270).

13.1.17. option save step;

By default, the initial stepsize of the aML search algorithm is 1, i.e., aML first checks whether the log-likelihood improves when one times the search direction is added to current parameter values. If the log-likelihood is improved, it repeatedly doubles the stepsize. Conversely, if the log-likelihood is worse at stepsize = 1, the algorithm repeatedly halves the stepsize to $1/2$ ($=2^{-1}$), $1/4$ ($=2^{-2}$), et cetera.

This option alters the search algorithm to begin search at the final stepsize from the previous iteration. For example, if the last iteration ended in stepsize 4, then the current iteration will begin with stepsize 4, and proceed to increase or decrease stepsize. Typically, the algorithm returns to stepsize 1 as the search reaches its optimum, but starting at the previously optimal stepsize may save search time. It often works well for continuous models.

13.1.18. option converge = {wgn | rfi | gn | rpc} < x [or ...];

Convergence may be based on any one of four criteria. The criteria are:

Criterion	Abbreviation	Example
weighted gradient norm	`wgn`	`wgn < .1`
relative function improvement	`rfi`	`rfi<.0001`
gradient norm	`gn`	`gn <1.2`
maximum relative parameter change	`rpc`	`rpc< .01`

The criteria are defined as follows:

- Weighted gradient norm: this criterion relates to the size of the search direction. The search direction is given by the product of the Hessian matrix and the vector of

first derivatives, $d = \left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \right)^{-1} \left(\frac{\partial \ln L}{\partial\theta} \right)$. The innerproduct of the search direction, weighted by the inverse of the covariance matrix of parameter estimates (at the optimum) is thus equal to $\left(\frac{\partial \ln L}{\partial\theta} \right)' \left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \right)^{-1} \left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \right) \left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \right)^{-1} \left(\frac{\partial \ln L}{\partial\theta} \right) =$ $\left(\frac{\partial \ln L}{\partial\theta} \right)' \left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \right)^{-1} \left(\frac{\partial \ln L}{\partial\theta} \right)$, where the Hessian is approximated using the BHHH scoring method as minus the sum over observations of the outer products of first derivatives: $\left(\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \right) = -\sum_{i=1}^{N} \left(\frac{\partial \ln L_i}{\partial\theta} \right) \left(\frac{\partial \ln L_i}{\partial\theta'} \right)$ (Berndt, Hall, Hall, Hausman, 1974).

The weighted gradient norm is defined as its square root, `wgn` = $\sqrt{\left(\frac{\partial \ln L}{\partial\theta} \right)' \left(-\frac{\partial^2 \ln L}{\partial\theta\partial\theta'} \right)^{-1} \left(\frac{\partial \ln L}{\partial\theta} \right)}$. It may be interpreted as the length of the search direction vector, corrected for the precision with which parameters are estimated.

- Relative function improvement: $\text{rfi} = \frac{\ln L_j - \ln L_{j-1}}{\ln L_{j-1}}$, where L_j is the likelihood of the *j*-th (most current) iteration.
- Maximum relative parameter change: $\text{rpc} = \max_i \left| \frac{\theta_i^j - \theta_i^{j-1}}{\theta_i^{j-1}} \right|$, where θ_i^j is the estimate of the *i*-th parameter in the *j*-th (most current) iteration . This criterion implies that the relative change threshold is met for every parameter.
- Gradient norm: $\text{gn} = \sqrt{\left(\frac{\partial \ln L}{\partial\theta} \right)' \left(\frac{\partial \ln L}{\partial\theta} \right)}$.

The default is "`option converge = wgn < .1`". Multiple criteria may be used in combination as alternatives. Examples include:

```
option converge = wgn < .01;
option converge = wgn<.01 or rfi<.000001;
option converge = wgn<.01 or gn<.156  ;
```

```
option converge = rfi<.000001 or gn<.156;
option converge = wgn<.01 or gn<.156  or rfi < .000001;
```

Multiple criteria are always alternatives, never cumulative requirements. In other words, they may only be linked by Boolean operator "`or`", not by "`and`".

Internally, aML automatically transforms some parameters. For example, standard deviations are constrained to be strictly positive, and internally the program searches on natural logarithms of standard deviations. The criteria always relate to untransformed parameters, i.e., to the standard deviation itself and not its natural logarithm.

13.1.19. option maximum specification space = n;

This option allocates memory for model specifications in your control file. Each time a building block enters a model specification requires between 2 and 7 integers internal storage. The default is 1,000. If this is insufficient, aML will instruct you to increase it.

13.1.20. option maximum model space = n;

This option allocates memory space for model specifications of every outcome in an observation. The default is 5,000. If this is insufficient, aML will instruct you to increase it.

13.1.21. option maximum hazard baseline space = n;

This option allocates memory space for derivatives of baseline survivor values at all nodes with respect to all spline slope parameters. The default is 4,000. If this is insufficient, aML will instruct you to increase it.

13.1.22. option maximum scratch data space = n;

This option allocates memory space for a scratch data array. After parsing the control file, aML reads the data file and creates a scratch data file with all outcomes, covariates, and other variables efficiently lined up. During the creation of that scratch data file, all variables for a single observation are temporarily stored in memory in a scratch data array. The default space for this array is `max(5000,2*dimX)`, where `dimX` is the size of the largest observation in the data set. If this is insufficient, aML will instruct you to increase it.

13.1.23. option maximum number of frequency categories = n;

This option specifies the maximum number of distinct values that may appear in frequency tables. For example, aML reports frequency tables of ordered probit outcomes, indirect reference numbers, et cetera. If the number of distinct values exceeds its maximum, aML will switch to accumulation of summary statistics (mean, standard deviation, minimum, maximum), rather than report a complete frequency table. The default maximum number of categories is 20.

If you specify a number that is smaller than five, aML will be unable to keep track of the number of occurrences, mean, standard deviation, minimum, and maximum value. If the number of actual categories exceeds the maximum number you specify, no information will be output.

13.1.24. option maximum number of residual draws = n;

This option allocates memory space for storing independent draw numbers of residuals. Residuals in aML are independent if their draw numbers differ. You specify those draw numbers in "`res(draw=`*`varname`*`, ...)`". In order to keep track of draw numbers that have already been used in the current observation, aML stores them in memory. By default, the maximum number of draw numbers is 100. If this is insufficient, aML will instruct you to increase it.

13.1.25. option maximum number of hazard baseline nodes = n;

This option allocates memory space for nodes of all duration splines entering a single hazard spell. aML supports multiple duration clocks in hazard spells. Their combined effect is determined by first sorting all nodes (corrected for the moment at which the respective clock started ticking). This option allocates sufficient space for that process. The default is 100. If this is insufficient, aML will instruct you to increase it.

13.1.26. option maximum number of reference numbers = n;

This option allocates memory space for storing reference numbers in the definitions of building blocks (regressor sets, parameters, distributions, et cetera). All reference numbers of all building blocks of a particular type are stored in one array. For example, you may define five regressor sets with three, four, two, zero, and zero reference numbers, for a total of nine reference numbers. These nine numbers, along with some overhead (delimiters) are stored in one array. There are similar arrays for other building block types, such as splines, parameters, and residuals. The default maximum size of the arrays is 300. If this is insufficient, aML will instruct you to increase it.

13.1.27. option maximum number of user-defined constants = n;

This option allocates memory space for user-defined constants. These include spline nodes, any number used in expressions, and many more. Indeed, operators in expressions are themselves stored as user-defined numbers. The default maximum number is 500. If this is insufficient, aML will instruct you to increase it.

13.1.28. option maximum number of hazard model splines = n;

This option allocates memory space for the number of duration splines in any one hazard spell. By default, the maximum is equal to the number of splines that you defined in the control file, plus 20. This should be sufficient, unless you use the same spline multiple times per hazard spell. If needed, aML will instruct you to increase it.

13.1.29. option scratchdsn = filename;

By default, aML creates a scratch data set in the current working directory, i.e., the directory from which the program is run. Upon completion of the program, this scratch data set is removed. It is sometimes more efficient to create the scratch file in a different location. For example, when the current working directory is on a network drive, you may wish to create the scratch data set on a local disk instead. "`Option scratchdsn`" allows you to specify the path and filename of the scratch data set. Be sure to include the path name!

The file specified in "option scratchdsn" is created permanently and will not be removed after aML terminates. You need to remove it yourself.

13.1.30. option gridfile = filename[.dct];

By default, aML writes output from grid searches (Sections 6.2 and 13.16) to the regular output (`.out`) file. Option gridfile directs the output to another file. This may be useful for subsequent analysis of the likelihood surface using a third-party software package.

If the optional gridfile name has extension "`.dct`", aML writes out the results in Stata's dictionary format. This is convenient for Stata users. Section 6.2 contains an example and sample Stata code.

13.1.31. option ensure positive definite = {yes|no};

Models in aML may involve normal distributions of any dimension. These distributions are defined with zero mean and a user-initiated covariance matrix (in the form of standard deviations and correlation coefficient). These covariance matrices, as any covariance matrix, must be strictly positive definite. At dimension three and higher, there may be combinations of correlations which cause the matrix to be non-positive definite. As aML iterates to find maximum likelihood parameters, the correlations may stray into illegitimate territory. If this happens, aML's default behavior is to reduce all correlation coefficients in equal proportion such that the matrix becomes (just) positive definite. This default behavior may be disabled by "`option ensure positive definite = no`". (Both "`option ensure positive definite`" and "`option ensure positive definite = yes`" are supported for symmetry, but result in the default and are thus without effect.)

Note that you may define distributions with the "`search=cholesky`" option. This makes aML search over covariance parameters in Cholesky-decomposed form, thereby guaranteeing that the covariance matrix remains positive definite. See page 304. Searching over Cholesky-decomposed parameters is typically the preferred way of ensuring that a covariance remains positive definite. However, the Cholesky search may not be combined with equality restrictions on parameters, such as equality of two standard deviations or two correlations (see page 303).

13.1.32. option check99999 = {yes|no};

You may specify models using indirectly referenced building blocks; see Section 13.3.4. If the reference variable evaluates to zero, the building block is excluded from the model. As a data integrity check, we recommend that you set transformation variables in duration splines and regressor splines to 99999 if the corresponding reference variable is zero. See Section 13.3.4. By default, aML will check that the corresponding transformation variables are indeed 99999 for duration and regressor splines that drop out of model specifications. This check may be disabled by specifying "`option check99999=no`". (Both "`option check99999`" and "`option check99999 = yes`" are supported for symmetry, but result in the default and are thus without effect.)

13.1.33. option version = n;

The features of aML version 2 are, for the most part, a superset of those of Version 1. Version 2 is therefore almost fully backward compatible with Version 1. Almost fully, but not entirely. The main exception is that fundamental changes were made to the parameterization of the negative binomial model (see Sections 2.7 and 13.12.). The

version option is included to make Version 2 fully backward compatible. The default setting, "`option version=2`", has no effect whatsoever. "`Option version=1`" has only two effects. First, it makes aML revert to the syntax and algorithms of Version 1 for negative binomial models. (Along with changes to the parameterization, Version 2's negative binomial syntax is different, so that attempts to run Version 2 on old control files, without the version option, will fail and result in an informative error message.) Second, it makes "`option step range = -10 to 4`" the default for all models, as it was under Version 1. Version 2's default is slightly different: the maximum stepsize is 2^4=16 for continuous and tobit models and 2^1=2 for all other models (see Section 13.1.16).

"`Option version=1`" does not preclude the use of features that are new to Version 2. For example, it may be combined with "`option numerical search`" to search on the basis of a numerically computed Hessian matrix.

13.2. Building Block Definitions

aML models must always be specified in terms of previously defined building blocks or "building blocks": regressions, distributions, splines, et cetera. (Note our terminology: you first *define* building blocks, and then *specify* models.) Conversely, each parameter to be estimated must be defined as (part of) a building block. Each building block implies a set of parameters defined for a particular purpose according to its own syntax and may be used in the specification of one or more models. Multiple building blocks of the same type may be specified (each having a different set of parameters values) and named uniquely or referenced indirectly by unique reference numbers. The following building blocks are supported: regressor sets, splines, scalar parameters, vectors, matrices, and distributions. Multiple building blocks of the same type (e.g., multiple regressor sets) may be defined as long as each is uniquely identified by name or reference number within type.

With minor exceptions, the user must provide initial starting values for all model parameters. These starting values must be given in the order in which their corresponding building blocks were defined.

The following lists all building block types and the page where their syntax, options, and implied parameters are documented in detail.

13.2.1. Define Parameter

Syntax	define parameter [*parname*]; [reference = *n*...*n*;] [range = {(0,inf) | (-1,1) | (0,1)};]
Starting Values	A parameter is a scalar; order of starting values is not applicable.
Use in a Model	parameter *parname* parameter (refvar = *varname*)

Description

A parameter may be used for a variety of purposes, and it may appear in all types of models. Most commonly, parameters are used in interactions with other building blocks. For example, one can interact regressor sets with a parameter, enabling the user to directly estimate structural parameters in a simultaneous equations model. One may also use a parameter to estimate an intercept, though this is typically done by including the number "1" in a regressor set.

Options and Features

define parameter [*parname*];

The *parname* may be any user-defined string of up to 12 alphanumeric characters or underscores ("_").

Parameters need not have a name. If it does not have a name, it must be identifiable by reference numbers, i.e., the reference option then becomes mandatory.

reference = *n*...*n*;

Reference numbers (if any) must be strictly positive integers. There may be no duplicate reference numbers across parameters or vectors. It is permissible, though, to re-use reference numbers in building blocks other than parameters and vectors.

range = {(0,inf) | (-1,1) | (0,1)};

By default, parameter values may be any real number. If desired, the range of a parameter may be restricted to three domains, as indicated in the syntax statement. The boundaries of these domains are not included. For example, suppose you wish to limit the range of parameter γ to $0 < \gamma < 1$:

```
define parameter gamma; range=(0,1);
```

Similarly, "`range=(-1,1)`" limits the parameter to $-1 < \gamma < 1$, and "`range=(0,inf)`" ensures $\gamma > 0$.

Range restrictions are implemented by internally transforming parameters. Denote by γ a user-defined parameter with limited range and by γ^* a transformation with unlimited range:

$$\begin{array}{lllll} \text{For} & 0 < \gamma < 1, & \gamma^* = \tan(\pi\gamma - \pi/2) & \Leftrightarrow & \gamma = 1/\pi \arctan(\gamma^*) + 1/2 \\ \text{For} & -1 < \gamma < 1, & \gamma^* = \tan(\gamma\, \pi/2) & \Leftrightarrow & \gamma = 2/\pi \arctan(\gamma^*) \\ \text{For} & \gamma > 0, & \gamma^* = \ln\gamma & \Leftrightarrow & \gamma = \exp(\gamma^*) \end{array}$$

Internally, aML's searches for the optimal value of γ^*, unhindered by any range restriction. In all results reports, the value of untransformed γ is reported, with suitably untransformed standard error, derivative, search direction, et cetera.

Output File

Definitions of parameters are reproduced in the output (.out) file, including restrictions and the name of the coefficient that you give it in the starting values. See Section 14.2.1.

13.2.2. Define Regressor Set

Syntax	define regressor set [*regsetname*]; [reference = *n...n*;] variables = *varlist*;
Starting Values	Each variable in *varlist* (or each transformation or interaction, see below) implies one or more parameters. These need to be initialized in the order they appear in *varlist*.
Use in a Model	regressor set *regsetname* regressor set (refvar = *varname*)

Description

A regressor set is a vector of variables, say X, forming a regression equation, $\beta'X$, for which coefficient vector β is to be estimated (or assigned a fixed value). The variables are typically just variables in the data set, but may be formed as transformations or interactions of variables. The resulting regression equation may be used in any model. Think of a regressor set as a $\beta'X$.

Options and Features

define regressor set [*regsetname*];

The *regsetname* may be any user-defined string of up to 12 alphanumeric characters or underscores ("_").

Regressor sets need not have a name. If it does not have a name, it must be identifiable by reference numbers, i.e., the reference option then becomes mandatory.

reference = *n...n*;

Reference numbers (if any) must be strictly positive integers. There may be no duplicate reference numbers across regressor sets. It is permissible, though, to re-use reference numbers in building blocks other than regressor sets.

variables = *varlist;*

Variables in a regressor set are typically simply data variables in the data set. aML recognizes one internally defined variable, "_id", which is equal to the observation's ID. Expressions, variable transformations, and interactions are allowed. This enables you to keep the size of the data set small, and facilitates experimentation with functional form specifications. Some illustrative permissible expressions and transformations are:

```
define regset BetaX;
   var = 1 (educ==1) (educ==3) log(income) sqrt(distance)
         log(x+sqrt(x*x+1))  spline(age, 20 50);
```

The very first "variable" is just the number "1"—it is probably the most commonly used expression, and captures an intercept. This illustrates an important point:

aML never assumes an intercept; it must be explicitly specified.

The `educ` transformations convert categorical variable `educ` into indicator (dummy) variables for education classes 1 and 3, respectively. Note that conditions are specified using double equality signs ("=="), similar to the conventions in C, Stata, UNIX, and more environments. The logarithm and square root transformations are self-evident. The spline transformation transforms one variable, `age`, into three variables. Each of the three new variables captures the effect of age on a particular segment: the first for ages under 20; the second for ages between 20 and 50; and the third for ages over 50.

A regressor set definition implies as many parameters as there are variables, expressions, or transformations, except that spline transformations account for multiple parameters. A spline with *n* nodes implies *n+1* segments and thus *n+1* slope parameters. The example above thus implies nine parameters. These should be initialized in the order in which they appear in the variable list.

You may interact scalar variables or expressions with spline transformations. For example: `(educ==1)*spline(age, 30 50)`. This expression would imply three parameters, as dictated by the spline transformation. It is not permissible to interact two or more spline transformations with each other.

There is no limit to combining transformations and expressions. For example,

```
min(age^3-12, sqrt(income), exp((survey-birthdt)/365.25))
```

is perfectly fine, should that make sense. (This expression evaluates to a scalar and thus implies just one parameter.) See Section 13.17 for a complete discussion of expressions and transformations.

You need not be concerned with the level at which variables are stored. Specifically, it is OK to list variables from any level in the regressor set. Also, you may use variables from different levels in an expression. For example, if `x2` is a level 2 variable and `x3` is a level 3 variable, `x2*x3` is permissible in a regressor set definition. The resulting variable varies at level 3, the lower (more disaggregated) level. (However, you should be concerned with logical rules pertaining to levels of variables. Specifically, independent variables must be at the same or higher level as dependent variables. For example, if the outcome in your model is a level 3 variable, independent variables in your regressor set may be level 1, 2, or 3 variables. A level 4 variable would not make sense. There is one exception to this rule: hazard models accept time-varying variables, i.e., variables at one level lower than the duration and censor variables.)

Note that you may define splines as building blocks and as part of a regressor set. The spline building block is almost always used to capture hazard baseline duration patterns, whereas splines that are part of a regressor set serve to allow the effect of an explanatory covariate be piecewise-linear. The underlying transformation formulas are the same.

Technical Suggestion

Suppose you wish to capture the effect of age as a piecewise-linear spline with nodes at 25 and 65 years. Allowing for an intercept and a simple shift effect of `sex`, the regressor set definition may read:

```
define regset AgeEffect;
   var = 1 (sex==1) spline(age, 25 65);
```

Data variable `sex` takes value 1 for males and 2 for females, so the parameter corresponding to `(sex==1)` measures the extent to which males differ from females. Having estimated the model, you want to allow for different age profiles for men and women. This may be achieved as follows:

```
define regset AgeEffect;
   var = 1 (sex==1)
         (sex==1)*spline(age, 25 65)
         (sex==2)*spline(age, 25 65);
```

This yields estimates of age profiles which are readily interpreted as belonging to males and females, respectively. The question arises whether the age profiles for males and females are significantly different. Since the first model is nested in the second, auxiliary program `amltest` is a convenient tool to answer this question (Section 15.3). Alternatively, the regressor set may be defined as follows:

```
define regset AgeEffect;
   var = 1 (sex==1) spline(age, 25 65)
         (sex==2)*spline(age, 25 65);
```

Mathematically, this formulation is equivalent to the preceding, and the resulting log-likelihood should be identical. The first age profile estimate relates to both males and females. The second relates to females only, i.e., it is a marginal age profile which captures the *difference* between female and male profiles. Put differently, the male profile is given by the first set of spline coefficients; the female profile is equal to the sum of the first and second sets of coefficients. Simple t-tests on the parameter estimates of the second spline tell whether males and females differ on individual age segments; their joint significance may again be determined by auxiliary program `amltest`.

13.2.3. Define Spline

Syntax

```
define spline [splinename];
   nodes = [x...x];
   [reference = n...n;]
   [intercept;]
   [effect = {right | full};]
```

Starting Values A spline definition with `n` nodes implies `1` optional intercept and `n+1` slope parameters. The optional intercept is initialized first, if included, then the slope parameters.

Use in a Model A spline may be used in two ways. First, to represent, or be part of, the baseline duration dependence (hazard models only):

```
duration spline(origin=varname, reference=splinename)
duration spline(origin=varname, refvar=varname)
```

and second, similar to a regressor set with one piecewise linear spline transformation (any type of model):

```
regressor spline(variable=varname, reference=splinename)
regressor spline(variable=varname, refvar=varname)
```

where all variable names may be expressions instead.

Description

A spline is a functional form which translates a continuous variable into a set of variables with a piecewise linear form in a regression. There are two very different uses of such a linear transformation. (1) In hazard models, the duration dependence is assumed to be piecewise linear in the log-hazard. Every duration dependence is characterized by a pattern (nodes and slopes) and an origin, i.e., the moment in time relative to the beginning of a spell at which the duration dependence clock started or will start ticking. The pattern is defined as part of the "`define spline`" statement; the moment in time at which the clock started or will start ticking is specified in the "`hazard model`" statement. (2) In any type of model (hazard, probit, continuous, binomial, etc), one may want to include a linear spline transformation of a continuous variable as a regressor. This may be done as part of a `regressor set`, or directly through the definition of a spline, which shown here. In the former case, the user specifies the variable to be transformed as part of the variable list in the regressor set definition; in the latter case, the variable name is part of the model specification.

A spline definition implies an intercept (if so defined) and slope parameters on each linear segment. The number of segments is equal to the number of nodes plus one, so the number of parameters to be initialized is the number of nodes plus one (without intercept) or plus two (with intercept). The intercept (if any) is initialized before the slopes.

There is rarely a good reason to define a spline and use it as a regressor spline. The only case that we have encountered (and for which the regressor spline was conceived) involves a model of simultaneous equations in which the latent hazard of one process enters as a covariate of another, non-hazard process. For example, consider a probit model of the decision to apply for graduate school. One of the factors affecting this choice is the hazard of becoming pregnant; a pregnancy and subsequent motherhood would make finishing the graduate program more difficult. The hazard of a pregnancy is a function of age and other variables. You would define a spline to capture the baseline duration pattern of age in the hazard of becoming pregnant. That same spline (and all other determinants of pregnancy risk) also affect the probit propensity of applying to graduate school. It would thus be used as a duration spline in the hazard equation and a regressor spline in the probit equation.

```
/* Pregnancy building blocks */
define spline AgeEffect; nodes = 18 25 35; intercept;
define regset Fertility; var = <varlist>;

/* School application building blocks */
define regset School; var = <varlist>;
define parameter Lambda;

/* Pregnancy equation */
hazard model;
   censor=censor;  duration=lower upper;
   model = regset Fertility +
           durspline(origin=age, ref=AgeEffect);

/* School application equation */
probit model;
   outcome=apply;
   model = regset School +
           par Lambda * regset Fertility +
           par lambda * regspline(var=age, ref=AgeEffect);
```

In the vast majority of cases, though, it is more convenient to capture piecewise-linear effects of covariates by including a spline transformation in a regressor set definition. Mathematically, the two approaches are equivalent.

Options and Features

define spline [*splinename*];

The *splinename* may be any user-defined string of up to 12 alphanumeric characters or underscores ("_").

Splines need not have a name. If it does not have a name, it must be identifiable by reference numbers, i.e., the `reference` option then becomes mandatory.

reference = *n...n*;

Reference numbers (if any) must be strictly positive integers. There may be no duplicate reference numbers across splines. It is permissible, though, to re-use reference numbers in building blocks other than splines.

nodes = [*x...x*];

The nodes are the points at which the effect of a continuous variable, or a duration dependence, switches to a different slope. Nodes are real numbers and are measured relative to the origin of the transformation, which is the origin (zero value) of the variable in the data that is being transformed. Nodes may thus be positive, zero or negative. However, when option "`effect=right;`" is specified, the spline is operative only after its origin (for positive values of the data-supplied variable being transformed), so that nodes must be strictly positive.

There need not be any nodes; in this case, "`nodes=;`" needs to be specified. The result is a Gompertz duration dependence, with a constant slope (i.e., one parameter, plus an intercept, if any). If the slope is fixed to zero (in the starting values), the model will have a constant log-hazard, i.e., this special case yields the double exponential model.

intercept;

By default, spline definitions do not have an intercept. Option "`intercept`" includes an intercept. It needs to be initialized in the starting values before the slope parameters.

With rare exceptions, models need to have an intercept. You typically have the choice to specify the intercept as part of a (duration) spline or of a regressor set. However, there are cases in which the spline intercept is crucial. For example, suppose that the hazard of conceiving a child should jump up as soon as the mother experiences the loss of one of her children: a 'replacement' effect. This jump may be realized by a spline duration dependence that kicks in if a child dies. This requires that its effect will

be "right", as explained below. It also requires that the spline contains an intercept, so that there is an instantaneous jump in the hazard pattern.

effect = {right | full};

This option determines whether the spline is operative over the full duration of the spell, "effect=full;", or on the positive, right-hand side of the origin variable, "effect=right;". (The origin variable is specified in the hazard model specification, not as part of the spline definition.) The default is "effect=full;".

In aML, there may be multiple hazard baseline dependencies on durations. For example, the hazard of conceiving a second child may depend on the duration since the previous birth, the woman's age (duration since her own birth), how long she has been married, calendar time (duration since an arbitrary point in history), et cetera. Each duration dependency has a particular shape (to be estimated) which is captured by a duration spline, and the way in which each contributes to the overall baseline duration pattern depends on how long its corresponding "clock" has been ticking. At issue is whether the negative part of the duration pattern should contribute to the baseline hazard.

If the spline is defined with one or more negative nodes, you will definitely want to account for its pattern before its clock starts ticking, i.e., before its corresponding origin variable. For example, you could specify a time trend with nodes in 1970 and 1990 by defining a spline with nodes at 10 and 30 years, and indicate in the hazard model specification that the clock should start ticking in 1960:

```
define spline TimeTrend; nodes = 10 30;
hazard model; <...>
   model = durspline(origin=time1950, ref=TimeTrend) ...;
```

where time1950 is a variable measuring the duration from January 1, 1950 to the beginning of the spell. The following is equivalent:

```
define spline TimeTrend; nodes = -5 15;
hazard model; <...>
   model = durspline(origin=time1975, ref=TimeTrend) +
           ...;
```

where time1975 measures the duration from 1/1/1975 to the beginning of the spell. For spells that begin before 1975, this duration is negative. (You could also specify "origin=time1950-25".) The two specifications yield equivalent results,[32] because by default, splines operate both before and after their origin.

Now suppose you wish to determine whether fertility behavior responds to the death of an older child. You could create a time-varying variable flagging whether the previous

[32] Equivalent but not identical, because the intercept will adjust to the new reference point.

child died, and include this variable in a regressor set. Alternatively, you could capture replacement behavior by a duration spline. If and when the previous child dies during the second birth interval (hazard spell), the hazard may shoot up; its pattern will indicate whether an elevated risk persists, or whether it increases or tapers off over time. If the previous child does not die, the replacement spline should never enter the equation:

```
define spline Replacement; ref=1; nodes=<...>;
   intercept; effect=right;
hazard model; <...>
   model = durspline(origin=mortdur, refvar=kiddied) +
           ...;
```

where `kiddied` is an indicator variable for whether the previous child died, and `mortdur` is the duration from the date of death to the beginning of the spell. Note that this duration will always be negative, because the spell starts at the birth date of the previous child. It is equal to minus the age at which the child died. Note that the duration spline is indirectly referenced using reference variable `kiddied`. If `kiddied`=1, the Replacement spline enters the equation; if `kiddied`=0, it does not (see Section 13.3.4; in that case, `mortdur` should preferably be 99999).

The essential feature of the replacement example is the need to specify "`effect=right`". We want the fertility hazard to be affected only *after* that moment, i.e., we want the replacement effect to "kick in" at a certain moment and not apply throughout. By default, (`effect=full`), the fertility hazard would be affected both before and after the child's death. Incidentally, the "`intercept`" option induces a jump in the fertility hazard upon the death of a child; intercept jumps are always almost desired for splines which need to kick in at a certain moment.

As may have become clear, the effect option is relevant only if the duration spline starts ticking sometime after the spell has started, i.e., if the origin variable is ever negative. Positive origin variables indicate that a clock starts ticking before the spell started, and it is then irrelevant whether its effect applies only after that moment, or before it as well.

13.2.4. Define Vector

Syntax

```
define vector [vectorname];
   [reference = n...n;]
   dimension = n;
   [range = {(0,inf) | (-1,1) | (0,1)};]
   [increasing = {yes|no};]
   [initial = {ghq weights | ghq points(std=x)};]
```

Starting Values A vector definition with dimension *n* implies *n* parameters which are initialized in the order they occupy in the vector, from 1 to *n*.

Use in a Model Vectors serve two purposes. First, to capture thresholds in ordered logit and ordered probit models:

```
thresholds = vectorname;
```

or as points or as weights in the definition of a finite mixture distribution. Note that vectors are used here as part of the definition of another building block, not directly as part of a model specification:

```
points = vectorname;
weights = vectorname;
```

Description

A vector is a set of one or more parameters used as a set. A vector may be used to specify the thresholds in ordered logit and ordered probit models or to specify the points and the weights in the definition of a finite mixture distribution.

Options and Features

`define vector [vectorname];`

The *vectorname* may be any user-defined string of up to 12 alphanumeric characters or underscores ("_").

Vectors need not have a name. If it does not have a name, it must be identifiable by reference numbers, i.e., the `reference` option then becomes mandatory.

reference = *n...n*;

Reference numbers (if any) must be strictly positive integers. There may be no duplicate reference numbers across parameters or vectors. It is permissible, though, to re-use reference numbers in building blocks other than parameters and vectors.

dimension = *n*;

This statement determines the number of parameters in the vector. There is no default. The number of starting values that needs to be initialized is equal to the number of parameters in the parameter set. A vector with unit dimension is equivalent to a parameter.

range = {(0,inf) | (-1,1) | (0,1)};

The `range` option allows the user to restrict all vector elements, similar to the range restrictions on parameters (see above). The boundaries of these ranges are not included.

increasing = {yes|no};

This option allows the parameters to be restricted to be in strictly increasing order as required by the models in which the parameters are used. The default is that strict ordering is imposed. (You may expressly specify this by "`increasing=yes`" or just "`increasing;`".) A vector may be used to specify the thresholds in ordered logit and ordered probit models or to specify the points and the weights in the definition of a finite mixture distribution. In both applications, the vector elements should be strictly increasing. By turning off the increasing restriction, you risk that the vector elements cross each other in the search procedure.

Internally, the strict increase of vector elements is achieved by transforming the second and subsequent vector elements into the natural logarithm of the difference between each parameter and its predecessor. Denote the user-defined, untransformed parameters by $\tau_1,...,\tau_n$ and the internal transformations by $\tau_1^*,...,\tau_n^*$, then:

$$\begin{aligned}\tau_1^* &= \tau_1 \\ \tau_2^* &= \ln(\tau_2 - \tau_1) \\ &\vdots \\ \tau_n^* &= \ln(\tau_n - \tau_{n-1})\end{aligned}$$

aML searches for the optimal value of $\tau_1^*,...,\tau_n^*$, unhindered by any range restriction (except possibly on τ_1^*, see next paragraph). In all results reports, the value of

untransformed $\tau_1,...,\tau_n$ is reported, with suitably untransformed standard errors, derivatives, search directions, et cetera.

The increasing option may not be used in combination with range restrictions (-1,1) or (0,1). The combination with range restriction (0,Inf) is supported. In that case, τ_1^* is transformed to be $\tau_1^* = \ln \tau_1$; see Section 13.2.1, in particular page 284.

initial = {ghq weights | ghq points(std=x)};

This option lets aML pick starting values for the vector. It should only be used for vectors that serve as weights or support points for finite mixture distributions (Section 13.2.9). Instead of specifying numerical values in the "`starting values`" statement, specify the word "`auto`".

A common research strategy for estimating a univariate distribution is to (1) assume that the distribution is normal, and estimate a standard deviation, and (2) relax the normality assumption in favor of a finite mixture distribution, and determine whether the resulting points and weights are at odds with the normality assumption. (A formal test is not available.) In step (2), you will want to initialize your parameters such that the finite mixture distribution has the same variance and zero mean as the estimated normal distribution.

For example, suppose you at first assumed the normal distribution and estimated its mean to be 1.25. You now decide to use a finite mixture with three points instead. As explained in Section 13.2.9, this requires two vectors. The first vector, of dimension 3, defines the points themselves; the second vector, of dimension 2, defines the weights corresponding to the three points. (Weights must add up to one, so only two weights are estimated.) The two vectors and their starting values may be specified as follows:

```
define vector Points;  dim=3; initial=ghq points(std=1.25);
define vector Weights; dim=2; initial=ghq weights;
define finite mixture distribution; dim=1;
   form=asymmetric;
   points=Points;
   weights=Weights;
   name=...;
<...>
starting values;
Point1    F   auto
Point2    T   auto
Point3    T   auto
Weight1   T   auto
Weight2   T   auto
<...>
```

```
;
```

Note that we fixed the first point, as it is perfectly collinear with the intercept term (not shown). aML automatically initializes the starting values such that the mean of the finite mixture distribution is zero, and its standard deviation 1.25. The points and weights correspond to those of Gauss-Hermite Quadrature, hence the "`ghq`" in the initialization options. For further details on finite mixture distributions see Section 13.2.9. aML comes bundled with auxiliary program `points`, which computes Gauss-Hermite Quadrature points and weights. For further details see Section 15.4 and Davis and Rabinovitz (1967).

13.2.5. Define Matrix

Syntax

```
define matrix matrixname;
   dimension = (n1,n2);
```

Starting Values A matrix definition with dimension `(n1,n2)` (i.e., `n1` rows and `n2` columns) implies `n1*n2` parameters which are initialized in accordance with matrix columns:

$$\begin{pmatrix} \text{first} & (n1+1)\text{-th} & \cdots & (n1*(n2-1)+1)\text{-th} \\ \text{second} & (n1+2)\text{-th} & & \vdots \\ \text{third} & \vdots & & \\ \vdots & \vdots & \ddots & \\ n1\text{-th} & (2*n1)\text{-th} & \cdots & (n1*n2)\text{-th} \end{pmatrix}$$

In other words, element `(i,j)` is initialized in position `(i+n1*(j-1))`.

Use in a Model Matrices are never used as entities; only matrix elements are used in model statements. Both the elements of a matrix and of its inverse may be used. This feature enables the specification of systems of simultaneous equations, as explained below. An element of a matrix or of an inverse- matrix may be used as follows:

```
parameter matrixname(i,j)
parameter inverse(matrixname(i,j))
```

Description

A matrix is a set of parameters that used together in simultaneous equations. You may use both elements of the matrix and of its inverse in model specifications. The direct use of matrix elements is equivalent to the use of parameters and adds little or no functionality. The use of elements of the inverse matrix allows specification of fully simultaneous models. Be warned: model specifications with inverse matrix elements are quite tricky; see Section 4.2.2.

Options and Features

`define matrix matrixname;`

The *matrixname* may be any user-defined string of up to 12 alphanumeric characters or underscores ("_").

Unlike most other building blocks, matrices must have a name. Identification of matrix elements by reference numbers is not implemented, primarily because it would not make sense with inverse matrix elements.

dimension = (n1,n2);

This statement determines the number of rows and number of columns, respectively, in the matrix. There is no default. The number of starting values that needs to be initialized is `n1*n2`.

Unlike in parameters and vectors, there is no option to impose range or cross-restrictions on the parameters that make up matrices.

13.2.6. Define Normal Distribution

Syntax

```
define normal distribution;
   dimension = n;
   [number of integration points = n [n...n];]
   name = [resname_1];  [reference = n...n;]
        :
   name = [resname_n];  [reference = n...n;]
   [restrictions ....;]
   [search = cholesky;]
```

Starting Values

A normal distribution with dimension `n` implies `n*(n+1)/2` parameters: `n` standard deviations and `n*(n-1)/2` correlation coefficients. The mean vector is always zero. The standard deviations are initialized first, followed by correlation coefficients on the lower triangle, column by column. Consider this matrix with standard deviations on the diagonal and correlation coefficients off-diagonally:

$$\begin{pmatrix} \text{first} & & & & \\ (n+1)-\text{th} & \text{second} & & & \\ (n+2)-\text{th} & (2*n)-\text{th} & \text{third} & & \\ \vdots & \vdots & \vdots & \ddots & \\ (2*n-1)-\text{th} & (3*n-3)-\text{th} & \cdots & (n*(n+1)/2)-\text{th} & n\text{-th} \end{pmatrix}$$

The covariance matrix is symmetric, and elements on the upper triangle are not initialized. Equivalently, one may view the initialization of correlation coefficients as row-by-row on the upper triangle.

If the optional cross-restrictions on standard deviations and/or correlation coefficients are imposed, only the value of the first parameter listed in (on the left-hand-side of) a restriction statement is initialized, as explained below.

Use in a Model Distributions are never used as entities; only residuals (normal variates composing the stochastic terms of the distribution) are used in model statements. Normal residuals may or may not be numerically integrated out, and may be directly or indirectly referenced:

```
[integrated] residual(draw=varname, ref=resname)
[integrated] residual(draw=varname, refvar=varname)
```

Normal residuals used in hazard, logit, binomial, and negative binomial models must be integrated out; non-integrated residuals are only allowed in continuous and probit models.

The same independent draw of a distribution may be used in multiple replications of outcomes, as controlled by the "`draw`" variable. The same value of *`varname`* triggers the same draw (same values) of the vector of normal variates for all model statements in all data structures. See Sections 4.1 and 13.3.6.

Description

A distribution is a relatively complex form of building block. It consists of one or more dimensions (residuals), each of which may be used in one or more model statements. Residuals may be directly referenced, by name, or indirectly, by a reference variable. More than one dimension of a distribution may be referenced in any one model statement.

aML automatically restricts standard deviations to be strictly positive and correlations to be strictly between -1 and 1. (For a brief technical description of the internal implementation of these restrictions, refer to the `range` option of the `define parameter` statement in Section 13.2.1.)

Options and Features

`define normal distribution;`

No name or optional items may be specified for the distribution per se. The distribution is never used as an entity and thus need not have a name. Only its dimensions (residuals) are used and may have names.

`dimension = n;`

This specifies the dimension of the distribution. It also determines the number of standard deviations and correlation coefficients that need to be initialized, unless there are restrictions (see below under "`option restrictions`"). A distribution of dimension `n` implies `n` standard deviations and `n*(n-1)/2` correlation coefficients. The user needs to specify starting values of these standard deviations and correlation coefficients, not of variances and covariances.

name = [*resname*];

The *resname* (residual name) may be any user-defined string of up to 12 alphanumeric characters or underscores ("_").

Residuals need not have a name. If it does not have a name, it must be identifiable by reference numbers, i.e., the `reference` option then becomes mandatory. However, there must always be a "`name=;`" statement, so that aML can assign reference numbers to the correct residuals.

You must specify as many residual names as there are dimensions of the distribution.

reference = *n...n*;

Reference numbers (if any) must be strictly positive integers. There may be no duplicate reference numbers across residuals, including residuals of other distributions. It is permissible, though, to re-use reference numbers in building blocks other than distributions.

Name and reference statements must always appear pairwise. In other words, a reference statement must immediately follow its corresponding residual name statement. For example:

```
define normal distribution; dim=2;
name=u1;  ref=10 20 30;
name=u2;  ref=45;
```

or:

```
define normal distribution; dim=2;
name=;  ref=10 20 30;
name=;  ref=45;
```

Note that there must be as many name statements as there are dimensions, even if one or more residuals are unnamed.

restrictions ....;

You may place equality restrictions on standard deviations and/or on correlation coefficients of distributions with at least two dimensions. Standard deviations may be restricted to be equal to each other, correlation coefficients may be restricted to be equal to each other, but a standard deviation may not be restricted to be equal to a correlation coefficient. Restrictions on standard deviations are specified as, e.g.,

```
restriction sigma(1)=sigma(2);
```

Restrictions on correlations are specified as, e.g.,

```
restriction rho(2,1)=rho(3,1);
```

More than two parameters may be restricted to be equal, e.g.,

```
restrictions rho(2,1)=rho(3,1)=rho(3,2);
```

As many restrictions may be imposed as desired, either in separate statements or in a combined statement:

```
restrictions sigma(1)=sigma(2)
             rho(2,1)=rho(3,1)=rho(3,2);
```

Each restriction reduces the number of parameters that fully determines the distribution. Be careful with the order in which standard deviations and correlations are initialized. Only standard deviations and correlations on the left-hand-side of the first equality should be initialized.

Standard deviations or correlation coefficients that enter on the right-hand-side of restriction statements (to the right of the first equality sign in a restriction statement) are not initialized in the list of starting values. Be careful with the initialization order and check the covariance matrix that is printed out in the output file to make sure that the program interpreted your restrictions and definitions correctly.

search = cholesky;

This option internally transforms standard deviations and correlation coefficients into a Cholesky decomposition of the covariance matrix. In addition, it transforms the diagonal elements of the Cholesky decomposition (which must be positive) into their natural logarithm. The maximum likelihood search procedure searches for the optimal values of the transformed covariance matrix, unhindered by any range restrictions. In all results reports, the value of the untransformed standard deviations and correlations are reported, with suitably untransformed standard errors, derivatives, search directions, et cetera.

The user specifies the covariance structure of a normal distribution in terms of its standard deviations and correlation coefficients. By default, aML imposes the restrictions that standard deviations must be positive and correlation coefficients between –1 and 1. This ensures that the implied covariance matrix is positive definite ("legitimate") for distributions up to bivariate (`dim=2`). It also helps maintain positive definiteness for trivariate and higher-dimensional distributions, but there is no guarantee. The search algorithm may generate values for the standard deviations and correlation coefficients which would imply a non-positive definite covariance matrix. When that happens, the likelihood function returns a log-likelihood equal to NaN ("not-a-number"), which is considered worse than any other value. The search algorithm will attempt to correct itself by stepping out less far, but it does not always succeed.

We suggest that you specify "`search=cholesky`" for distributions of dimension three or higher, if the maximum likelihood search procedure has trouble converging. You could always specify it for trivariate and higher-dimensional distributions, but experience shows that the search on Cholesky-decomposed parameters is sometimes less efficient than on standard deviations and correlations that have been transformed using the default transformations.

The Cholesky option may not be combined with equality restrictions on standard deviations or correlations ("`restrictions`" option, above).

Note that aML has a second line of defense against non-positive definite covariance matrices. By default, if a non-positive definite covariance is encountered, aML will proportionally reduce all correlation coefficients until the matrix is positive definite. See "`option ensure positive definite`" on page 280.

`number of integration points = n [n...n];`

The likelihood of some models with normally distributed residuals does not have a closed form solution. This is true for hazard models, (ordered, multinomial) logit models, (negative) binomial models, Poisson models, some (ordered, multinomial) probit models, etc. In those cases, aML offers an approximation: numerical integration of the residuals. The approximation's accuracy depends on the number of support points or "integration points". The more integration points, the more accurate the approximation to a normal distribution is, but also the more computations need to be performed.

Each support point requires a function evaluation. The number of support points for multivariate distributions is the product of the numbers for every dimension. For example, a trivariate distribution with six support points per dimension requires 6^3=216 function evaluations.

By default, aml uses twelve support points for univariate normal distributions, eight for both dimensions of bivariate distributions (i.e., 64 function evaluations), and six per dimension for distributions of dimension three or more. The user may specify fewer or more support points as part of the definition of the distribution. For time-consuming problems, in particular, you may wish to reduce the number of points while developing your model and revert to greater precision for the final run.

You may specify different numbers of points for different dimensions of the distribution. For example, suppose you define a trivariate normal distribution with residuals delta, eps, and eta, and you wish to approximate the three residuals by five, four, and two integration points:

```
define normal distribution; dim = 3;
   number of integration points = 5 4 2;
   name = delta;
   name = eps;
```

```
name = eta;
```

If the number of integration points is the same for all dimensions, you may specify just one number. In other words, the following are equivalent:

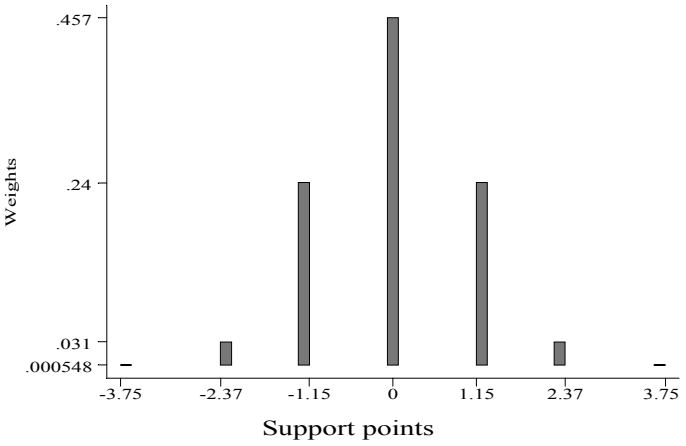
```
define normal distribution; dim = 3;
   number of integration points = 5 5 5;
```

and

```
define normal distribution; dim = 3;
   number of integration points = 5;
```

The points are chosen using Gauss-Hermite Quadrature. Suppose you specify a normal distribution with k dimensions and n integration points in each dimension. aML first computes n standardized points which approximate a univariate standard normal distribution. These follow from zero solutions of the Hermite polynomial of order n, and weights are found as corresponding coefficients of the n-th order Gauss-Hermite Quadrature formula of degree $2n$-1 (Davis and Rabinovitz, 1967). This method is extended to handle k-variate integration by selecting all n^k possibilities of k draws out of n points (with replacement). The appropriate weight for each set is the product of k individual (univariate) weights. These n^k sets of points and weights approximate a k-variate standard normal. In order to approximate other k-variate normals (with zero means), aML premultiplies each set of k draws by the Cholesky decomposition of the covariance matrix of the distribution. The marginal likelihood is computed as the properly weighted accumulation of n^k conditional likelihoods.

The figure illustrates the points and weights that are chosen for a univariate distribution with seven integration points. The more points, the closer the approximation to a normal distribution. aML comes bundled with auxiliary program `points`, which computes points and weights for any number of integration points; see Section 15.4.



Residuals from a distribution which is defined with a number of integration points may be integrated out, but they need not be. For example, suppose you estimate a multilevel selection model with a continuous and a probit equation. Both equations include a person-specific normal residual (heterogeneity component) which is common over multiple replications of the outcomes of interest. You may wish to integrate-out the probit heterogeneity component, but ask aML to compute the closed-form solution to the continuous modules. aML will first compute the distribution of the probit residual conditional on the continuous outcomes, and then integrate numerically. The bivariate distribution then essentially implies a univariate integration, which reduces the number of calculations that needs to be performed..

The computational burden increases roughly linearly with the number of points in a univariate distribution, roughly quadratically in a bivariate distribution, et cetera. For models with multivariate integrated distributions, we recommend that you specify a relatively low number of integration points (say, four in each dimension) during the model exploration stage, and move to a higher level of accuracy (say, twelve points in each dimension) for production runs.

13.2.7. Define ARMA(p,q) Distribution

Syntax

```
define {ar(p) | ma(q) | arma(p,q)} distribution;
   [dimension=1;]
   timevar [(within level n)] = varname;
   [increasing={yes|no};]
   [ardomain = {(-1,1) | (0,1)};]
   [madomain = {(-1,1) | (0,1) | (0,Inf)};]
   name = [resname]; [reference = n...n;]
```

Starting Values The first p parameters initialized are the autoregressive coefficients; the next parameter is the standard deviation of the innovation term; the last q are moving average coefficients.

Use in a Model The distribution is not used as an entity; only the stochastic term is used in model statements, by reference to its name or by reference number. An ARMA residual may only be used in continuous models, and may not be integrated out. The syntax for direct and indirect reference is:

```
residual(draw=_iid, ref=resname)
residual(draw=_iid, refvar=varname)
```

A time series vector of values of the normal residual term represented by the distribution are correlated according to the ARMA(p,q) model. Independent draws of random vectors from the same distribution may be used in multilevel modeling. All innovation terms must be drawn independently (`draw=_iid`).

Description

aML supports normally distributed ARMA(p,q) processes with arbitrary autoregressive and moving average orders.[33] Only univariate ARMA distributions are supported. ARMA distributions are defined in a similar manner as other distributions, and the corresponding residuals are used in a similar manner, too. ARMA distributions may only be used in continuous models. The general formulation of an ARMA(p,q) process is:

$$v_t = \sum_{i=1}^{p} \phi_i v_{t-i} + e_t + \sum_{j=1}^{q} \theta_j e_{t-j} \ ,$$

where $\phi_1,\ldots,\phi_p$ are autoregressive coefficients, e_t is the innovation term, and $\theta_1,\ldots,\theta_q$ are moving average coefficients.

[33] Subject to hardwired maximum orders of $p \le 9$ and $q \le 9$.

Since v_t depends on v_{t-1} and prior values of the residual, you must specify a variable that contains "time points" as part of the definition of an ARMA distribution. That variable must be at the same level as the continuous outcome variable. These time points are used to determine the lags between successive continuous records. The time points must be integer-valued, except for AR(1) distributions in which the autoregressive parameter is merely restricted to be positive. Usually, points in time will be spaced one period apart, but points in time may be missing (omitted from the data) so that points in time are unequally spaced. The time points must be strictly increasing.

An ARMA(p,q) distribution has p+1+q parameters. They are initialized in this order: $(\phi_1,\ldots,\phi_p,\sigma_e,\theta_1,\ldots,\theta_q)$. Note that, in accordance with the convention for other types of distributions, the standard deviation of the innovation term is initialized and estimated, not the variance.

Options and Features

define {ar(*p*) | ma(*q*) | arma(*p*,*q*)} distribution;

This statement specifies the autoregressive and/or moving average order. No distribution name may be specified.

dimension=1;

ARMA distributions always have only one dimension. For consistency with the normal distribution, a dimension statement may be specified, but it is optional.

ardomain = {(-1,1) | (0,1)};
madomain = {(-1,1) | (0,1) | (0,Inf)};

The optional `ardomain` and `madomain` statements restrict the autoregressive and moving average parameters to be in the indicated ranges. The default is that all autoregressive parameters are within `(-1,1)`, and that moving average parameters are unrestricted. If the `timevar` is not integer-valued, `ardomain` must be restricted to `(0,1)`; this is only valid in AR(1) distributions. The domain restrictions are strict, i.e., boundary values (-1, 0, or 1 as relevant) are excluded from the domain.

Internally, the restrictions are implemented through parameter transformations, as explained in Section 13.2.1, particularly page 284.

name = [*resname*];

The *resname* (residual name) may be any user-defined string of up to 12 alphanumeric characters or underscores ("_").

ARMA residuals need not have a name. If it does not have a name, it must be identifiable by reference numbers, i.e., the `reference` option then becomes mandatory. However, there must always be a "`name=;`" statement.

reference = *n...n*;

Reference numbers (if any) must be strictly positive integers. There may be no duplicate reference numbers across residuals, including residuals of other distributions (normal, ARMA, finite mixture, or other). It is permissible, though, to re-use reference numbers in building blocks other than distributions.

timevar [(within level n)] = varname;

Each continuous outcome occurs at a point in time, forming a vector of time series observations within a continuous model statement. The values of the variable *varname* are used to determine the timing of each outcome in the vector. These values of time must be integer values, except in the AR(1) distribution. Distances in time between points are calculated as the differences in the values of *varname* specified for each point. The values of the continuous outcome variable named in the model statement are linked by their respective points in time. The time distance between pairs of points determines the correlation (and covariance) between the two corresponding outcome values. These correlations are well defined even when there are time gaps in the time series.

The `within` specification of the `timevar` statement indicates whether all residuals are autocorrelated, or whether autocorrelation only exists within branches of a certain level. The default is `(within level 1)`, i.e., all residuals are autocorrelated. See below for more details.

increasing={yes|no};

By default, aML checks that the time variable is strictly increasing within its level. If for some reason your data are not ordered in increasing time, you may turn off this check with "`increasing=no`".

Autocorrelation within level

The default is that all residuals within a certain observation are autocorrelated. However, it may be the case that residuals are only autocorrelated within well-defined subsets of the data, or within certain time-intervals, and uncorrelated across those subsets or intervals. This is controlled by the `within` specification of the `timevar` statement.

Consider an example. Suppose one wishes to estimate an income-generating process with annual data from 1980 through 1988 (nine years). Variable `time` takes corresponding values

1980 through 1988 (or 80 through 88, or other sets of nine contiguous integer values). The observation thus contains nine outcomes. Suppose the observation is divided into two level 2 branches; the first branch contains two outcomes (annual income records), and the second branch is in turn subdivided into two level 3 branches with four and three outcomes each. Denote the variance of the ARMA(p,q) process by γ_0 and the first through eighth autocovariances by γ_1 through γ_8.

The default is that all residuals are autocorrelated, "`timevar = time;`", so that the covariance matrix is given by:

$$\Sigma = \begin{pmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 \\ \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 \\ \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 \\ \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 \\ \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \gamma_5 & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\ \gamma_6 & \gamma_5 & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 \\ \gamma_7 & \gamma_6 & \gamma_5 & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 \\ \gamma_8 & \gamma_7 & \gamma_6 & \gamma_5 & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 \end{pmatrix}$$

If the time variable was specified as "`timevar (within level 2) = time;`", the residuals would be independent across level 2 branches:

$$\Sigma = \left(\begin{array}{cc|ccccccc} \gamma_0 & \gamma_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma_1 & \gamma_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 \\ 0 & 0 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 \\ 0 & 0 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ 0 & 0 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\ 0 & 0 & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 \\ 0 & 0 & \gamma_5 & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 \\ 0 & 0 & \gamma_6 & \gamma_5 & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 \end{array}\right),$$

and if "`timevar (within level 3) = time;`" were specified, there would be no autocorrelation across level 3 subbranches:

$$\Sigma = \left(\begin{array}{cc|cccc|ccc} \gamma_0 & \gamma_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma_1 & \gamma_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & 0 & 0 & 0 \\ 0 & 0 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & 0 & 0 & 0 \\ 0 & 0 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & 0 & 0 & 0 \\ 0 & 0 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & \gamma_0 & \gamma_1 & \gamma_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1 & \gamma_0 & \gamma_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \gamma_2 & \gamma_1 & \gamma_0 \end{array}\right).$$

13.2.8. Define Cumulative AR(1) Distribution

Syntax

```
define cumulative ar(1) distribution;
timevar [(within level n)]  = varname;
   [ardomain = {(-1,1) | (0,1)};]
   name = [resname];  [reference = n...n;]
```

Starting Values Similar to the AR(1) distribution, the first parameter is the autoregressive coefficient and the second is the standard deviation of the innovation term.

Use in a Model The distribution is not used as an entity; only the stochastic term (residual) is used in model statements, by reference to its `name` or by reference number. A CAR(1) residual is used exactly as an AR(1) residual. It may only be used in continuous models, and may not be integrated out. The syntax is for direct and indirect reference is:

```
residual(draw=_iid, ref=resname)
residual(draw=_iid, refvar=varname)
```

A time series vector of values of the normal residual term represented by the distribution are correlated according to the CAR(1) model. Independent draws of random vectors from the same distribution may be used in multilevel modeling. All innovation terms must be drawn independently (`draw=_iid`).

Description

The Cumulative AR(1) is very similar to the AR(1) distribution. The difference is that a cumulative AR(1) process is assumed to start at t=1, i.e., when the time variable "`timevar`" *varname* takes the value one, whereas AR(1) processes are assumed to have an infinite history. The definition of the variable representing points in time in the data must be defined in reference to the initial period, the time when the process began. Unlike in AR(1) models, you may thus not measure time relative to an arbitrary point in history. There need not be an actual record with "`timevar`" variable *varname*=1; the process is assumed to start at that point, but the corresponding record may be missing.

AR(1) versus CAR(1)

Both the AR(1) and the CAR(1) distributions imply two parameters: the autoregressive correlation and the standard deviation of the innovation term. Denote

$$v(t_2) = \rho^{|t_2 - t_1|} v(t_1) + u(t_2),$$

where ρ is the autoregressive parameter and *u* is the innovation. The standard deviation of the innovation is σ_u. The essential difference between the AR(1) and the CAR(1) is that the AR(1) distribution assumes an infinite history of the innovations so that the distribution is time-stationary with equal variances over time and equal covariances and correlations among equally distanced periods. (Alternatively, the AR(1) process assumes that its variance in the initial period (only) is $\sigma_v^2 \Big/ \left(1-\rho^2\right)$ so that the process is jump-started into stationarity.) The AR(1) is thus characterized by:

$$\sigma_v^2 = \sigma_u^2 / \left(1+\rho^2\right) \quad \text{and} \quad \sigma_{v_\tau v_t} = \sigma_u^2 \rho^{|t-\tau|} \quad \text{and} \quad \rho_{v_\tau v_t} = \rho^{|t-\tau|}.$$

Instead. the CAR(1) distribution assumes a finite history of the innovations beginning at period 1 so the distribution is not time stationary—the variances and covariances and correlations among equally distanced periods increase over time:

$$\sigma_{v_t}^2 = \sigma_u^2 \frac{1-\rho^{2t}}{1-\rho^2} \quad \text{and} \quad \sigma_{v_\tau v_t} = \sigma_u^2 \rho^{t-\tau} \frac{1-\rho^{2\tau}}{1-\rho^2} \quad \text{and} \quad \rho_{v_\tau v_t} = \rho^{t-\tau} \sqrt{\frac{1-\rho^{2\tau}}{1-\rho^{2t}}}$$

where $t \geq \tau \geq 1$. Note that for the CAR(1), $\rho_{v_\tau v_t} \leq \rho^{t-\tau}$ and $\rho_{v_\tau v_t} \to \rho^{t-\tau}$ as $t, \tau \to \infty$, for given $|t-\tau|$.

Options and Features

All options and features are identical to those of the AR(1) process (described above).

13.2.9. Define Finite Mixture Distribution

Syntax

```
define finite mixture distribution;
   [dimension = 1;]
   form = asymmetric;
   points = vectorname;
   weights = vectorname;
   name = [resname];  [reference = n...n;]
```

Starting Values The definition of a finite mixture distribution does not imply any new parameters. It is determined entirely by its points and weights, which follow from previously defined vectors.

Use in a Model Distributions are never used as entities; only the implied stochastic term is used in model statements. A finite mixture stochastic term must always be integrated out. The syntax for direct and indirect references is:

```
integrated residual(draw=varname, ref=resname)
integrated residual(draw=varname, refvar=varname)
```

Independent draws of the finite mixture stochastic terms may be used in multilevel models. Values from the same draw may be linked by the use of a common value of a variable or expression, "`draw=varname`".

Description

aML supports univariate asymmetric finite mixture distributions. These may only be used as integrated residuals. The definition of a finite mixture distribution requires two vectors that have been previously defined. The first vector represents the finite mixture points; the second (is transformed to represent) weights. There are several features that need improvement; the current implementation is preliminary. In particular, the way in which weights are specified is cumbersome, and symmetric finite mixture distributions have not yet been implemented.

Consider the following example which defines a finite mixture distribution with four points:

```
define vector Points;  dim=4; increasing=yes;
define vector Weights; dim=3; increasing=yes;
define finite mixture distribution; dim=1;
   form=asymmetric;
   points=Points; weights=Weights;
   name=eps;
```

Note that vector `Weights` only contains three points. The weights are computed as follows:

$$wt_1 = \Phi(\tau_1)$$
$$wt_2 = \Phi(\tau_2) - \Phi(\tau_1)$$
$$wt_3 = \Phi(\tau_3) - \Phi(\tau_2)$$
$$wt_4 = 1 - \Phi(\tau_3)$$

where τ_1, τ_2, and τ_3 represent the three parameters in parameter set `Weights`. (It would be more elegant to define `Weights` in terms of probabilities; this is yet to be implemented.) Since weights are restricted to sum to one, only three parameters are needed to specify four weights.

A common research strategy for estimating a finite mixture distribution is to (1) assume that the distribution is normal, and estimate a standard deviation, and (2) relax the normality assumption in favor of a finite mixture distribution. In step (2), you will want to initialize your parameters such that the finite mixture distribution has the same variance and zero mean as the estimated normal distribution. aML offers a convenient way of doing this.

For example, suppose you at first assumed the normal distribution and estimated its mean to be 1.25. You now wish to use a finite mixture with four points instead. This requires two vectors. The first vector, of dimension 4, defines the points themselves; the second vector, of dimension 3, defines the weights corresponding to the three points. The two vectors and their starting values may be specified as follows:

```
define vector Points;  dim=4; initial=ghq points(std=1.25);
define vector Weights; dim=3; initial=ghq weights;
define finite mixture distribution; dim=1;
   form=asymmetric;
   points=Points;
   weights=Weights;
   name=...;
<...>
starting values;
Point1    F   auto
Point2    T   auto
Point3    T   auto
Point4    T   auto
Weight1   T   auto
Weight2   T   auto
Weight3   T   auto
<...>
;
```

Note that we fixed the first point, as it is perfectly collinear with the intercept term (not shown). aML automatically initializes the starting values such that the mean of the finite mixture distribution is zero, and its standard deviation 1.25. The points and weights correspond to those of Gauss-Hermite Quadrature, hence the "`ghq`" in the initialization options. aML comes bundled

with auxiliary program `points`, which computes Gauss-Hermite Quadrature points and weights. For further details see Section 15.4 and Davis and Rabinovitz (1967).

Identification

Special care must be exercised in making sure that your model is identified. Since the finite mixture distribution is not symmetric, it will generally have a non-zero mean. This implies that one cannot estimate both a regression intercept and all finite mixture points. In the example above, we therefore fixed the first point at its starting value by specifying an "`F`". Alternatively, you may omit or fix the intercept term, or fix one of the finite mixture points to any value. However, you may not fix one finite mixture point to a value of your choice (say, zero) in combination with asking aML to generate starting values automatically.

13.3. Model Specifications—General

All models must be specified in terms of previously defined building blocks. This allows for handling of parameter restrictions across models. For example, if a $\beta'X$ appears in multiple models (with the same coefficients β), it only needs to be defined once ("`define regressor set`") and may be subsequently used in multiple model specifications.

Each equation in your model is specified in one or more model statements. Even two or more equations that are part of one system of simultaneous equations must be specified in separate model statements. (Indeed, strictly speaking, we should speak of equation statements rather than model statements.) Correlation across equations follows from correlated residuals, which are defined as part of distributions outside the model specifications.

The syntax of model statements is:

```
<type> model;
   [data structure=n;]
   [{keep|drop} if condition;]
   [numerator;]  [denominator;]
   <outcome specification>
   <explanatory specification>
```

The current version of aML support the following types of models: continuous, hazard, (ordered) probit/logit and (negative) binomial. This section describes model specification features that are common to all model types. Section 13.3.1 documents data structure statements, keep/drop conditions, and numerator/denominator statements, which are identical for all model types. Section 13.3.2 discusses the use of building blocks in explanatory specifications, i.e., specifications of the "right-hand-side" of your model equations. Direct and indirect references to building blocks are described in Sections 13.3.3 and 13.3.4, respectively, and Section 13.3.5 documents building block interactions. Features that are specific to individual model types are documented in subsequent sections.

13.3.1. Options Common to All Model Statements

This section describes several features that are common to all models, including data structure specifications, keep and drop statements, and likelihood formulation in the presence of conditioning model statements.

Statement	Description
`data structure = n;`	Data structure number to which the model applies. Mandatory if data are organized in data structures, and not allowed if data are not organized in data structures.
`[keep if condition;]`	Keep only data for which the condition is true (i.e., equal to 1)
`[drop if condition;]`	Drop data for which the condition is true (i.e., equal to 1)
`[numerator;]`	Use model statement in numerator of likelihood function
`[denominator;]`	Use model statement in denominator of likelihood function, for conditioning events

`data structure = n;`

As discussed in Section 9.2, your data may be organized in data structures. A data structure is a subset of variables in the data. It contains both outcome variables and explanatory variables. For a number of reasons it may be convenient or necessary to create subsets of variables into distinct data structures. Data structures do not have a name; they may be identified by a unique strictly positive integer number instead.

If your data are organized in data structures, you must specify the data structure which contains the outcomes that you wish to model. This is done in the "`data structure=n`" statement, where `n` is the data structure number. The data structure statement is irrelevant if your data are not organized in data structures.

We stated above that each equation in your model must be specified in one or more model statements. Why not in just one model statement? The answer lies in the way you organized your data. If you decided to assign, say, males and females into separate data structures, but the same model equation applies to both sexes, then you need to specify your model twice, once for each data structure. In other words, you need a model statement for each data structure that contains your outcome of interest. (Some people like to create many separate but similar data structures. The use of macros may then help reduce user errors in repetitive model specifications; see Section 16.2.)

{keep | drop} if *condition;*

Model specifications may contain a keep or drop option. This is useful if you wish to include only a subset of outcomes in the model, or if you want to specify different models for various outcome types.

The "*condition*" must be a numerical expression which evaluates to either zero or one. It may contain any of the operators that aML's expressions allow (Section 13.17), including the Boolean "and", "or", and "not" (but excluding "spline", as it would evaluate to a vector). Examples include:

```
keep if sex==1;
drop if (sex!=1);
keep if (age>=18 and male==1);
drop if (x*log(y*min(v,w))<z);
```

Note that equality tests require double equality signs ("=="). Also note that, in the second through last example, we enclosed the conditions in parentheses. This is not required, but we like it to improve readability.

Keep/drop conditions may help keep your data compact. For example, suppose you wish to estimate a two-equation model of earnings. The first equation is a probit for whether someone earned anything at all; the second is a continuous model to explain the amount (in logarithmic form). The equations may be specified as follows:

```
probit model;
   outcome = (earnings>0);
   model = ...;

continuous model; keep if earnings>0;
   outcome = log(earnings);
   model = ...;
```

Many more applications are possible, including switching regressions models.

For logical reasons, variables in keep/drop statements may only be at the level of the outcome or higher (more aggregated). Consider an example of school test scores, where level 1 corresponds to schools, level 2 to students, level 3 to years in school, and level 4 to individual tests. Whether a student passed to the next grade level is a level 3 variable. It would not make sense to use a level 4 variable (such as the subject area of an individual test) in a keep or drop statement.

[numerator;] [denominator;]

In some applications, it may be necessary to condition on the probability (likelihood) of inclusion in the sample. A classic example is truncated regression; the continuous dependent

variable in a standard regression model is observed only if it exceeds (or falls below) a known threshold, and is not included in the data otherwise. For example, in a sample of recent car buyers, expenditures on cars are always positive. Individuals with zero expenditures (non-owners) are not in the sample. The likelihood function for each observation in this type of model is given by the ratio of the normal density to the probability of sample inclusion conditional on the values of the explanatory variables. This concept of conditioning on the probability of sample inclusion, by dividing by its probability, is generalized for all model statements with a "`denominator`" statement to indicate that it is a conditioning event. Its probability (likelihood) enters in the denominator of the overall likelihood.

By default, of course, the likelihood of a module enters into the numerator of the overall likelihood function. You may explicitly specify "`numerator`"; the result is the same as when you omit it. In some cases, you may want a module's likelihood to enter both into the numerator and the denominator of the overall likelihood function. This is achieved by specifying both "`numerator`" and "`denominator`". In that case, "`numerator`" must be explicitly specified.

Consider the example of a truncated regression of car expenditures. This may be modeled as:

```
probit model;
   denominator;  /* condition on prob of spending something */
   outcome = (expen>0);
   model = ...;

continuous model;
   outcome = expen;
   model = ...;
```

The sample only includes recent car buyers, so the car expenditures variable `expen` is always strictly positive. It may seem silly to specify "`outcome=(expen>0)`" in the probit model; why not simply "`outcome=1`"? The answer is that, as explained below, outcomes must always be specified in terms of variables, not constants. aML needs to determine the level of the outcome, so that it knows how many replications to model. A constant would not provide that information.

Consider a more complex example. You wish to model the hazard of divorce using a sample of individuals that were married at the time of the first interview, t_1, and that were interviewed several more times. Without heterogeneity, the likelihood that the marriage survived through time *t* may be written as

$$L(t) = \frac{S(t|t_0)}{S(t_1|t_0)} = \frac{S(t|t_1)S(t_1|t_0)}{S(t_1|t_0)} = S(t|t_1)$$

where t_0 is the wedding date and $S(t|t_1)$ is the value of the survival function at time *t*, given that the respondent was married at time t_1. The period from wedding date to the first interview, t_0 to t_1, drops out of the equation and may be ignored. But suppose you wish to control for unobserved

heterogeneity. With heterogeneity component ε, the conditioning probability does not cancel anymore:

$$L(t) = \frac{\int S(t|t_0,\varepsilon)dF(\varepsilon)}{\int S(t_1|t_0,\varepsilon)dF(\varepsilon)} = \frac{\int S(t|t_1,\varepsilon)S(t_1|t_0,\varepsilon)dF(\varepsilon)}{\int S(t_1|t_0,\varepsilon)dF(\varepsilon)}$$

where $F(\varepsilon)$ is the cumulative density function of heterogeneity component ε, and all integrals run from $-\infty$ to ∞. The likelihood function cannot be simplified anymore, and the probability that the marriage survived from wedding date to first interview must be modeled. (It is a hazard outcome which is always censored.) You could create two data structures 100 and 200 for the period from wedding date to first interview and for the period from wedding date to *t* (dissolution, death, widowhood, or last interview), respectively, and specify the model as follows:

```
hazard model; data structure=100;  /* wedding to interview */
   denominator;
   <other statements>

hazard model; data structure=200;  /* wedding to t */
   numerator;  /* may also omit this statement */
   <other statements>
```

This specification corresponds to $\int S(t|t_0,\varepsilon)dF(\varepsilon) \Big/ \int S(t_1|t_0,\varepsilon)dF(\varepsilon)$ in the equation above. Equivalently, you could create two data structures 100 and 300 for the period from wedding date to first interview and for the period from first interview to *t* (dissolution, death, widowhood, or last interview), respectively, and specify the model as follows:

```
hazard model; data structure=100;  /* wedding to interview */
   numerator; denominator;
   <other statements>

hazard model; data structure=300;  /* interview to t */
   numerator;  /* may also omit this statement */
   <other statements>
```

This specification corresponds to $\int S(t|t_1,\varepsilon)S(t_1|t_0,\varepsilon)dF(\varepsilon) \Big/ \int S(t_1|t_0,\varepsilon)dF(\varepsilon)$.

13.3.2. Use of Building Blocks in Models

Each model (equation) consists of an outcome (dependent variable) and a explanatory portion (independent variables, right-hand-side of the equation). The right-hand-side of the equation must be specified in terms of building blocks that have been defined elsewhere: regressor sets, (integrated) residuals, duration splines, parameters, regressor splines, (inverse) matrix elements, and simple variables. This section describes the rules governing the use of building blocks in model specifications.

The right-hand-side of hazard, continuous, and (ordered) probit/logit equations is specified as follows:

```
model = <building block {+|-} building block {+|-} ...>;
```

The syntax is slightly different in binomial, Poisson, negative binomial, multinomial probit, and multinomial logit models, but in all cases, the explanatory portion(s) of the model consist of one or more building blocks that are added or subtracted. Building blocks may also be interacted with each other, thereby allowing for nonlinear models, models with random coefficients, and more.

A single building block may be used in multiple data statements, i.e., it may enter in multiple equations. This enables you to impose restrictions on coefficients across equations.

13.3.3. Directly Referenced Building Blocks

This section documents how building blocks may be "directly referenced" in model statements. A directly referenced building block is a building block that enters a model unconditionally and that is used by specifying its name. By contrast, indirectly referenced building blocks enter models conditional on the value of a data variable; see Section 13.3.4.

The following documents direct referencing of regressor sets, parameters and (inverse) matrix elements , residuals and integrated residuals, duration splines, and regressor splines.

Regressor Sets

```
regressor set regsetname
```

Regressor sets may be used in all model specification statements. Think of a regressor set as a $\beta'X$. The same regressor set may enter in any combination of models at the same time. Multiple regressor sets may be used in the same model specification statement. Regressor sets may be interacted with each other and with all other building blocks.

Parameters and (Inverse) Matrix Elements

```
parameter parname
parameter matrixname(i,j)
parameter inverse(matrixname(i,j))
```

Parameters may be used in all model specification statements. The same parameter may enter in any combination of models at the same time. Multiple parameters may be used in the same model specification statement. Parameters may be interacted with each other and with all other building blocks. The rules for matrix elements and inverse matrix elements are the same as for parameters.

Residuals and Integrated Residuals

```
residual(draw=varname, reference=resname)
integrated residual(draw=varname, reference=resname)
```

Where "*varname*" may be an expression possibly involving multiple variables. It may also be a constant. For logical reasons, the draw variable must be at at least the same level as the outcome variable, i.e., at the same level or more aggregated.

(Integrated) residuals may enter in all model specifications, including incidence but not dispersion specifications in negative binomial models. The same residual may enter in multiple model specification. Multiple residuals may be used in the same model specification. Residuals

may not be interacted with each other. They may also not be interacted with duration splines. Other interactions are allowed.

Residuals may come from normal, finite mixture, ARMA, and CAR distributions. Finite mixture residuals must always be integrated out. ARMA and CAR residuals may never be integrated-out numerically. Normal residuals must be integrated-out numerically in models for which no closed-form solution to the likelihood exists: hazard, (ordered) logit, and (negative) binomial models. In continuous and (ordered) probit models, you have a choice between computing the closed-form solution or numerical integration. However, if the overall covariance matrix of (ordered) probit models is not diagonal and of dimension four or higher, numerical integration is required.

It is very important to specify which residuals are independent and which share the same draw. This is explained in detail in Section 13.3.6. In summary, independence is specified through draw variables. If the same residual (heterogeneity component) applies to multiple outcomes, then the corresponding draw variables must have the same value. By contrast, if residuals are independent, their draw variables must take on distinct values. Only residuals in the same observation may be correlated, i.e., residuals in different observations are always independent regardless of their draw variables. In addition to the syntax above, aML supports a way to specify that all draws are independent:

```
residual(draw=_iid, reference=resname)
integrated residual(draw=_iid, reference=resname)
```

Note the "`_iid`" keyword. It is not a variable. The innovation term of ARMA and CAR residuals must always be drawn independently (`draw=_iid`).

Let's take an example. In an analysis of test scores, you may wish to specify a (level 1) school effect, a (level 2) student effect, a (level 3) schooling year effect, and a (level 4) test effect; the latter captures whatever is not absorbed by covariates and higher-level residuals and is often called the "transitory" effect. Denote schools, students, schooling years, and tests by subscripts i, j, k, and l, so that the model for test score Y_{ijkl} is:

$$Y_{ijkl} = \beta' X_{ijkl} + \delta_i + \varepsilon_{ij} + \eta_{ijk} + u_{ijkl} .$$

Your aML data contain school IDs (`schoolID`), student IDs (`personID`), and school years (`year`). Student IDs are unique within school, but may contain duplicates across school. School year is, say, grade level, which ranges from 1 to 12. Test scores are in level 4 variable "`score`". The model may be specified as:

```
define regset BetaX; var = ...;
define normal distribution; dim=1; name=delta;
define normal distribution; dim=1; name=eps;
define normal distribution; dim=1; name=eta;
define normal distribution; dim=1; name=u;
```

```
continuous model;
   outcome = score;
   model = regset BetaX +
   res(draw=1, ref=delta) +
   res(draw=personID, ref=eps) +
   res(draw=100*personID+year, ref=eta) +
   res(draw=_iid, ref=u);
```

We wrote "`draw=1`" in the representation of δ_i. The unit of observation is a school, and there is thus only one draw for δ_i in any observation. We could have written "`draw=10`" or "`draw=143`" and even "`draw=schoolID`" with same result; all that matters is that the draw is the same for all replications (test score equations). For ε_{ij}, we wrote "`draw=personID`", as this will be the same for all tests of a particular student, and different across students. For η_{ijk}, we wrote "`draw=100*personID+year`". If we had written "`draw=year`", the same draw would apply to all students in the same grade level. The addition of "`100*person`" makes the draw unique across years and students. (The highest value of year is 12, so that "`100*personID+year`" uniquely maps `personID` and `year`. We could have chosen any number above 12, such as "`13*personID+year`".) For u_{ijkl}, we wrote "`draw=_iid`". We could have created some other unique number, such as "`10000*personID+100*year+testnum`", where `testnum` is the test number (`testnum`<100), but shorthand "`_iid`" is more convenient.

Duration Splines

`duration spline(origin=`*`varname`*`, reference=`*`splinename`*`)`

Where "*`varname`*" may be an expression involving possibly multiple variable names. It may also be a constant. For logical reasons, the origin variable must be at at least the same level as the outcome variable, i.e., at the same level or more aggregated.

Spline building blocks may be used as duration splines or regressor splines (see below). Duration splines may only be used in hazard models. They serve to specify the baseline duration pattern (baseline log-hazard). The same spline may enter in any combination of hazard models at the same time. There must be at least one duration spline in a hazard model. If there are multiple duration splines, they combine additively to form the overall baseline duration pattern. Duration splines may not be interacted with each other. They may be interacted with parameters, regressor sets, and regressor splines.

Each spline corresponds to a certain pattern, as defined by the user with nodes and slopes. The pattern may start at the beginning of the hazard spell, or it may start earlier or later. In aML parlance, each duration dependency corresponds to a "clock" which may start ticking before, at, or after the beginning of the hazard spell. The start of the clock is controlled by the origin variable. Positive origins indicate that the clock started ticking before the beginning of the spell; zero corresponds to the beginning of the spell, and positive values indicate that the clock started ticking

sometime during the spell. If the origin variable is negative and larger in absolute value than the end of the spell, the duration spline essentially does not enter the model. Different spline patterns are individually identified because their respective clocks start ticking at different points in time.

Even though the origin of the spline may be sometime during the spell, its effect by default extends backward as well as forward. To ensure that a spline "kicks in" during the spell, specify option "`effect=right`" in the definition of the spline (page 293).

Regressor Splines

`regressor spline(variable=`*`varname`*`, reference=`*`splinename`*`)`

Where "*varname*" may be an expression involving possibly multiple variable names. It may also be a constant. For logical reasons, the variable to be transformed must be at at least the same level as the outcome variable, i.e., at the same level or more aggregated.

Regressor splines have the same effect as splines that are defined as part of a variable list in regressor sets. In hazard models, the difference with duration splines is that in regressor splines, the variable of which the spline transformation is taken is a constant, whereas in duration splines, the variable continuously increases with the duration of the spell. The variable to which the spline transformation is applied may be time-varying, resulting in a time-varying set of variables, but the changes over time are discreet. By contrast, duration splines result in smoothly changing patterns over the duration of a hazard spell. Regressor splines result in proportional shifts of the baseline hazard; duration splines are part of the baseline hazard. There is rarely a good reason to use regressor splines. For additional details see Section 13.2.3.

Variables

varname

New in Version 2 is the ability to enter variables on the right-hand-side of equations, without coefficient. For example, the following model:

$$y_i = \beta' x_i + z_i + u_i$$

may be expressed as the sum of a regressor set ($\beta' x_i$), a variable (z_i), and a residual (u_i). The key difference between a variable that enters an equation directly and one that enters by inclusion in a regressor set is that the latter is multiplied by a regression parameter that can be estimated.

Inclusion of a variable without coefficient is not often very meaningful. It may, for example, fulfill the role of an exposure variable in a Poisson or negative binomial model (e.g., footnote 15 on page 63). More commonly, a variable may be interacted with other building blocks, notably residuals in random coefficients models (Section 5.6).

13.3.4. Indirect Referencing of Building Blocks

This section documents how building blocks may be "indirectly referenced" in model statements. An indirectly referenced building block enter models conditional on the value of a data variable. The syntax always involves a reference variable (`refvar`) specification:

```
regressor set (refvar=varname)
parameter (refvar=varname)
residual(draw=varname, refvar=varname)
integrated residual(draw=varname, refvar=varname)
duration spline(origin=varname, refvar=varname)
regressor spline(variable=varname, refvar=varname)
```

where "*varname*" may also be an expression potentially involving multiple variables. For logical reasons, the variable(s) must be at at least the level of the outcome, i.e., at the same level or more aggregated.

Indirect referencing enable the selection of building blocks on the basis of characteristics of the outcome of interest. For example, in a probit model suppose you wish to specify different regressor sets for each of four educational categories. The categories are distinguished by variable "`educ`" which may be 1, 2, 3, and 4. One approach, using direct referencing, is:

```
define regset HSDropout;   ref=1; var = ...;
define regset HSGraduate;  ref=2; var = ...;
define regset SomeCollege; ref=3; var = ...;
define regset CollegeGrad; ref=4; var = ...;

probit model; keep if educ==1; outcome = ...;
   model = regset HSDropout + ...;
probit model; keep if educ==2; outcome = ...;
   model = regset HSGraduate + ...;
probit model; keep if educ==3; outcome = ...;
   model = regset SomeCollege + ...;
probit model; keep if educ==4; outcome = ...;
   model = regset CollegeGrad + ...;
```

Alternatively, you may use indirect referencing and select the appropriate regressor set on the fly:

```
probit model;
   outcome = ...;
   model = regset (refvar=educ) + ...;
```

Instead of four model statements, there is only one. Note that the names of the regressor sets are no longer used. (Indeed, they may be omitted from their definitions, though it may be good to leave them there for your own documentation purposes.) Instead, the regressor sets' reference

numbers are used. For each probit outcome, aML evaluates reference variable `educ` and attempts to find its value among the reference numbers of all define regressor sets. The regressor set with matching reference number is used in the equation.

The example applies equally to all other types of models and all other types of building blocks that may be defined with reference numbers.

Indirect referencing requires that candidate building blocks have been defined with reference numbers. Almost all building blocks may be named (with a 12-character name) and/or identified by reference numbers ("`ref=n...n`"). Reference numbers must be strictly positive integers. You may assign as many reference numbers to any one building block as you wish. (There is an internal maximum, but it may be increased—see "`option maximum number of reference numbers`" on page 279.)

What if the reference variable evaluates to a number which does not appear among reference numbers? There are two possibilities. First, the reference variable is nonzero: aML then aborts with an instructive error message. Second, the reference variable is zero: aML then omits the building block from the equation. Indirect referencing thus allows for conditional inclusion of building blocks.

If a reference variable evaluates to zero, its building block is excluded from the model specification. If the building block is interacted with other building blocks (Section 13.3.5), all interacted blocks are also excluded.

This feature is particularly useful when the number of building blocks varies across equations. For example, you wish to include the effect of the death of an older child on the hazard of conceiving a next child to determine whether there is evidence of child replacement. You include a spline duration dependence which starts operating (kicks in) at the moment that a child in the family dies. However, fortunately, not all families experience child mortality in every conception spell (birth interval). The replacement effect thus enters conditionally on the death of a child. You could even include multiple conditional replacement effects, corresponding to multiple deaths, to account for potentially multiple infant's or child's mortality in one family.

Indirect referencing may involve expressions rather than just reference variables. The above example specified separate regressor sets by education. Suppose that in addition to `educ`, one wants interactions by sex as well. The data contain a variable `sex` which is `1` for males and `2` for females, say. The number of possibilities is now eight, and we select reference numbers that are unique combinations of `educ` and `sex`:

```
define regressor set; ref = 11; var = ...;
define regressor set; ref = 12; var = ...;
define regressor set; ref = 13; var = ...;
define regressor set; ref = 14; var = ...;
```

```
define regressor set; ref = 21; var = ...;
define regressor set; ref = 22; var = ...;
define regressor set; ref = 23; var = ...;
define regressor set; ref = 24; var = ...;
```

Accordingly, the model statement may include:

```
model = regset (refvar=10*sex+educ) + ...;
```

13.3.5. Interactions of Building Blocks

You may interact building blocks and estimate an extraordinarily rich class of models. For example, interactions of multiple regressor sets enable nonlinear models; interactions of regressor sets and residuals enable modeling heteroskedasticity; interactions of variables and residuals enable random coefficients models; interactions of parameters and inverse matrix elements enable estimation of structural parameters in systems of simultaneous equations. See Section 5 for a detailed discussion of these types of advanced models.

Parameters, regressor sets, regressor splines, and variables may be freely interacted with each other, and with duration splines, residuals, and integrated residuals. Duration splines and (integrated) residuals may not be interacted with each other.

Vectors, which never directly enter model specifications, may not be interacted with any other building block. Examples of permissible interactions include:

```
par Lambda * par Gamma
par Lambda * regset BetaX
par Lambda * varname
regset BetaX1 * regset BetaX2
par Lambda * res(draw=year, ref=eps)
regset BetaX * intres(draw=year, ref=eps)
varname * res(draw=year, ref=eps)
par Lambda * durspline(origin=age, ref=AgeEffect)
regset BetaX * durspline(origin=age, ref=AgeEffect)
par Lambda * regset BetaX * res(draw=year, ref=eps)
```

Et cetera. There is no hard limit to the number of interaction terms. Regressor splines and (inverse) matrix elements may be used in the same way as parameters and regressor sets. The examples all use directly referenced building blocks, but the same rules apply to indirectly referenced building blocks.

If the reference variable of an indirectly referenced building block evaluates to zero, that building block does not enter the equation. All other building blocks with which it is interacted then also drop out of the equation.

The best way to interpret these interactions is to view a regressor set as $\beta'X$; this entity is multiplied by a parameter, say, λ, so that $\lambda(\beta'X)$ enters the model: all regressor coefficients β are scaled. Similarly, the interaction of two regressor sets yields $(\beta_1'X_1)(\beta_2'X_2)$.

The interaction of a parameter with a residual or integrated residual ε simply means that not ε, but $\lambda\varepsilon$ enters the model specification. Similarly, the interaction of a regressor set $\beta'X$ and a residual ε yields $(\beta'X)\varepsilon$. This allows for heteroskedasticity, i.e., it enables the user to let the standard deviation of the variance component be a function of regressors. Under most circumstances, some normalization must be imposed, for example, by fixing the standard deviation of the residual to one. For example, suppose you want to test whether the standard deviation of a variance component has remained constant over time. If you suspect that the standard deviation has increased or decreased linearly, you can define a regressor set with the number "1" and variable `time`. If a more complicated time pattern is suspected, the regressor set should contain the number "1" and a spline transformation of `time`, e.g., `spline(time, ...)`. Or perhaps you suspect a quadratic development: "`1 time time*time`". Note that the value of the regressor set $\beta'X$ affects the standard deviation linearly; the variance is related to the square of $\beta'X$. Also note that the standard deviation is equal to the absolute value of the regressor set; if $\beta'X$ is negative, the signs of some correlation coefficients are implicitly reversed.

The interaction of a variable and a residual yields factors like $z\varepsilon$. This type of interaction is particularly useful in random coefficients models.

A spline duration dependence may be expressed as $\gamma T(t)$, where γ is a vector of slopes and $T(t)$ is a vector of linear spline transformations of the duration at any point *t* during the period of risk. (We are ignoring the spline intercept term, if any.) Interaction with a parameter or regressor set then yields $\lambda(\gamma T(t))$ or $(\beta'X)(\gamma T(t))$, i.e., all slopes (and the intercept, if any) are multiplied by a parameter or regressor set. This tilts the duration dependence: the slope on each segment between nodes is multiplied by λ or $\beta'X$. The intercept is also multiplied by λ or $\beta'X$. There is an important restriction to the interaction of a regressor set and a duration spline: the regressor set may not contain variables defined below the outcome level (i.e., may not be time-varying). Only variables defined at at least the level of the outcome are permissible, so that $\beta'X$ is constant for the entire period of risk. Note that the interaction of a regressor set with a regressor spline is different from the interaction of a regressor set with a duration spline: in the latter case, the entire term is part of the baseline hazard function, whereas the former case only shifts the baseline hazard.

In hazard models, interactions of regressor sets with duration splines or with integrated residuals are subject to the restriction that the regressor set not contain variables defined below the outcome level. Only variables defined at or above the level of the outcome are permissible, so that $\beta' X$ is constant over the entire spell. Interactions of regressor splines and regressor sets are not subject to this restriction.

The order in which interaction terms appear is subject to the rule that duration splines and (integrated) residuals must be listed after parameters, regressor sets, and regressor splines.

In other words,

```
durspline ... * par ...
intres ... * regset ...
```

will result in an error message, but

```
par ... * regset ... * durspline ...
par ... * regset ... * res ...
```

are permissible and equivalent to:

```
regset ... * par ... * durspline ...
regset ... * par ... * res ...
```

13.3.6. Correlated and Independent Residual Draws

Residuals may be included in equations through the following specification:

```
res(draw=drawvar, ref=resname)
```

where *drawvar* is the name of a data variable or expression, and *resname* is the name of the residual. Residuals may also be indirectly specified:

```
resname(draw=drawvar, refvar=refvar)
```

where *refvar* is the name of a data variable or expression. In multilevel and multiprocess models, there may be many equations and potentially very many residuals. It is critical to correctly specify which residuals are correlated with other residuals, and which residuals are independent. The basic rule is:

Residuals are correlated if and only if (1) they were defined as part of the same distribution, and (2) they have the same draw numbers.

The draw numbers must be explicitly specified by a draw variable or expression. In particular, residuals with "`draw=_iid`" are always independent from every other residual. We could add a third condition, namely that the residuals pertain to outcomes in the same observation. By the definition of an observation, nothing is correlated across observations.

It may be helpful to think of a draw as a realization of the residual. Each time an outcome is observed, one or more residuals have been realized and combined with regressors to form the outcome. Different realizations are different "draws."

If a residual is specific to all outcomes of an observation, we tend to specify "`draw=1`". This does *not* mean that the residual takes on value 1; it means that the residual takes on a draw that is uniquely identified as draw number 1. "`Draw=1`" ensures that the residual has the same draw number in all equations in which it appears. Key is here that draw is always the same, i.e., there is only one realization of this residual. We could have specified "`draw=536`" or "`draw=_id`" or any other expression that evaluates to the same number for all equations of an observation. This illustrates another rule:

The actual draw number is irrelevant. The only thing that matters is whether the draw number is the same as draw numbers in other equations.

If a residual is specific to a subset of equations within an observation, we tend to use a draw variable. The variable must have the same number for all equations in the subset of interest, and other draw numbers outside the subset.

Example 1

To focus thoughts, consider a simple continuous outcome model with three levels:

$$Y_{ijt} = \beta' X_{ijt} + \varepsilon_i + \eta_{ij} + u_{ijt},$$

where i denotes, say, an individual (level 1), j a job (level 2), and t a year (level 3). The unit of observation is an individual. We observe annual data over a career with potentially many jobs per individual and potentially multiple years on each job. The outcome of interest is annual earnings (Y_{ijt}) of person i on job j in year t. X_{ijt} represents explanatory covariates; ε_i represents

unobserved heterogeneity specific to the individual (ambition, intelligence, …); η_{ij} represents unobserved heterogeneity specific to the job (quality of worker-job match, non-pecuniary job aspects, …); and u_{ijt} captures residual variation.

To facilitate the discussion, suppose the first observation is an individual for whom annual earnings are observed over three jobs. On these three jobs, we observe two, three, and one annual earnings figures, respectively. There are thus six outcomes to be explained, corresponding to the following six equations:

$$\begin{cases} Y_{111} = \beta' X_{111} + \varepsilon_1 + \eta_{11} + u_{111} \\ Y_{112} = \beta' X_{112} + \varepsilon_1 + \eta_{11} + u_{112} \\ Y_{121} = \beta' X_{121} + \varepsilon_1 + \eta_{12} + u_{121} \\ Y_{122} = \beta' X_{122} + \varepsilon_1 + \eta_{12} + u_{122} \\ Y_{123} = \beta' X_{123} + \varepsilon_1 + \eta_{12} + u_{123} \\ Y_{131} = \beta' X_{131} + \varepsilon_1 + \eta_{13} + u_{131} \end{cases}$$

or:

$$\begin{pmatrix} Y_{111} \\ Y_{112} \\ Y_{121} \\ Y_{122} \\ Y_{123} \\ Y_{131} \end{pmatrix} = \beta' \begin{pmatrix} X_{111} \\ X_{112} \\ X_{121} \\ X_{122} \\ X_{123} \\ X_{131} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \varepsilon_1 + \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_{11} \\ \eta_{12} \\ \eta_{13} \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_{111} \\ u_{112} \\ u_{121} \\ u_{122} \\ u_{123} \\ u_{131} \end{pmatrix}.$$

Note that, for this first observation, there is only one ε, three distinct η's, and six distinct u's. In draw-terms, there is only one draw of ε, three of η, and six of u. While we know how many distinct values (draws) there are, we do not know their values. The task is to tell aML that (1) all equations of this first observation get the same ε; (2) that the first two equations get one draw of η, the next three another, and the last one a third value of η; and (3) that all equations get a different u.

More generally, we need to specify the residual structure $\varepsilon_i + \eta_{ij} + u_{ijt}$ in the aML control file. This requires that the data contain a variable that is unique to each job, say, `jobid` ("job ID"). The control file may then contain:

```
define regressor set BetaX; var = …;
define normal distribution; dim=1; name=eps;
define normal distribution; dim=1; name=eta;
define normal distribution; dim=1; name=u;

continuous model;
   outcome = y;
```

```
model = regset BetaX +
   res(draw=1, ref=eps) +
   res(draw=jobid, ref=eta) +
   res(draw=_iid, ref=u);
```

Exactly what do the draws in these residual specifications mean? The first, "`res(draw=1, ref=eps)`" states that residual `eps` enters the model with the same draw (value) in every equation. That value is *not* equal to 1. The only relevant thing about "`draw=1`" is that the draw is the same for every equation (within an observation; across observations, all residuals are by definition uncorrelated). We could have written "`draw=536`" or "`draw=3*55-10`" or anything else that evaluates to a constant number for all equations of a specific observation. Some people write "`draw=_id`", where `_id` is the built-in variable denoting observation ID. This, too, is constant for all equations within observation. Again, the draw is not 1 or 536 or whatever; the draw specification serves merely to identify which equations get the same ε.

The second residual specification, "`res(draw=jobid, ref=eta)`" tells aML that all equations with the same job ID get the same draw. Suppose the jobs are numbered 1, 2, and 3. The first two equations have `jobid=1`, the next three have `jobid=2`, and the last has `jobid=3`. aML counts the number of distinct `jobid` values (three) and acts as-if there are three independent distributions of η (namely corresponding to η_{l1}, η_{l2}, and η_{l3}, above). It does not matter what values `jobid` takes on; all that matters is that the first two equations have the same `jobid`, the next three the same `jobid` (and different from the first value), and the last yet another value. These values could be 10662, 7884, and 8770. i.e., they need not be ordered.

The third residual specification, "`res(draw=_iid, ref=u)`" indicates that all values of *u* are independent from each other. If all six earnings observations pertain to different calendar years, we could write "draw=year". Specifying "draw=_iid" is just shorthand to ensure that all draws are independent.

The above equation may be written more compactly as $Y_1 = \beta' X_1 + \Lambda_\varepsilon \varepsilon_1 + \Lambda_\eta \eta_1^* + \Lambda_u u_1^*$, where Λ's are load matrices and asterisks reflect multiple independent draws of residuals. The overall covariance matrix is:

$$\Lambda_\varepsilon \sigma_\varepsilon^2 \Lambda_e' + \Lambda_\eta \begin{pmatrix} \sigma_\eta^2 & 0 & 0 \\ 0 & \sigma_\eta^2 & 0 \\ 0 & 0 & \sigma_\eta^2 \end{pmatrix} \Lambda_\eta' + \Lambda_u \begin{pmatrix} \sigma_u^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_u^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_u^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_u^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_u^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_u^2 \end{pmatrix} \Lambda_u',$$

which further illustrates that there is one draw of ε, three independent draws of η, and six independent draws of *u*. Again, we do not know (and do not need to know) what the various realizations (values) are, just that there are one, three, and six, respectively.

Draws in Multiprocess Settings

So far, the discussion covered a multilevel setting with a single process (annual earnings). Now consider various extensions to multiprocess settings.

Suppose educational attainment is among the explanatory covariates. Intelligent individuals ($\varepsilon_i > 0$) are likely to obtain more education than their less-intelligent counterparts, so that educational attainment may be positively correlated with an unobservable in the earnings equation. This would cause the estimated effect of education on earnings to be positively biased. Suppose we have a test score for every individual, T_i. We jointly estimate:

$$\begin{cases} Y_{ijt} = \beta' X_{ijt} + \varepsilon_i + \eta_{ij} + u_{ijt} \\ T_i = \alpha' X_i + \delta_i \end{cases}$$

where ε_i and δ_i both capture aspects of intelligence and are may therefore be correlated. The control file would contain something like:

```
define normal distribution; dim=2;
   name=eps;
   name=delta;

continuous model;   /* Model for annual earnings */
   outcome = y;
   model = ... + res(draw=1, ref=eps) + ...;

continuous model;   /* Model for test score */
   outcome = t;
   model = ... + res(draw=1, ref=delta) + ...;
```

Key is that `eps` and `delta` get the same draw. Equivalently, we could write "`res(draw=536, ref=eps)`" and "`res(draw=536, ref=delta)`", but *not* "`res(draw=1, ref=eps)`" and "`res(draw=536, ref=delta)`". Recall the basic rule:

Residuals are correlated if and only if (1) they were defined as part of the same distribution, and (2) they have the same draw numbers.

In the example, there is only one test score, i.e., only one equation per observation. Alternatively, there could be a battery of scores, each explained by some covariates, a common residual δ_i, and independent transitory residuals. The specifications of ε_i and δ_i would remain as above.

Now suppose a job characteristic that enters as explanatory covariate is correlated with unobservable job characteristics. (It is difficult to think of a good example; perhaps the explanatory covariate is occupation, and perhaps beach lifeguards and park rangers earn less but have pleasant unobservable job characteristics.) For each job, we jointly estimate a model explaining that job characteristic with the model of annual earnings (and possible also joint with the test score model):

$$\begin{cases} Y_{ijt} = \beta' X_{ijt} + \varepsilon_i + \eta_{ij} + u_{ijt}; \\ C_{ij} = \gamma' X_{ij} + \kappa_{ij}, \end{cases}$$

where η_{ij} and κ_{ij} may be correlated. The control file would contain something like:

```
define normal distribution; dim=2;
   name=eta;
   name=kappa;

continuous model;   /* Model for annual earnings */
   outcome = y;
   model = ... + res(draw=jobid, ref=eta) + ...;

continuous model;   /* Model for job characteristic */
   outcome = c;
   model = ... + res(draw=jobid, ref=kappa) + ...;
```

Residuals `eta` and `kappa` have the same draw variable, `jobid`, ensuring proper correlations across equations. Data for the earnings and job characteristics models may be in different data structures, as long as both data structures contain a consistently coded `jobid` variable.

Finally, there may be correlation at the transitory (annual) level, i.e., with u_{ijt}. For example, we only observe annual earnings if the individual is working, so we jointly estimate a annual labor force participation. This amounts to a multilevel extension of the Heckman model:

$$\begin{cases} Y_{ijt} = \beta' X_{ijt} + \varepsilon_i + \eta_{ij} + u_{ijt}; \\ P^*_{it} = \pi' X_{it} + v_{it}, \end{cases}$$

where P^*_{ijt} is the probit-propensity that individual *i* works in year *t* and u_{ijt} may be correlated with v_{it}. (This example for illustration only; conditional on earnings, the probit residuals may be correlated, which may cause troubles if there are more than three years per individual.) The control file would contain something like:

```
define normal distribution; dim=2;
   name=u;
   name=v;
```

```
continuous model;  keep if (p>0);  /* Model for annual earnings
*/
   outcome = y;
   model = ... + res(draw=year, ref=u);

probit model;   /* Model for labor force participation */
   outcome = p;
   model = ... + res(draw=year, ref=v);
```

Careful! Above, we wrote "`res(draw=_iid, ref=u)`", indicating that all *u*'s were independent. If we were to do so here as well, the *u*'s would also be independent of the *v*'s, regardless of the draw specification of *v*'s. We can only obtain correlation by explicitly tying the residuals together with common draws. (If there were earnings on multiple jobs for any one year, the "`draw=year`" specification would not suffice.)

Example 2

Consider another example. In a study of test scores in a sample of schools (level 1) with multiple students per school (level 2), multiple years in school per student (level 3), and multiple tests per school year (level 4), students may be identified by a student ID, `student`. A residual that is specific to the school may have "`draw=1`"; residuals that are student-specific should have "`draw=student`". Student IDs are unique within schools, so outcomes from different students will have different draws. It does not matter that student IDs are not unique across schools, because different schools belong to different observations, and everything across observations is by definition uncorrelated. It also does not matter that some students may have student=1. It will not be related to the draw=1 of school-specific residuals, because the school and student residuals are part of different distributions.

Continuing the school test example, we also want to allow for student-year effects. These effects should be the same for all tests in a particular school year by a particular student, and uncorrelated across years and across students. Suppose school years are identified by variable `year`. If we were to specify "`draw=year`", then we would get the same draw for all students in a particular year. Instead, we want unique draws for all students and all years. This may be achieved by, for example, "`draw=10*student+year`". The expression is unique for every student and every year, as long as $0 \leq \texttt{year} \leq 9$. More generally, we could specify "`draw=(maxyear+1)*student+year)`", where maxyear is a constant that is equal to the maximum value year ever takes.

Naturally, we will also want test-specific effects, i.e., transitory effects, or "truly residual" effects. We could specify "`draw=100*student+10*year+testnum`", where testnum is the test number, and $0 \leq \texttt{testnum} \leq 9$. However, since this is the lowest level, and there are no other equations that are correlated at this level with the test equations, we may more concisely write "`draw=_iid`". If there were other equations in the model, such as for the student's mental state at

the time of the test, we may want to allow for correlation with that other equation. In that case, we could not use "`draw=_iid`", because it ensures independence from everything else.

A common model specification error is to specify a residual with too many draws. Consider an analysis of conception intervals with data that contain zero or more conceptions (level 2) per woman (level 1). We are interested in woman-level unobserved heterogeneity and include a woman-specific residual (heterogeneity component). Suppose conceptions are numbered by variable `parity`. Woman-level heterogeneity should be specified with "`draw=1`", not by "`draw=parity`". The latter would draw a new realization of the heterogeneity component for every conception spell. Except in very specific cases, this implies that the heterogeneity component cannot be identified. The aML search path will bring its standard deviation closer and closer to zero.

If the standard deviation of a residual is estimated to be very small, verify whether you correctly specified its draw variable. Chances are that the model drew more independent realizations than you intended.

If a correlation coefficient is estimated to be very small, verify whether you correctly specified the draw variables of the corresponding two residuals. Chances are that the model did not match the residual pairs as you intended. Recall that residuals are only correlated if they are part of the same distribution and have the same draw number. A "`draw=_iid`" specification is guaranteed to fail to contribute to identification of correlations.

13.4. Continuous Models

This section documents continuous outcome models. Only aspects that are specific to this type of model are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
continuous model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = varname;
   model = <building blocks>;
```

where the outcome may also be an expression. For example:

```
outcome = income;
outcome = log(income);
outcome = log(income+sqrt(income^2+1));
outcome = weight/(height^2);
```

and the right-hand-side of the equation may include parameters (including elements of matrices and inverse matrices), regressor sets and regressor splines, integrated residuals, and non-integrated residuals.

Unique Options and Features

Continuous models must have at least one (non-integrated) residual from a normal distribution with independent draws for every continuous outcome. They may have multiple residuals (variance components) from normal, finite mixture, ARMA(*p*,*q*), and CAR(1) distributions. Residuals from normal distributions may be integrated out ("`intres`") or enter directly ("`res`"). Residuals from finite mixtures must be integrated out. Residuals from ARMA(*p*,*q*) and CAR(1) distributions must enter directly.

For technical reasons, there may only be one independent draw for integrated residuals, if any. This is rarely an issue, because it is almost always more efficient to include residuals as non-integrated, and let aML compute the closed form solution to the likelihood.

Naturally, you may specify residuals in continuous models which are correlated with residuals or integrated residuals in other types of models. In such cases, the distribution of the residual in the non-continuous models is conditional on the vector of continuous outcomes. See below for several examples.

Continuous models (and only continuous models) may include ARMA(p,q) and CAR(1) residuals; see Section 13.2.7 and 13.2.8, respectively.

Discussion

Continuous models are models in which the outcome is a continuous variable. The program's implementation allows for many features that make it especially well suited for panel data analysis. These features express themselves in several levels of aggregation at which variance components (residuals) may apply. The result is a large degree of flexibility, but you need to be very well aware of what you are specifying.

Continuous models may be multilevel. For example, suppose you want to analyze wage incomes in a household. The unit of observation (level 1) is the household. There may be multiple persons (level 2 branches), namely for husband's wages, wife's wages, possibly children's wages, et cetera. Each person may have zero or more jobs (level 3 branches). These jobs may overlap (multiple jobs simultaneously); this can be handled correctly. Within each job, there may be multiple wage observations (level 4 branches). For example, someone held a job for several years, and the annual panel survey provides information on wages at several points in time within this job. The outcome variable, a wage (or log-wage) is thus a level 4 variable. Occupation is a level 3 variable; sex is a level 2 variable; household structure may be an observation level variable.

Consider a few examples, in increasing complexity. First a very simple continuous model:

$$y = \beta' x + u \,,$$

where we suppressed the observation subscript. There is only one outcome per observation. The control file would contain:

```
define regset BetaX; var=...;
define normal distribution; dim=1; name=u;

continuous model;
   outcome=y;
   model = regset BetaX +
           res(draw=1, ref=u);
```

The draw variable of residual "u" is always one, implying that u's draw is always the same within an observation. Since there is only one outcome per observation, and one u, this is fine. For the interested reader, the likelihood is the normal probability density:

$$L = \frac{1}{\sigma_u \sqrt{2\pi}} \exp\left\{ -\frac{\left(y - \beta' x\right)^2}{2\sigma_u^2} \right\},$$

where σ_u^2 is the covariance of residual u.

Second, suppose you wish to estimate the continuous model simultaneously with a probit selection model:

$$\begin{aligned} y &= \beta' x + u \\ p^* &= \alpha' x + v, \quad p = 0 \text{ if } p^* < 0 \\ &\qquad\quad p = 1 \text{ if } p^* > 0 \end{aligned}$$

where p^* is the probit propensity. Its binary outcome (variable "p") is among the covariates in the first equation. If the correlation between *u* and *v* is nonzero, estimation of the first equation without regard to the second equation will be biased. The control file for this simple selection model is:

```
define regset BetaX; var=...;
define regset AlphaX; var = ...;
define normal distribution; dim=2; name=u; name=v;

continuous model;
   outcome=y;
   model = regset BetaX +
           res(draw=1, ref=u);

probit model;
   outcome=p;
   model = regset AlphaX +
           res(draw=1, ref=v);
```

Note that "u" and "v" are correlated because (1) they are part of the same distribution and (2) their draw variables are identical. As a user, you need to be able to write down the model's mathematical equations, but not necessarily the likelihood function. For those who are interested, the joint likelihood of the continuous and probit outcomes may be separated into a continuous and a probit part, where probit residual *v* becomes conditional on the realized value of *u*, and thus on the continuous outcome:

$$L = L_1 L_2, \text{ where } L_1 = \frac{1}{\sigma_u \sqrt{2\pi}} \exp\left\{ -\frac{(y - \beta' x)^2}{2\sigma_u^2} \right\} \text{ and } L_2 = \begin{cases} 1 - \Phi\left(\dfrac{\alpha' x + \mu_{v|u}}{\sigma_{v|u}} \right) & \text{if } p = 0 \\ \Phi\left(\dfrac{\alpha' x + \mu_{v|u}}{\sigma_{v|u}} \right) & \text{if } p = 1 \end{cases}$$

Note that the distribution of *v* conditional on *u* is:

$$\begin{pmatrix} u \\ v \end{pmatrix} \sim N\begin{pmatrix} \sigma_u^2 & \\ \sigma_{vu} & \sigma_v^2 \end{pmatrix} \Rightarrow v|u \sim N\left(\frac{\sigma_{uv}}{\sigma_u^2}(y - \beta' x), \sigma_v^2 - \frac{\sigma_{uv}^2}{\sigma_u^2} \right).$$

Third, consider a simple continuous model with repeated outcomes (panel data model) and a variance component:

$$y_i = \beta' x_i + \varepsilon + u_i ,$$

where subscript i (i=1,...,k) indicates a replication number within an observation (the observation subscript has been suppressed) and ε is common (same draw) across all replications. Suppose the panel consists of annual waves, and wave i may be identified by data variable "`year`". If ε is not integrated out, the model may be specified as:

```
define regset BetaX; var=...;
define normal distribution; dim=1; name=eps;
define normal distribution; dim=1; name=u;

continuous model;
   outcome=y;
   model = regset BetaX +
           res(draw=1, ref=eps) +
           res(draw=year, ref=u);
```

Note draw variable "year" in the specification of residual "u". Since it takes on distinct values for each replication of the outcome, all residuals will be independent. (We could have specified "`draw=_iid`" with the same result.). The likelihood function is the multivariate normal density:

$$L = (2\pi)^{-k/2} \left|\Sigma_{\varepsilon+u,\varepsilon+u}\right|^{-1/2} \exp\left\{-\tfrac{1}{2}(Y - X\beta)' \Sigma^{-1}_{\varepsilon+u,\varepsilon+u}(Y - X\beta)\right\},$$

where $\Sigma_{\varepsilon+u,\varepsilon+u}$ is the covariance matrix of the k-vector of residuals and $\left|\Sigma_{\varepsilon+u,\varepsilon+u}\right|$ is its determinant. Note that

$$\Sigma_{\varepsilon+u,\varepsilon+u} = \begin{pmatrix} \sigma_\varepsilon^2 + \sigma_u^2 & & & \\ \sigma_u^2 & \sigma_\varepsilon^2 + \sigma_u^2 & & \\ \vdots & \vdots & \ddots & \\ \sigma_u^2 & \sigma_u^2 & \cdots & \sigma_\varepsilon^2 + \sigma_u^2 \end{pmatrix}.$$

Equivalently, you may write the likelihood as the integral over ε of the conditional likelihood (conditional on ε):

$$L = \int_\varepsilon \tfrac{1}{\sigma_e}\phi\left(\tfrac{\varepsilon}{\sigma_\varepsilon}\right)(2\pi)^{-k/2}\left|\Sigma_{uu}\right|^{-1/2} \exp\left\{-\tfrac{1}{2}(Y - X\beta - \varepsilon)' \Sigma^{-1}_{uu}(Y - X\beta - \varepsilon)\right\} d\varepsilon ,$$

where Σ_{uu} is the covariance matrix of the k-vector of residuals u. If you specify ε as an integrated residual in the continuous model statement, aML maximizes a numerical approximation to that likelihood:

$$L = \sum_{i=1}^{n} w_i (2\pi)^{-k/2} \left|\Sigma_{uu}\right|^{-1/2} \exp\left\{-\tfrac{1}{2}\left(Y - X\beta - e_i\right)' \Sigma_{uu}^{-1}\left(Y - X\beta - e_i\right)\right\},$$

where n is the number of integration points (user-specified in the define normal distribution statement) and w_i and e_i are Gauss-Hermite weights and support points, respectively. See Section 13.2.6, especially page 305. The higher the number of points n, the closer the approximation. Suppose you decide on six support points:

```
define regset BetaX; var=...;
define normal distribution; dim=1;
   number of integration points=6; name=eps;
define normal distribution; dim=1; name=u;

continuous model;
   outcome=y;
   model = regset BetaX +
           intres(draw=1, ref=eps) +   /* Note the intres */
           res(draw=year, ref=u);
```

With sufficiently many numerical integration points, the estimates of continuous models with directly entering higher-level residuals ("res") and integrated residuals ("intres") are identical. Since numerical integration requires more computations, we suggest that you directly enter higher-level residuals into the continuous outcome equation.

However, in systems of simultaneous equations involving continuous and probit models, you may be forced to integrate-out higher-level residuals. As a fourth example, suppose you wish to estimate the panel data continuous model simultaneously with a probit selection model:

$$y_i = \beta' x_i + \varepsilon + u_i$$
$$p_i^* = \alpha' x_i + \eta + v_i$$

where p_i^* is the probit propensity. Its binary outcome (variable "p") may be among the covariates in the first equation. You wish to allow for both "permanent" selection $\left(\text{cov}(\varepsilon,\eta) \neq 0\right)$ and "transitory" selection $\left(\text{cov}(u,v) \neq 0\right)$. Correlation in the residuals across equations implies that "p", as an explanatory covariate in the first equation, is correlated with the residuals in that equation, so that estimation of the continuous model without regard to the probits yields biased results. Suppose ε enters your continuous equation directly, and η is integrated out:

```
define regset BetaX; var=...;
define regset AlphaX; var = ...;
define normal distribution; dim=2; numintpoints=6;
   name=eps; name=eta;
```

```
define normal distribution; dim=2; name=u; name=v;

continuous model;
   outcome=y;
   model = regset BetaX +
           res(draw=1, ref=eps) +
           res(draw=year, ref=u);

probit model;
   outcome=p;
   model = regset AlphaX +
           intres(draw=1, ref=eta) +
           res(draw=year, ref=v);
```

Note that "eps" and "eta" are correlated because (1) they are defined as part of the same distribution and (2) they have the same draw variable. The same holds for "u" and "v", which through draw variable "year" are pairwise correlated. (You may not specify "draw=_iid", because all residuals would then be independent.) Similar to the single-level selection equation above, the distribution of v_i is conditional on continuous outcomes. Let's write the systems of equations in matrix form:

$$Y = X\beta + \Lambda_\varepsilon \varepsilon + u$$
$$P^* = X\alpha + \Lambda_\eta \eta + v$$

where Λ_ε and Λ_η are both *i*-vectors of ones, $\Lambda_\varepsilon = \Lambda_\eta = (1,1,\ldots,1)'$, ε and η are scalars, and *u* and *v* are *k*-vectors. Note that

$$\begin{pmatrix}\varepsilon\\ \eta\end{pmatrix} \sim N\left(0, \begin{pmatrix}\sigma_\varepsilon^2 & \\ \sigma_{\varepsilon\eta} & \sigma_\eta^2\end{pmatrix}\right) \text{ and } \begin{pmatrix}u\\ v\end{pmatrix} \sim N\left(0, \begin{pmatrix}\Sigma_{uu} & \\ \Sigma_{vu} & \Sigma_{vv}\end{pmatrix}\right) = N\left(0, \begin{pmatrix}I\sigma_u^2 & \\ I\sigma_{vu} & I\sigma_v^2\end{pmatrix}\right),$$

where *I* is the *k*-by-*k* identity matrix. Conditional on continuous outcomes vector *Y*, the distribution of *v* is:

$$v|Y \sim N\left(\Sigma_{vu}\left(\Lambda_\varepsilon \Sigma_{\varepsilon\varepsilon} \Lambda_\varepsilon' + \Sigma_{uu}\right)^{-1}\left(Y - XB\right), \Sigma_{vv} - \Sigma_{vu}\left(\Lambda_\varepsilon \Sigma_{\varepsilon\varepsilon} \Lambda_\varepsilon' + \Sigma_{uu}\right)^{-1}\Sigma_{vu}'\right).$$

Since we condition on *Y* (ε and *u*) instead on just *u*, the resulting conditional covariance matrix is not diagonal. The *k* probit modules may thus not be separated, and a *k*-variate cumulative normal probability algorithm is required. In aML, this is implemented up to trivariate, so for *k*>3, aML will issue and error message and suggest that you try to remove the source of the correlation. This is most easily accomplished by integrating-out ε in the continuous equations. This is one of the very few cases where you have no choice but to integrate-out higher-level residuals in continuous models. As stated above, integrated residuals typically require more computations than directly entering residual.

In the next example, you wish to estimate a panel data continuous model in which one of the explanatory covariates is the outcome of a correlated hazard process. For example, job tenure may be endogenous in an analysis of annual wages. Suppose your data contain information on multiple jobs per person and multiple annual wage records per job. The model may be:

$$y_{jt} = \beta' x_{jt} + \varepsilon + u_j + w_{jt}$$
$$\ln h_j(t) = \gamma T(t) + \alpha' x_j + \eta$$

where j indicates job number, t time period, $\ln h_j(t)$ is the log-hazard of separation from job j at time t, and $\gamma T(t)$ is the duration dependency (baseline log-hazard). There are no time-varying covariates in the hazard model, but the example remains virtually the same with time-varying covariates. This model has three levels in wage outcomes and two in duration on the job. If ε is integrated out, no correlation remains in the conditional likelihoods, and continuous and hazard modules may be readily separated. If ε is not integrated out, the likelihood function is more complicated. First the model specification:

```
define regset BetaX; var=...;
define spline JobDuration; nodes=...;
define regset AlphaX; var = ...;
define normal distribution; dim=2; numintpoints=6;
   name=eps; name=eta;
define normal distribution; dim=1; name=u;
define normal distribution; dim=1; name=w;

continuous model;
   outcome=y;
   model = regset BetaX +
           res(draw=1, ref=eps) +
           res(draw=jobnum, ref=u) +
           res(draw=_iid, ref=w);

hazard model;
   censor=...; duration=...;
   model = durspline(origin=0, ref=JobDuration) +
           regset AlphaX +
           intres(draw=1, ref=eta);
```

Recall that, in hazard models, all residuals must be integrated out. To derive the likelihood function, it is again useful to write the model in matrix notation:

$$Y = X\beta + \Lambda_\varepsilon \varepsilon + \Lambda_u u + w$$
$$\ln H(t) = \mathrm{T}(t)\gamma + X\alpha + \Lambda_\eta \eta$$

Suppose there are J jobs and $(N_1,...,N_J)$ annual wage records per job. Residuals ε and η are scalar, u is a J-vector, and w is an N-vector, where N is the sum over $(N_1,...,N_J)$. The load matrices on residuals are Λ_ε, an N-vector of ones, Λ_η, a J-vector of ones, and

$$\Lambda_u = \begin{pmatrix} 1_{N_1} & 0 & \dots & 0 \\ 0 & 1_{N_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1_{N_J} \end{pmatrix},$$

where 1_n is an n-vector of ones. Conditional on continuous outcomes Y, the distribution of η is:

$$\eta|Y \sim N\left(\Sigma_{\eta\varepsilon}\Lambda_\varepsilon'\Sigma_{yy}^{-1}(Y - X\beta), \Sigma_{\eta\eta} - \Sigma_{\eta\varepsilon}\Lambda_\varepsilon'\Sigma_{yy}^{-1}\Lambda_\varepsilon\Sigma_{\eta\varepsilon}'\right),$$

where $\Sigma_{yy} = \Lambda_\varepsilon\Sigma_{\varepsilon\varepsilon}\Lambda_\varepsilon' + \Lambda_u\Sigma_{uu}\Lambda_u' + \Sigma_{ww}$, $\Sigma_{\eta\varepsilon} = \sigma_{\eta\varepsilon}$, and $\Sigma_{\eta\eta} = \sigma_\eta^2$. We used $\Sigma_{\eta\varepsilon}$ and $\Sigma_{\eta\eta}$ as they generalize to the case where ε and η are sub-vectors of a higher-dimensional integrated distribution. AML integrates-out η corresponding to this formula. It adds the conditional mean to Gauss-Hermite support points and pre-multiplies the result by Λ_η to obtain the integration point in each of the hazard equations.

Internally, aML closely follows the load matrix approach. However, internally, it represents the above model as:

$$Y = X\beta + \begin{pmatrix} \Lambda_\varepsilon & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \varepsilon \\ \eta \end{pmatrix} + \begin{pmatrix} \Lambda_u & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} + \Lambda_w w$$
$$\ln H(t) = \mathrm{T}(t)\gamma + X\alpha + \begin{pmatrix} 0 & 0 \\ 0 & \Lambda_\eta \end{pmatrix}\begin{pmatrix} \varepsilon \\ \eta \end{pmatrix}.$$

From this notation, it will be clear how aML allows for multiple residuals from the same distribution (e.g., both ε and η) in the same equation: the corresponding load matrix will have fewer zero elements. As a user, you simply specify additional residuals in the model statement.

Finally, consider how heteroskedasticity may be captured in a three-level continuous model:

$$Y = X\beta + \Lambda_\varepsilon\varepsilon + \Lambda_u u + \Lambda_w w$$

where all symbols are as in the previous example, and Λ_w is the N-by-N load matrix on the transitory residuals. So far, we assumed that Λ_w is the identity matrix, but transitory residuals may be multiplied by parameters and regressor sets in the same way as other residuals. Suppose the control file contains something like:

```
continuous model;
   outcome=y;
   model = regset BetaX +
           regset ThetaX * res(draw=1, ref=eps) +
           par phi * res(draw=jobnum, ref=u) +
           regset PsiX * res(draw=_iid, ref=w);
```

The load matrices, which in the examples above consisted of zeroes and ones only, now contain the values of $\theta' x_{jt}$, ϕ, and $\psi' x_{jt}$ where there used to be ones. They become part of the overall covariance matrix and are handled appropriately.

13.5. Probit Models

This section documents simple (binary) probit models. For ordered probit models see Section 13.6; for multinomial probit models Section 13.15. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
probit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = varname;
   [threshold = {parname | (refvar=varname)};]
   model = <building blocks>;
```

Unique Options and Features

A probit model is used to specify a model with a binary outcome. You may specify both non-integrated and integrated residuals, subject to a restriction discussed below.

Underlying all probit equations is an underlying propensity equation. For example, a very simple propensity equation is given by:

$$y^* = \beta' x + u$$

The value of the probit outcome variable depends on the range in which y^* falls. The only difference between probit and logit models is in the distribution of u: normal in the probit model and logistic in the logit model.[34] For the probit, the density and cumulative distribution functions are

$$f(u) = \left(2\pi\sigma_u^2\right)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}u^2\right\} = \phi(u) \text{ and } F(u) = \int_{\tau=-\infty}^{u} \left(2\pi\sigma_u^2\right)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}\tau^2\right\} d\tau = \Phi(u),$$

respectively. Typically, σ_u^2 is normalized, $\sigma_u^2 = 1$, but aML allows any value. Multivariate extensions are allowed up to trivariate, as discussed below.

[34] The two distributions are very similar, and the two models thus yield very similar results. The main difference is that the logistic distribution has a larger standard deviation (about $\pi/\sqrt{3} \approx 1.8138$) than the normal distribution, resulting in parameter estimates that are larger in absolute value. The ratio of coefficient value to standard deviation of the residual, though, is very close.

The right-hand-side specification of a probit model must always include a regressor set, parameter, or regressor spline, plus a non-integrated residual from a normal distribution with independent draws for every outcome. Optionally, you may specify integrated residuals from normal or finite mixture distributions, and additional non-integrated residuals from normal distributions. You may not include duration splines, ARMA(*p*,*q*), or CAR(1) residuals. Very often, probit models contain an independently and identically distributed standard normal residual (*iid* N(0,1)). You may define your own standard normal distribution and use its residual, you may use an implicitly defined univariate standard normal distribution, or you may even omit the standard normal residual altogether:

```
define regset BetaX; var = ...;
define normal distribution; dim=1; name=u;
probit model; ...;
   model = regset BetaX + res(draw=_iid, ref=u);
```

or:

```
   model = regset BetaX + res(draw=_iid, ref=N(0,1));
```

or:

```
   model = regset BetaX;
```

Probit equations must always include a non-integrated normal residual with independent draws for every outcome. However, if a probit equation is specified without non-integrated residual, aML inserts an *iid* N(0,1) residual. If you explicitly specify any non-integrated residual, aML does not insert any other residual.

aML requires that residuals which generate correlation across probit equations be integrated out, except when the number of probit modules is three or fewer. To better understand this rule, consider a system of *k* probit equations with a variance component, $y_i^* = \beta' x_i + \varepsilon + u_i$, i=1,…,k. If ε enters directly, the covariance matrix is:

$$\Omega = 1_k 1_k' \sigma_\varepsilon^2 + I_k \sigma_u^2 = \begin{pmatrix} \sigma_\varepsilon^2 + \sigma_u^2 & \sigma_\varepsilon^2 & \dots & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 + \sigma_u^2 & \dots & \sigma_\varepsilon^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \dots & \sigma_\varepsilon^2 + \sigma_u^2 \end{pmatrix}$$

The joint likelihood of the *k* probit modules involves a *k*-variate cumulative normal probability with this covariance matrix (where covariance (*i*,*j*) becomes negative if either probit outcome *i* or *j*, but not both, is zero.) This has been implemented in aML for $k \le 3$. For higher dimensions, it will issue an error message and suggest that you remove the correlation by integrating-out its source. In this case, that source is ε. Without ε, the covariance matrix is $I_k \sigma_u^2$, i.e., diagonal.

If the covariance matrix is diagonal, the modules may be separated into *k* univariate probits. Of course, with sufficiently many integration points, the specification with integrated higher-level residuals is equivalent to the specification with directly entering residuals.

Non-integrated probit residuals at higher levels than the outcome introduce correlation across probit equations. If there are only three or fewer probit outcomes in any one observation, this is handled without problem. However, at higher dimensionalities, you need to integrate-out the higher-level residuals.

In continuous models, numerical integration of residuals requires many more computations than directly entering residuals. This is much less the case in probit models, so we recommend that you always integrate-out higher-level residuals in probit models.

outcome = *varname*;

The outcome value provided by the variable *varname* must binary, taking a value of zero or one. Note that, instead of a variable name, varname may be an expression which evaluates to zero or one. For example,

```
outcome = (earnings>0);
outcome = (educ==2);
outcome = (disabled==1 or health<=2);
```

threshold = {*parname* | (refvar=*varname*)};

The "threshold" statement is optional and allows for nonzero thresholds. Consider the basic probit model:

$$y^* = \beta' x + u, \quad y = \begin{cases} 0 \text{ if } y^* < \tau \\ 1 \text{ if } y^* \geq \tau \end{cases}$$

where τ is the threshold. In most applications, you will want to fix the threshold (implicitly) at zero, and estimate an intercept. This is the default which applies when no "threshold" is specified. (You remain responsible for specifying the intercept, most commonly by including the number "1" in a regressor set.) Alternatively, you may define a parameter and estimate its value as the threshold:

```
define parameter tau;
define regset BetaX; var = ...;
probit model;
   outcome=y;
   threshold=tau;
   model=regset BetaX;
```

The threshold may also be referenced indirectly. For example, to specify separate thresholds for men (`sex=1`) and women (`sex=2`):

```
define parameter MaleTau;   ref = 1;
define parameter FemaleTau; ref = 2;
<...>
   threshold = (refvar=sex);
```

Note that the threshold is perfectly collinear with the intercept, so you should not specify both a threshold and an intercept. (The threshold and intercept are the same with opposite signs.)

The likelihood of a binary probit module is

$$L = \begin{cases} \Phi(\tau - \beta' x) & \text{if } y = 0; \\ 1 - \Phi(\tau - \beta' x) & \text{if } y = 1, \end{cases}$$

where $\Phi(u)$ is the cumulative normal probability function. A multivariate extension applies up to trivariate. If there is an integrated residual in the probit equation, say, ε, the likelihood modules becomes conditional on the Gauss-Hermite support point, say, e, and in the above, $\Phi(\tau - \beta' x - e)$ replaces $\Phi(\tau - \beta' x)$. In a system of simultaneous equations involving continuous models, probit residuals and integrated residuals may be conditional on continuous outcomes. aML will compute the conditional means and (co)variances, and compute the likelihood correspondingly.

`model` = *<building blocks>;*

The right-hand-side specification must always include a regressor set, parameter, or regressor spline. In addition, a probit model requires at least one non-integrated normal residual; aML automatically inserts an *iid* N(0,1) residual if you do not specify any non-integrated residual. Optionally, you may specify integrated residuals from normal or finite mixture distributions. You may not include duration splines or ARMA(p,q), or CAR(1) residuals.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models and heteroskedasticity; see Section 13.3.5.

13.6. Ordered Probit Models

This section documents ordered probit models. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

Thresholds to be estimated:

```
ordered probit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcomes = varname1 varname2;
   thresholds = {vectorname | (refvar=varname)};
   model = <building blocks>;
```

Thresholds are given by data variables:

```
ordered probit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   threshold vars = varname1(-Inf=x1) varname2(Inf=x2);
   model = <building blocks>;
```

Unique Options and Features

An ordered probit model is used to specify a model with a qualitative ordered outcome, i.e., to specify the probability that a random variable (or combination of random variables) is within a specified range of the real line. The range may have known or unknown (to be estimated) thresholds, which influences parameter identification. You may specify both non-integrated and integrated residuals, subject to a restriction discussed below.

Underlying all ordered probit equations is an underlying propensity equation. For example, a very simple propensity equation is given by:

$$y^* = \beta' x + u$$

The value of the ordered probit outcome variable depends on the range in which y^* falls. The density and cumulative distribution functions are

$$f(u) = \left(2\pi\sigma_u^2\right)^{-½} \exp\left\{-\tfrac{1}{2}u^2\right\} = \phi(u) \text{ and } F(u) = \int_{\tau=-\infty}^{u} \left(2\pi\sigma_u^2\right)^{-½} \exp\left\{-\tfrac{1}{2}\tau^2\right\} d\tau = \Phi(u),$$

respectively. Typically, σ_u^2 is normalized, $\sigma_u^2 = 1$, but aML allows any value. Multivariate extensions are allowed up to trivariate, as discussed below.

The right-hand-side specification of an ordered probit model must always include a regressor set, parameter, or regressor spline, plus a non-integrated residual from a normal distribution with independent draws for every outcome. Optionally, you may specify integrated residuals from normal or finite mixture distributions, and additional non-integrated residuals from normal distributions. You may not include duration splines, ARMA(*p*,*q*), or CAR(1) residuals. Very often, ordered probit models contain an independently and identically distributed standard normal residual (*iid* N(0,1)). You may define your own standard normal distribution and use its residual, you may use an implicitly defined univariate standard normal distribution, or you may even omit the standard normal residual altogether:

```
define regset BetaX; var = ...;
define normal distribution; dim=1; name=u;
ordered probit model; ...;
   model = regset BetaX + res(draw=_iid, ref=u);
```

or:

```
   model = regset BetaX + res(draw=_iid, ref=N(0,1));
```

or:

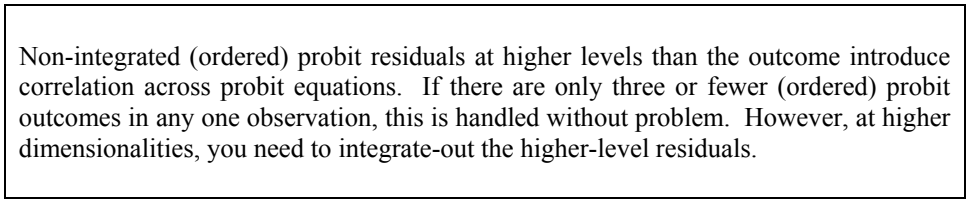
```
   model = regset BetaX;
```

Ordered probit equations must always include a non-integrated normal residual with independent draws for every outcome. However, if an ordered probit equation is specified without non-integrated residual, aML inserts an *iid* N(0,1) residual. If you explicitly specify any non-integrated residual, aML does not insert any other residual.

aML requires that residuals which generate correlation across (ordered) probit equations be integrated out, except when the number of (ordered) probit modules is three or fewer. To better understand this rule, consider a system of *k* (ordered) probit equations with a variance component, $y_i^* = \beta' x_i + \varepsilon + u_i$, *i*=1,…,*k*. If ε enters directly, the covariance matrix is:

$$\Omega = 1_k 1_k' \sigma_\varepsilon^2 + I_k \sigma_u^2 = \begin{pmatrix} \sigma_\varepsilon^2 + \sigma_u^2 & \sigma_\varepsilon^2 & \cdots & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 + \sigma_u^2 & \cdots & \sigma_\varepsilon^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \cdots & \sigma_\varepsilon^2 + \sigma_u^2 \end{pmatrix}$$

The joint likelihood of the *k* probit modules involves a *k*-variate cumulative normal probability with this covariance matrix (where covariance (*i*,*j*) becomes negative if either probit outcome *i* or *j*, but not both, is zero.) This has been implemented in aML for $k \leq 3$. For higher dimensions, it will issue an error message and suggest that you remove the correlation by integrating-out its source. In this case, that source is ε. Without ε, the covariance matrix is $I_k \sigma_u^2$, i.e., diagonal. If the covariance matrix is diagonal, the modules may be separated into *k* univariate probits. Of course, with sufficiently many integration points, the specification with integrated higher-level residuals is equivalent to the specification with directly entering residuals.



In continuous models, numerical integration of residuals requires many more computations than directly entering residuals. This is much less the case in ordered probit models, so we recommend that you always integrate-out higher-level residuals in ordered probit models.

aML supports ordered probit models with unknown thresholds that are the same for all observations and that may need to be estimated and ordered probit models with known thresholds that may vary by observation. The next two subsections discuss these model types in turn.

13.6.1. Ordered Probit Models with Unknown Thresholds

```
ordered probit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcomes = varname1 varname2;
   thresholds = {vectorname | (refvar=varname)};
   model = <building blocks>;
```

The data structure specification, keep/drop condition, and numerator/denominator statement are described in Section 13.3.1. The other statements are documented here.

outcomes = *varname1 varname2*;

The ordered probit outcome variable, with unknown thresholds, is defined to be a positive integer in a pre-specified contiguous range, say $Y = 1,2,\cdots,K$, or some range of these values. For example, individuals may be asked to rank their health status as poor, fair, good, very good, or excellent. Underlying their response is actual health status, presumably a continuous variable. Depending on the continuous value of health status and the respondent's perception of what

thresholds separate poor, fair, good, very good, and excellent health, the respondent will provide a health category. For estimation, the categories need to be converted to contiguous integers, such that the lowest category is assigned a "1" and higher categories higher values. But what if a respondent is not sure about his category, or refuses to narrow it down? For example, he only states that his health is good or very good. aML is capable of dealing with such ranges of responses. In fact, it always requires you to specify a range. Most of the time, that range consists of just one category, but any contiguous range is acceptable.

The outcome is represented by two positive integer-valued variables (or expressions), denoted above by *varname1* and *varname2*. The values of *varname1* and *varname2* represent the subscript numbers of the lower and upper thresholds, respectively, of the interval in which an underlying real values index function value lies, as explained below.

thresholds = {*vectorname* | (refvar=*varname*)};

The "threshold" statement in the binary probit model is optional; the "thresholds" statement here is mandatory. The threshold values are parameters that are organized in a vector, and may be estimated (or fixed at pre-specified values). For example:

```
define vector Cutoffs; dim=4;
ordered probit model; ...;
   thresholds = Cutoffs;
```

or using indirect referencing, for example on the basis of age:

```
define vector Young; ref=1; dim=4;
define vector Old;   ref=2; dim=4;
ordered probit model; ...;
   thresholds = (refvar = 1+(age>=70));
```

The thresholds must be strictly ordered. To prevent thresholds from crossing over during the iterative search procedure, we recommend that the vector be defined with the "increasing" option, which is done by default (page 296). The program will abort with an error message should the thresholds not be strictly ordered.

The following logic underlies these definitions. Let the real-valued continuous index value underlying the qualitative outcome be y^* and the positive integer valued outcome be denoted $y\,(=1,2,\cdots,K)$. The correspondence between the K values of the outcome variable y and K intervals on the real line is determined by K-1 thresholds and is given by

$$y^* = \beta' x + u, \quad y = \begin{cases} 1 & \text{if } \tau_0 \le y^* < \tau_1 \\ 2 & \text{if } \tau_1 \le y^* < \tau_2 \\ \vdots & \vdots \\ K & \text{if } \tau_{K-1} \le y^* < \tau_K \end{cases}$$

The end-points of the first and last intervals are defined to by $\tau_0 = -\infty$ and $\tau_K = \infty$, respectively. Verify that this leaves *K*-1 thresholds to distinguish *K* categories.

It may seem redundant to specify two variables in order to represent a single categorical outcome variable. The reason for taking this seemingly complicated route is the ability to handle censored and multi-category cases. The formulation is flexible in that outcomes corresponding to a set of contiguous integers is easily represented by specifying the subscripts of the thresholds between which the real-valued index y^* must lie. Examples include

$$y = \begin{cases} 1 & \text{if } \tau_0 \le y^* < \tau_1 & \text{varname1} = 0 & \text{varname2} = 1 \\ j & \text{if } \tau_{j-1} \le y^* < \tau_j & \text{varname1} = j-1 & \text{varname2} = j \\ 3,4 & \text{if } \tau_2 \le y^* < \tau_4 & \text{varname1} = 2 & \text{varname2} = 4 \\ 7-9 & \text{if } \tau_6 \le y^* < \tau_9 & \text{varname1} = 6 & \text{varname2} = 9 \\ m-n & \text{if } \tau_{m-1} \le y^* < \tau_n & \text{varname1} = m-1 & \text{varname2} = n \\ 1-5 & \text{if } \tau_0 \le y^* < \tau_5 & \text{varname1} = 0 & \text{varname2} = 5 \\ 8-K & \text{if } \tau_7 \le y^* < \tau_K & \text{varname1} = 7 & \text{varname2} = K \end{cases}$$

Recall that *varname1* and *varname2* may be expressions. If the data only contain single-category outcomes in, say, variable `category`, the outcomes may be conveniently specified as:

```
outcomes = category-1  category;
```

The likelihood of an outcome (integer of range of integer values) is given by the probability of the corresponding interval on the real line. Take the simple example of $y^* = \beta' x + u$, then

$$P\left(\tau_j \le y^* < \tau_k\right) = \Phi\left(\tau_k - \beta' x\right) - \Phi\left(\tau_j - \beta' x\right),$$

where $\Phi(\cdot)$ denotes the cumulative normal probability function. Ordered probit residuals may have any variance, not just the standard unit variance. Suppose *u* has standard deviation σ_u, then

$$P\left(\tau_j \le y^* < \tau_k\right) = \Phi\left(\frac{\tau_k - \beta' x}{\sigma_u}\right) - \Phi\left(\frac{\tau_j - \beta' x}{\sigma_u}\right).$$

A multivariate extension applies up to trivariate. If there is an integrated residual in the ordered probit equation, say, ε, the likelihood modules becomes conditional on the Gauss-Hermite support point, say, *e*, and in the above, $\Phi\left(\tau - \beta' x - e\right)$ replaces $\Phi\left(\tau - \beta' x\right)$. In a system of simultaneous equations involving continuous models, ordered probit residuals and integrated residuals may be conditional on continuous outcomes. aML will compute the conditional means and (co)variances, and compute the likelihood correspondingly.

model = <*building blocks*>;

The right-hand-side specification must always include a regressor set, parameter, or regressor spline. In addition, an ordered probit model requires at least one non-integrated normal residual; aML automatically inserts an *iid* N(0,1) residual if you do not specify any non-integrated residual. Optionally, you may specify integrated residuals from normal or finite mixture distributions. You may not include duration splines or ARMA(p,q), or CAR(1) residuals.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models and heteroskedasticity; see Section 13.3.5.

13.6.2. Ordered Probit Models with Known Thresholds

```
ordered probit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   threshold vars = varname1(-Inf=x1) varname2(Inf=x2);
   model = <building blocks>;
```

The data structure specification, keep/drop condition, and numerator/denominator statement are described in Section 13.3.1. The other statements are documented here.

threshold vars = *varname1*(-Inf=*x1*) *varname2*(Inf=*x2*);

Ordered probit models with known thresholds are fully analogous to their counterparts with unknown thresholds, except that the thresholds are known real numbers given in the data. There is no "`outcome`" statement; the outcome consists of an interval of which the lower and upper bound are specified in the "`threshold vars`" statement. The lower bound is *varname1*, which may be a variable name or an expression; the upper bound *varname2* may also be an expression. These values may be different or may be the same across observations or evaluations within observations.

In the case with unknown thresholds, $-\infty$ and ∞ are represented by integers 0 and K, corresponding to implicit thresholds $\tau_0 = -\infty$ and $\tau_K = \infty$. In the case with known thresholds, the threshold variable must somehow communicate to aML what $-\infty$ and ∞ are. This is implemented by letting the user specify special values for $-\infty$ and ∞. These values are communicated to aML in the "`(-Inf=x1)`" and "`(Inf=x2)`" specifications, where x1 and x2 are real numbers.

Consider an example. Suppose a household survey asks for individuals' total income. If the respondent is unable or unwilling to provide an exact dollar figure, he is asked: "Is it more or less than $40,000?" If the respondent indicates that it is less, the next question is "Is it more or less

than $15,000?" Or, if income exceeds $40,000, he is asked "Is it more or less than $100,000?" If he is unable or unwilling to reveal any information at all, the outcome of interest is missing and the record must be dropped from the analysis. If he answers the $40,000 question but not the follow-up question, we know that his income is either under $40,000 or above $40,000. If the follow-up question is answered, we know that his income is under $15,000, between $15,000 and $40,000, between $40,000 and $100,000, or above $100,000. (If he at any point indicates that his income is about equal to the question threshold, he effectively provided an exact dollar figure.) Note that there are four categories, and that the potential exists for responses that span two categories. You decide, somewhat arbitrarily, to assign -$999,999 to denote $-\infty$ and $999,999 to denote ∞. Using your (SAS, Stata, SPSS) data preparation package, you create two new variables, say, `lower` and `upper`, and recode the information as follows:

Response	lower	upper
income<15,000	-999999	15000
15,000<income<40,000	15000	40000
40,000<income<100,000	40000	100000
income>100,000	100000	999999
income<40,000	-999999	40000
income>40,000	40000	999999

Suppose your income model is $y = \beta' x + u$. With range data rather than exact income values, you may estimate an ordered probit with known thresholds:

```
define regset BetaX; var = ...;
define normal distribution; dim=1; name=u;

ordered probit model;
   threshold vars = lower(-Inf=-999999) upper(Inf=999999);
   model = regset BetaX +
           res(draw=_iid, ref=u);
```

In the income range example, we explicitly defined the probit residual and specified it in the probit model. If it were omitted, aML would automatically insert an *iid* N(0,1) residual. This would constrain its standard deviation to be one, which is very unlikely in this model. (This is also the reason why it is not a good idea to estimate the model as an ordered logit, as the logistic residual has a non-estimable standard deviation.)

Note that need to specify special values for $-\infty$ and ∞ makes it dangerous to specify the outcome (threshold variables) as expressions. For example, if all income values were positive, you might want to estimate the model in terms of logarithms, which would involve something like "`log(upper)(Inf=log(999999))`". This may or may not work, because of numerical precision issues. Internally, all data variables are stored in single precision, whereas most other numbers are in double precision. The logarithm of a single precision variable equal to 999999 may not be exactly the same as the log of a double precision number that is equal to 999999. It is

thus better to transform ordered probit threshold variables in the data preparation stage, before raw2aml.

What if the example pertained to earnings, not total income? Total income may include business losses and may thus be negative. With earnings, the lower limit is zero, and $-\infty$ is never relevant. Correspondingly, variable "lower" may be zero, but never –999999. For technical reasons, you must still assign a special number such as "(-Inf=-999999)", even though the data never contain that number. Do not specify "(-Inf=0)", as your zero is truly a zero, not a representation of $-\infty$.

The example readily extends into a switching regression application in which individuals' exact responses are modeled using the same underlying model as the range responses. Suppose that for exact responses, you set both variables lower and upper equal to the exact income value, and thus to each other. The model may be specified as:

```
define regset BetaX; var = ...;
define normal distribution; dim=1; name=u;

ordered probit model;  keep if lower<upper;
   threshold vars = lower(-Inf=-999999) upper(Inf=999999);
   model = regset BetaX +
           res(draw=_iid, ref=u);

continuous model;  keep if lower==upper;
   outcome = lower;
   model = regset BetaX +
           res(draw=_iid, ref=u);
```

aML switches between a continuous and an ordered probit model based on the values of lower and upper. The same regressor set and the same residual enter in both model specifications, so the coefficients in the two models are restricted to be equal. In other words, both exact and range responses contribute to the identification of the model. As formulated here, the underlying model is exactly the same for the two types of responses. In a real application, you would probably include an additional regressor set in the ordered probit with flags for "don't know" and "refuse" responses to capture any systematic differences that are not picked up by other observables.

13.7. Logit Models

This section documents simple (binary) logit models, also known as logistic regression models. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
logit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = varname;
   [threshold = {parname | (refvar=varname)};]
   model = <building blocks>;
```

Instead of "`logit`", you may write "`logistic`".

Unique Options and Features

A logit model is used to specify a model with a binary outcome. Residuals (unobserved heterogeneity) must be integrated out.

Underlying all logit equations is an underlying propensity equation. For example, a very simple propensity equation is given by:

$$y^* = \beta' x + u$$

The value of the logit outcome variable depends on the range in which y^* falls. The density and cumulative distribution functions of u are

$$f(u) = \frac{\exp\{-x\}}{\left(1+\exp\{-x\}\right)^2} \text{ and } F(u) = \frac{1}{1+\exp\{-u\}},$$

respectively.[35] The logistic density is also known as the sech2 or Fisk density (Johnson and Kotz, 1970, volume 2).

[35] The only difference between logit and probit models is in the distribution of the residual: normal for probits and logistic for logits. The two distributions are very similar and the two models thus yield very similar results. The main difference is that the logistic distribution has a larger standard deviation (about $\pi/\sqrt{3} \approx 1.8138$) than the normal distribution, resulting in parameter estimates that are larger in absolute value. The ratio of coefficient value to standard deviation of the residual, though, is very close.

The right-hand-side specification of a logit model must always include a regressor set, parameter, or regressor spline. Optionally, you may specify integrated residuals from normal or finite mixture distributions. You may not include duration splines or non-integrated residuals from normal, ARMA(*p*,*q*), or CAR(1) distributions. A logit model with integrated normal residual(s) is sometimes referred to as the normal probability logit model.

outcome = *varname*;

The outcome value provided by the variable *varname* must binary, taking a value of zero or one. Instead of a variable name, *varname* may be an expression which evaluates to zero or one. For example,

```
outcome = (earnings>0);
outcome = (educ==2);
outcome = (disabled==1 or health<=2);
```

threshold = {*parname* | (refvar=*varname*)};

The "threshold" statement is optional and allows for nonzero thresholds. Consider the basic logit model:

$$y^* = \beta' x + u, \quad y = \begin{cases} 0 \text{ if } y^* < \tau \\ 1 \text{ if } y^* \geq \tau \end{cases}$$

where τ is the threshold. In most applications, you will want to fix the threshold (implicitly) at zero, and estimate an intercept. This is the default which applies when no "threshold" is specified. (You remain responsible for specifying the intercept, most commonly by including the number "1" in a regressor set.) Alternatively, you may define a parameter and estimate its value as the threshold:

```
define parameter tau;
define regset BetaX; var = ...;
logit model;
   outcome=y;
   threshold=tau;
   model=regset BetaX;
```

The threshold may also be referenced indirectly. For example, to specify separate thresholds for men (sex=1) and women (sex=2):

```
define parameter MaleTau;   ref=1;
define parameter FemaleTau; ref=2;
<...>
   threshold = (refvar=sex);
```

The threshold is perfectly collinear with the intercept, so you should not specify both a threshold and an intercept. (The threshold and intercept are the same with opposite signs.)

The likelihood of a binary logit module is

$$L = \begin{cases} F(\tau - \beta' x) & \text{if } y = 0; \\ 1 - F(\tau - \beta' x) & \text{if } y = 1, \end{cases}$$

where $F(u)$ is the cumulative logistic function, $F(u) = \left(1 + \exp\{-u\}\right)^{-1}$. If there is an integrated residual, say, ε, the likelihood modules becomes conditional on the Gauss-Hermite support point, say, *e*, and in the above, $F(\tau - \beta' x - e)$ replaces $F(\tau - \beta' x)$. In a system of simultaneous equations involving continuous models, logit integrated residuals may be conditional on continuous outcomes. aML will compute the conditional means and (co)variances, and compute the likelihood correspondingly.

model = <*building blocks*>;

The right-hand-side specification must always include a regressor set, parameter, or regressor spline. Non-integrated residuals may not enter logit specifications. (aML always automatically inserts an *iid* logistic residual.) Optionally, you may specify integrated residuals from normal or finite mixture distributions. You may not include duration splines or ARMA(*p*,*q*), or CAR(1) residuals.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models and heteroskedasticity; see Section 13.3.5.

13.8. Ordered Logit Models

This section documents ordered logit (logistic regression) models. Section 13.7 discusses simple (binary) logit models, Section 13.14 multinomial logit models. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

Thresholds to be estimated:

```
ordered logit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcomes = varname1 varname2;
   thresholds = {vectorname | (refvar=varname)};
   model = <building blocks>;
```

Thresholds are given by data variables:

```
ordered logit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   threshold vars = varname1(-Inf=x1) varname2(Inf=x2);
   model = <building blocks>;
```

Instead of "`logit`", you may write "`logistic`".

Unique Options and Features

An ordered logit model is used to specify a model with an ordered qualitative outcome, i.e., to specify the probability that a random variable (or combination of random variables) is within a specified range of the real line. The range may have known or unknown (to be estimated) thresholds, which influences parameter identification. Residuals must be integrated out.

Underlying all ordered logit equations is an underlying propensity equation. For example, a very simple propensity equation is given by:

$$y^* = \beta' x + u$$

The value of the ordered logit outcome variable depends on the range in which y^* falls. The density and cumulative distribution functions of u are

$$f(u) = \frac{\exp\{-x\}}{\left(1+\exp\{-x\}\right)^2} \text{ and } F(u) = \frac{1}{1+\exp\{-u\}},$$

respectively. The logistic density is also known as the sech2 or Fisk density (Johnson and Kotz, 1970, volume 2).

The right-hand-side specification of an ordered logit model must always include a regressor set, parameter, or regressor spline. Optionally, you may specify integrated residuals from normal or finite mixture distributions. You may not include duration splines or non-integrated residuals from normal, ARMA(*p*,*q*), or CAR(1) distributions. An ordered logit model with integrated normal residual is sometimes referred to as the normal probability ordered logit model.

aML supports ordered logit models with unknown thresholds that are the same for all observations and that may need to be estimated, and ordered logit models with known thresholds that may vary by observation. The next two subsections discuss these model types in turn.

13.8.1. Ordered Logit Models with Unknown Thresholds

```
ordered logit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcomes = varname1 varname2;
   thresholds = {vectorname | (refvar=varname)};
   model = <building blocks>;
```

The data structure specification, keep/drop condition, and numerator/denominator statement are described in Section 13.3.1. The other statements are documented here.

`outcomes = varname1 varname2;`

The ordered logit outcome variable, with unknown thresholds, is defined to be a positive integer in a pre-specified contiguous range, say $Y = 1,2,\cdots,K$, or some range of these values. For example, individuals may be asked to rank their health status as poor, fair, good, very good, or excellent. Underlying their response is actual health status, presumably a continuous variable. Depending on the continuous value of health status and the respondent's perception of what thresholds separate poor, fair, good, very good, and excellent health, the respondent will provide a health category. For estimation, the categories need to be converted to contiguous integers, such that the lowest category is assigned a "1" and higher categories higher values. But what if a respondent is not sure about his category, or refuses to narrow it down? For example, he only states that his health is good or very good. aML is capable of dealing with such ranges of responses. In fact, it always requires you to specify a range. Most of the time, that range consists of just one category, but any contiguous range is acceptable.

The outcome is represented by two positive integer-valued variables (or expressions), denoted above by *varname1* and *varname2*. The values of *varname1* and *varname2* represent the subscript numbers of the lower and upper thresholds, respectively, of the interval in which an underlying real values index function value lies, as explained below.

thresholds = {*vectorname* | (refvar=*varname*)};

The "threshold" statement in the binary logit model is optional; the "thresholds" statement here is mandatory. The threshold values are parameters that are organized in a vector, and may be estimated (or fixed at pre-specified values). For example:

```
define vector Cutoffs; dim=4;
ordered logit model; ...;
   thresholds = Cutoffs;
```

or using indirect referencing, for example on the basis of age:

```
define vector Young; ref=1; dim=4;
define vector Old;   ref=2; dim=4;
ordered logit model; ...;
   thresholds = (refvar = 1+(age>=70));
```

The thresholds must be strictly ordered. To prevent thresholds from crossing over during the iterative search procedure, we recommend that the vector be defined with the "increasing" option, which is done by default (page 296). The program will abort with an error message should the thresholds not be strictly ordered.

The following logic underlies these definitions. Let the real-valued continuous index value underlying the qualitative outcome be y^* and the positive integer valued outcome be denoted $y\,(=1,2,\cdots,K)$. The correspondence between the K values of the outcome variable y and K intervals on the real line is determined by K-1 thresholds and is given by

$$y^* = \beta' x + u, \quad y = \begin{cases} 1 & \text{if } \tau_0 \le y^* < \tau_1 \\ 2 & \text{if } \tau_1 \le y^* < \tau_2 \\ \vdots & \vdots \\ K & \text{if } \tau_{K-1} \le y^* < \tau_K \end{cases}$$

The end-points of the first and last intervals are defined to by $\tau_0 = -\infty$ and $\tau_K = \infty$, respectively. Verify that this leaves K-1 thresholds to distinguish K categories.

It may seem redundant to specify two variables in order to represent a single categorical outcome variable. The reason for taking this seemingly complicated route is the ability to handle censored and multi-category cases. The formulation is flexible in that outcomes corresponding to a set of contiguous integers is easily represented by specifying the subscripts of the thresholds between which the real-valued index y^* must lie. Examples include

$$y=\begin{cases}1 & \text{if } \tau_0 \le y^* < \tau_1 & \text{varname1}=0 & \text{varname2}=1\\ j & \text{if } \tau_{j-1} \le y^* < \tau_j & \text{varname1}=j-1 & \text{varname2}=j\\ 3,4 & \text{if } \tau_2 \le y^* < \tau_4 & \text{varname1}=2 & \text{varname2}=4\\ 7-9 & \text{if } \tau_6 \le y^* < \tau_9 & \text{varname1}=6 & \text{varname2}=9\\ m-n & \text{if } \tau_{m-1} \le y^* < \tau_n & \text{varname1}=m-1 & \text{varname2}=n\\ 1-5 & \text{if } \tau_0 \le y^* < \tau_5 & \text{varname1}=0 & \text{varname2}=5\\ 8-K & \text{if } \tau_7 \le y^* < \tau_K & \text{varname1}=7 & \text{varname2}=K\end{cases}$$

Recall that *varname1* and *varname2* may be expressions. If the data only contain single-category outcomes in, say, variable `category`, the outcomes may be conveniently specified as:

```
outcomes = category-1  category;
```

The likelihood of an outcome (integer of range of integer values) is given by the probability of the corresponding interval on the real line. Take the simple example of $y^* = \beta' x + u$, then

$$P\left(\tau_j \le y^* < \tau_k\right) = F\left(\tau_k - \beta' x\right) - F\left(\tau_j - \beta' x\right),$$

where $F(\cdot)$ denotes the cumulative logistic function, $F(u) = \left(1+\exp\{-u\}\right)^{-1}$.

If there is an integrated residual in the ordered logit equation, say, ε, the likelihood modules becomes conditional on the Gauss-Hermite support point, say, *e*, and in the above, $F(\tau - \beta' x - e)$ replaces $F(\tau - \beta' x)$. In a system of simultaneous equations involving continuous models, logit integrated residuals may be conditional on continuous outcomes. aML will compute the conditional means and (co)variances, and compute the likelihood correspondingly.

model = <*building blocks*>;

The right-hand-side specification must always include a regressor set, parameter, or regressor spline. Non-integrated residuals may not enter ordered logit specifications. (aML always automatically inserts an *iid* logistic residual.) Optionally, you may specify integrated residuals from normal or finite mixture distributions. You may not include duration splines or ARMA(*p*,*q*), or CAR(1) residuals.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models and heteroskedasticity; see Section 13.3.5.

13.8.2. Ordered Logit Models with Known Thresholds

ordered logit model;

```
[data structure = n;]
[{keep | drop} if condition;]
[numerator;] [denominator;]
threshold vars = varname1(-Inf=x1) varname2(Inf=x2);
model = <building blocks>;
```

The data structure specification, keep/drop condition, and numerator/denominator statement are described in Section 13.3.1. The other statements are documented here.

threshold vars = *varname1*(-Inf=*x1*) *varname2*(Inf=*x2*);

Ordered logit models with known thresholds are fully analogous to their counterparts with unknown thresholds, except that the thresholds are known real numbers given in the data. There is no "outcome" statement; the outcome consists of an interval of which the lower and upper bound are specified in the "threshold vars" statement. The lower bound is *varname1*, which may be a variable name or an expression; the upper bound *varname2* may also be an expression. These values may be different or may be the same across observations or evaluations within observations.

Ordered logit models with known thresholds are rarely estimated because of scale issues. The standard deviation of the logistic residual in (ordered) logit models is fixed and cannot be estimated. This property makes parameter estimates sensitive to the scale (metric) of the data variables that represent the thresholds. For this reason, ordered probit models (with estimable standard error) are almost always preferred. See Section 13.6.2.

The syntax and algorithms of ordered logit models with known thresholds are analogous to that of their ordered probit counterparts (Section 13.6.2).

13.9. Hazard Models

This section documents hazard models. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Hazard models, also known as failure time or intensity models, are used in situations where the outcome of interest is a duration until the occurrence of some event: a recovery from an illness, a death, a birth, a marriage, a divorce, a machine failure, a change of jobs, et cetera (Kalbfleisch and Prentice, 1980). The hazard at time *t* is the probability density of the event's occurrence at time *t*, conditional on the fact that the event did not take place before time *t*. The period between the moment at which the event became at risk of occurring and the actual occurrence is known as a spell or episode. We often deal with survey data in which the event of interest has not taken place yet: the patient has not yet recovered, the couple is still married, et cetera. Such spells are known as open or censored spells. Uncensored spells are also known as closed spells. The outcome of a hazard process is thus a combination of (1) an indicator variable for whether the spell is censored or not, and (2) a duration between the moment at which the event became at risk and either the timing of the event (uncensored spell) or the survey date (censored spell).

Model Statement

```
hazard model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   censor = varname;
   duration = [durvar1 durvar2];
   [timemarks = varname;]
   model = <building blocks>;
```

Unique Options and Features

aML only supports hazard models with piecewise-linear log-hazard duration dependencies, i.e., generalized Gompertz models. The general formulation (without interactions) is:

$$\ln h_j(t) = \gamma T_j(t) + \beta' X_{jt} + \varepsilon ,$$

where $\ln h_j(t)$ is the log-hazard of spell *j* at time *t*. It is a function of one or more duration dependencies $\gamma T_j(t)$, (potentially time-varying) regressors $\beta' X_{jt}$, and heterogeneity ε. Time is written as an argument in $T(t)$ to indicate that it varies continuously over the duration of the spell, and as a subscript in X_{jt} to indicate that covariates may vary over the duration of the spell, but

must be constant over a finite number of intervals within the spell. Explanatory covariates add to the log-hazard and thus shift the hazard proportionally.

Hazard models in aML are continuous-time hazard models. It is, however, easy to construct special cases which are discrete-time hazard models, either by using logit or probit models for each period, or by using a hazard model with a constant baseline hazard for each period.

The aML hazard models that most users estimate are proportional hazard models. However, aML allows covariates that are interacted with duration effects, $\left(\beta' X_{jt}\right)\left(\gamma T_j(t)\right)$, thus tilting the baseline log-hazard pattern. With this type of model, aML supports some non-proportional hazard models.

There may be multiple sources of duration dependence. For example, the hazard of mortality may be a function of the person's age (to reflect the general risk pattern) and calendar time (to reflect time trends, for example due to medical innovations.) Using other software packages, one typically accounts for the effect of calendar time by including year of birth as a non-time-varying covariate (in which case medical innovation during the person's life is ignored) or current year as a time-varying covariate (in which case the effect of calendar time is a discrete step function.) Incorporating calendar time as a duration dependency captures its effect continuously. Furthermore, there is no need for creating time-varying covariates, so the data remain compact. We often refer to multiple duration dependencies as multiple "clocks", because the various effects start at different points in time.

There may be heterogeneity in the hazard of the event occurrence. Both the normal and finite mixture distributions are supported. In the equation above, we denoted heterogeneity by ε, perhaps hiding that there may be multiple heterogeneity components; that they may be part of a multivariate distribution and be correlated with residuals in other processes; and that there may be residuals with independent draws across hazard equations. As a general rule, the data must contain at least some observations with two or more uncensored hazard spells to which a particular draw applies. It is very difficult to identify heterogeneity if there is only one hazard spell per draw. In such cases, the standard deviation of the heterogeneity component will tend to go to zero. If you observe this in your model runs, check whether your heterogeneity is statistically identified.

Conditional on heterogeneity component ε (which may be a vector, or may be interacted with parameters or regressor sets), the likelihood of hazard module *j* is:

$$L_j(\varepsilon) = \begin{cases} S_j\left(t^*, \varepsilon\right) & \text{if the spell is censored at time } t^*; \\ S_j\left(t^l, \varepsilon\right) - S_j\left(t^u, \varepsilon\right) & \text{if the event occurred between } t^l \text{ and } t^u, \end{cases}$$

where $S_j(t, \varepsilon)$ is the survivor function at time *t*. In the absence of time-varying covariates,

$$S_j(t, \varepsilon) = S_{0j}(t)^{\exp\{\beta' x_j + \varepsilon\}}$$

where $S_{0j}(t)$ is the baseline survivor function at time *t*, i.e., the survivor function based on the baseline duration dependency (or dependencies) only:

$$S_{0j}(t) = \exp\left\{-\int_{\tau=t_b}^{t} h_{j0}(\tau)d\tau\right\},$$

where $\ln h_{0j}(t) = \gamma T_j(t)$ and t_b denotes the beginning of the hazard spell. In the presence of time-varying covariates, the survivor function is given by:

$$S_j(t,\varepsilon) = \prod_{i=1}^{n}\left(\frac{S_{0j}(t_i)}{S_{0j}(t_{i-1})}\right)^{\exp\{\beta' x_{jt_i}+\varepsilon\}}$$

where the period between the beginning of the spell and time *t* is divided into *n* intervals within which time-varying covariates are constant, such that t_0 is the beginning of the spell and $t_n = t$. The window within which an event occurred is delimited by t^l and t^u. These points in time may but need not correspond to time marks t_i.

Conditional on heterogeneity, these likelihood modules are independent. The joint likelihood of multiple hazard spells in the presence of heterogeneity is thus found by numerically integrating-out the heterogeneity from the product of conditional likelihoods of individual hazard modules:

$$L = \sum_{k=1}^{K} w_k \prod_j L_j(e_k),$$

where w_k is the weight corresponding to the *k*-th support point e_k. These support points may be finite mixture points, or Gauss-Hermite approximations to the normal distribution. The integral may consist of multiple intervals if there are multiple heterogeneity components, and may consist of products of sub-integrals if there are multiple draws.

censor = ***varname;***
duration = [***durvar1 durvar2***];

These two statements specify the outcome variables of a hazard process, i.e., the length of the spell and whether it ended with an occurrence of the event of interest.

A hazard spell may be open ("censored") or closed ("uncensored"). If the event occurred at the end of the spell, it is uncensored. If the spell ended without occurrence of the event, it is censored. Whether a spell is censored is specified in the "`censor`" statement with a variable or expression which evaluates to zero (uncensored) or one (censored).

The "`duration`" statement gives the begin and end durations (relative to the moment at which the event became at risk of occurring) of the interval in which the event is known to have occurred. Recognizing that the timing of events is rarely known precisely, aML requires that you

specify a window within which the event occurred. The duration statement therefore consists of two variables or expressions. If the spell is closed, *durvar1* indicates the length of the period between the moment at which the event became at risk of occurring and the lower bound of the event window (t^l, above); *durvar2* indicates the duration from the beginning of the spell to the upper bound of the event window (t^u). If the spell is open, *durvar1* and *durvar2* must be equal to each other, and indicate the duration from the beginning to the end of the spell.

Uncertainty in the end point of the spell (moment at which the event occurred) is incorporated into the hazard probability model by specifying a pair of duration variables, rather than just one variable. Oftentimes, there is additional uncertainty in the begin point of the spell (the moment at which the event became at risk of occurring). We suggest that you start the spell at a best-guess begin date.[36]

All time concepts (clocks) must be measured in the same units (e.g., years, months, days, or minutes). This includes (1) the duration variables indicating when the event occurred or when the spell is censored; (2) timemarks; (3) the initial durations of clocks (duration dependencies) that originated before the spell (e.g., age at the begin date of the spell, marriage duration, calendar time); (4) durations of clocks beginning after the beginning of the spell (e.g., conception dates within a marriage); and (5) nodes in the definition of splines that are used as baseline duration dependencies.

You may select any time unit, as long as you are measuring time consistently across all time variables. The choice is largely arbitrary, as you will obtain substantively identical results regardless of the unit you chose. For most applications, we find that expressing all durations in years works best. Very small units, such as days, sometimes lead to numerical imprecision, especially when the starting values of parameters are poor.

The censor and duration variables must be at the same data level. Typically, they are level 2 variables, but any level is acceptable. The level of the censor and duration variables dictates permissible levels for other variables. For example, it would not make sense to use level 3 variables in a keep or drop statement if the outcome variables are at level 2.

timemarks = ***varname;***

This optional statement is required when your model includes time-varying covariates. Time-varying covariates are variables which are constant over intervals within the hazard spell, but may have different values across intervals. The number of intervals is arbitrary. Intervals need not have equal length, and their lengths matters for the computation of the likelihood. Think of intervals as delimited by time marks. The first interval runs from the beginning of the spell to the

[36] Short of integrating over all possible begin dates (which requires information on the distribution of the begin date and possibly additional covariate values), we know of no satisfactory method for incorporating this type of uncertainty.

first time mark; the second interval runs from the first through the second time mark ; et cetera. In the formulae above, these time marks were denoted by $t_0, t_1, ...$

Since there may be multiple intervals per spell, time marks (and time-varying covariates) are implemented by an additional data level. Time marks and time-varying covariates must be one level lower than the model level. For example, if the censor and duration variables are at level 2, time marks and time-varying covariates must be at level 3.[37] If there are, say, *n* intervals in a hazard spell, then there must be (at least) *n* subbranches. The time marks must be represented by a single variable. The "`timemarks`" statement thus does not accept expressions or multiple variable names.

The first time mark should not be in the data, because it always corresponds to the beginning of the spell, $t_0 = 0$. With *n* intervals within a spell, there are thus *n* (level 3) subbranches to the (level 2) data branch that contains the censor and duration variables. The *n* subbranches contain *n* time marks $t_1, ..., t_n$ and *n* sets of time-varying covariate values corresponding to their values in the *n* intervals.

Time marks must be measured in the same time unit as the duration variables (typically, in years), and are interpreted relative to the beginning of the spell. Time marks must thus all be positive; a value of, say, 3 implies 3 time units (years) since the beginning of the spell. Time marks must also be increasing, though not necessarily strictly increasing. The last time mark must be at least as large as the upper bound of the event window (t^u, specified as `durvar2`); otherwise, the program would not know the value of time-varying covariates beyond the last time mark. Time marks beyond t^u are ignored.

`model = <building blocks>;`

Hazard process outcomes may be explained by duration dependencies $\gamma T_j(t)$, covariates $\beta' X_{jt}$, and heterogeneity ε. These translate into the following building blocks. Duration dependencies are "duration splines," based on defined splines. Covariates typically enter through regressor sets, but regressor splines are also accepted. Heterogeneity is in the form of one or more residuals. Since a closed form solution of the likelihood function is generally not available, these residuals must be integrated out numerically. Heterogeneity thus enters in the form of integrated residuals. These may be part of a normal or a finite mixture distribution.

Regressor splines are allowed, but they rarely offer any added benefit over regressor sets with splines. The main difference between duration splines and regressor splines is that the effect of a duration spline continuously changes over the duration of the spell. The effect of a regressor spline is constant or stepwise constant. It is determined by the value of the transformation variable

[37] Section 10.5 suggests that you may collapse all levels to level 2, if this is convenient given the structure of your (SAS, Stata, SPSS) data. This is not an option for the level containing time marks and time-varying covariates.

as of the beginning of the spell (if the variable is at the same or higher level as the censor and duration variables), or stepwise throughout the spell (if the variable is time-varying).

There must always be at least one duration spline. All other building blocks are optional. Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models, heteroskedasticity in variance components, and even non-proportional hazard models; see Section 13.3.5. Specifically, the interaction of a regressor set and a duration dependency, $\left(\beta' X_{jt}\right)\left(\gamma T_j(t)\right)$, results in a non-proportional hazard model. In that model, covariates do not proportionally shift the hazard, but instead tilts the baseline duration dependency.

Baseline Duration Pattern

The baseline duration pattern is the model's dependency on time without any covariates or heterogeneity. In the model above, it is represented by $\gamma T_j(t)$. Unlike most other hazard software packages, aML allows for multiple additive duration patterns, i.e., a dependency on multiple durations (since the time the event became at risk of occurring, since birth, since marriage, since divorce, since graduation, since some event that occurred during the spell, et cetera).

All hazard model statements must have at least one duration dependence, which must be piecewise-linear (piecewise-Gompertz). A constant baseline hazard may be achieved by defining a spline with intercept and without nodes, and fixing the slope coefficient to zero. A Gompertz (linear) log-hazard may be specified by defining a spline without nodes, so that the slope is the Gompertz slope. A stepwise-constant hazard may be achieved by estimating regression coefficient on time-varying indicator variables (possibly on-the-fly off a single time-varying covariate) which flag individual segments. (For technical reasons, you would still need to include a dummy duration spline with slope fixed to zero.)

Piecewise-linear duration patterns are very attractive because they adjust to any pattern in the data (with sufficiently many nodes), and because linear combinations of piecewise-linear patterns are again piecewise-linear. Formally, the baseline hazard duration is based on the following transformation of the spell duration (the duration since the moment that the event became at risk of occurring), *t*:

$$T(t) = \begin{pmatrix} \min\left[t, \nu_1\right] \\ \max\left[0, \min\left[t-\nu_1, \nu_2-\nu_1\right]\right] \\ \vdots \\ \max\left[0, \min\left[t-\nu_{n-1}, \nu_n-\nu_{n-1}\right]\right] \\ \max\left[0, t-\nu_n\right] \end{pmatrix},$$

where $\nu_1, \nu_2, \cdots, \nu_n$ denote the nodes. The vector of slopes, γ thus consists of n+1 slope coefficients(!) aML allows arbitrary numbers of nodes. If the spline was defined with an intercept, vector $T(t)$ contains an extra "1" as its first element; γ then has dimension n+2, where the first element is the intercept.

Duration dependencies are specified with "`duration spline`" building blocks, abbreviable to "`durspline`". The word "duration" serves to distinguish duration splines from regressor splines. Duration splines are specified as follows:

`durspline(origin=`*`varname`*`, ref=`*`splinename`*`)`

or

`durspline(origin=`*`varname`*`, refvar=`*`varname`*`)`

where the *`varname`* in both the origin and refvar may be the name of a variable or an expression. Very commonly, hazard models contain a duration spline with "`origin=0`" to indicate that the clock starts ticking at the beginning of the spell. The difference between direct (`ref`) and indirect (`refvar`) referencing is explained in Sections 13.3.3 and 13.3.4.

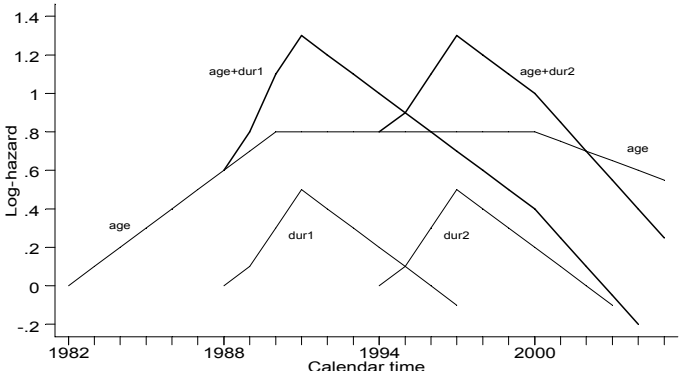
The "origin" specification indicates how long the duration clock has been ticking before the hazard spell starts. Consider a model of fertility that focuses on the conception of second and higher-order children. The hazard spells run from the birth of the previous child to the conception of the index child (if the conception takes place) or to the last time the woman is observed (if censored). The main duration dependency is on duration since the beginning of the spell, capture. In addition, the mother's age matters, measured since, say, her twelfth birthday. Also, for married women, the duration since the wedding. Time trends that are not explained by covariates may be captured by yet another duration dependency. Explanatory covariates are defined in a regressor set, and there may be unobserved mother-specific heterogeneity. Variables `censor`, `time_low`, and `time_up` represent whether the spell is censored and when the event occurred:

```
define durspline SpellDur; intercept; nodes = ...;
define spline MotherAge; nodes = 8 18;
define spline MarDur; ref=1; nodes = ...;
define spline Time1980; nodes = -10 5;
define regset BetaX; var = ...;
define normal distribution; numintpoints=6; dim=1; name=eps;

hazard model;  keep if (parity>=2);
   censor=censor;  duration=time_low time_up;
   model = durspline(origin=0, ref=SpellDur) +
           durspline(origin=age-12, ref=MotherAge) +
           durspline(origin=mardur, refvar=married) +
           durspline(origin=year-1980, ref=Time1980) +
           regset BetaX +
           intres(draw=1, ref=eps);
```

The example concisely illustrates many features. For details please refer to sections above pertaining to splines (13.2.3), regressor sets (13.2.2), and normal distributions (13.2.6). The main duration dependency is on spell duration (`SpellDur`). We defined its spline with an intercept; equivalently, we could have defined the regressor set with an intercept. Note that its origin corresponds to the beginning of the spell (`origin=0`). The clock on maternal age (spline `MotherAge`) starts ticking at the woman's twelfth birthday (`origin=age-12`), so the nodes at 8 and 18 years correspond to ages 20 and 30. The marriage duration spline (`MarDur`) should only apply for married women. We therefore indirectly referenced it with reference variable "`married`". If this variable is one, aML substitutes the spline that has a reference variable equal to one, which is `MarDur`. If `married=0`, the spline drops out of the equation (see Section 13.3.4). To make sure you really intended this effect, aML by default checks that the origin variable, `mardur`, is 99999 if the reference variable is zero. You may switch off this check; see the `check99999` option on page 281. The time trend is defined relative to 1980 (`origin=year-1980`). The nodes of the Time1980 spline thus correspond to 1980-10=1970 and 1980+5=1985. We label the use of multiple duration dependencies as "overlapping splines." The regressor set, `BetaX`, should not contain an intercept since one of the splines (`SpellDur`) is already defined with an intercept. It is completely arbitrary whether you specify an intercept as part of a spline or a regressor set, provided that the building block always enters the equation. The integrated residual is specified with "`draw=1`" so that all hazard spells receive the same draw of this component. (That same draw should also enter the hazard spell for the first conception, not shown here.)

The intercept in this example corresponds to the log-hazard at the beginning of the spell if the woman were 12 years of age, if she were unmarried or married on the day of the beginning of the spell, if it were 1980, and if all covariates were zero. We arbitrarily selected maternal age 12 and calendar time 1980 as reference points. If you change these reference points, the intercept will adjust accordingly, and all other coefficients will remain unchanged.



The figure illustrates overlapping splines. We show just two duration dependencies, namely on maternal age and duration since the last birth. Consider two women that are both born on January 1, 1970. Assume the age effect is linearly increasing from age 12 to 20, constant until age 30, and decreasing thereafter; see the "age" pattern in the figure. Assume that the risk of conceiving due to duration since the last birth increases moderately for twelve months, then more steeply for two years, and decreases thereafter. One woman gives birth to her first child on her 18-

th birthday, so her second conception spells starts on 1/1/1988; see "dur1". Her age and spell duration effect combine to form the overall baseline duration dependency, "age+dur1". Similarly, the other woman gave birth to her first baby and started her second conception spell on 1/1/1994; see "dur2." Her combined baseline dependency is illustrated by "age+dur2." Due to differences in their ages at the beginning of the spell, the two women experience different baseline dependencies. This difference allows for separate identification of age and spell duration patterns. Other duration dependencies may be added analogously. Section 5.9 contains a more elaborate illustration.

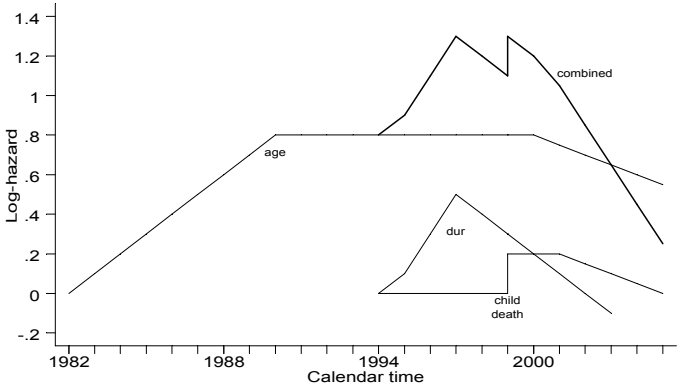
In the example, most duration clocks started ticking before or at the beginning of the spell, i.e., the origin variable was positive. (The exception is the origin of time, 1980, which may be before, during, or after the spell; see below.) However, the origin of a clock may also be sometime during the spell. For example, a woman or couple may react to the death of a previous child by attempting to replace the lost child (Panis and Lillard, 1993). Suppose there are up to eight children per woman, each of which may die and affect subsequent fertility behavior. Their effects may be captured by:

```
define spline ChildDeath; ref=10; nodes=...;
    intercept; effect=right;
<other definitions>
hazard model;
   <other statements>
   model = durspline(origin=kiddur1, ref=10*kiddied1) +
           durspline(origin=kiddur2, ref=10*kiddied2) +
           durspline(origin=kiddur3, ref=10*kiddied3) +
           durspline(origin=kiddur4, ref=10*kiddied4) +
           durspline(origin=kiddur5, ref=10*kiddied5) +
           durspline(origin=kiddur6, ref=10*kiddied6) +
           durspline(origin=kiddur7, ref=10*kiddied7) +
           durspline(origin=kiddur8, ref=10*kiddied8) +
           <other building blocks>;
```

We defined one spline, ChildDeath, to capture the effect of a child's death. It should only enter the equation for couples that experience a child's death. We therefore reference it indirectly using a reference expression equal to 10 times an indicator variable for whether a specific child died (kiddied1 through kiddied8). We multiply it by 10 so that its value does not conflict with the marriage indicator "married", above, which conditionally enters the marriage duration spline. The origin variable indicates the period from the beginning to the spell until the death of each child. If a child died before the hazard spell (say, the first child dies before the second child is born, and the index spell is the waiting time until the third conception), the origin variable should be positive and the effect will be felt throughout the spell. If a child died sometime during the spell, the origin variable should be negative so that its effect kicks in during the spell. If the effect should not extent back to the period before the origin, such as in this example, the spline must be defined with the "effect=right" option. The default (effect=full) is to extrapolate

backward; see page 293. We wanted that default behavior for the calendar time effect, which should apply both before and after its origin, 1980. Here, however, we want the effect of a child's death to generate a jump in the hazard. The intercept that is part of the `ChildDeath` spline generates the jump. Note that multiple children may die, in which case the `ChildDeath` spline enters multiple times in the equation. If a child dies after the end of the index hazard spell, the absolute value of the origin variable exceeds the duration variables, and the spline has no effect. If a child does not die, as is of course the most common case, the `kiddied`*n* indicator variable should be zero. The `kiddurn` variable is then without meaning. To confirm the integrity of the data, aML will check that the `kiddurn` variable is equal to 99999. Also, it will check that relevant origins are never equal to 99999. You may turn off those checks using the check99999 option; see page 281 and Section 13.2.3.

The figure illustrates the effect of a spline that "kicks in" during the spell. It takes the example above of the woman whose second conception spell starts on 1/1/1994, and adds a duration dependence on the time since an older child dies. Assume that the effect of a child's death is to make the hazard jump up, stay constant for two years, and then taper off. The figure shows this dependency for a child that dies five years into the conception spell. The origin variable must thus be –5. Note that its effect on the combined baseline pattern is to push the declining log-hazard sharply up.



It is important to understand the difference between a duration spline and a spline which enters through a regressor set or regressor spline. In a duration spline, the variable which is transformed is the duration since the beginning of the spell (possibly starting at some non-zero value, per the origin specification). That duration varies continuously over the life of the spell. In a regressor spline, the variable which is transformed is constant over the entire spell (if the variable is at the same level as the censor and duration variables) or changes discretely (if the variable is time-varying). For example, if the hazard of mortality is presumed to be affected by medical innovations as they take place over the lifetime of the person, the effect of calendar time should be a duration spline. If the hazard of mortality is presumed to be affected by the state of medical technology as of the person's birth, and subsequent medical innovations do not affect the person's life expectancy, then the effect of calendar time (birth cohort) should be in a regressor spline. In that case, a snapshot is taken and applied over the entire spell. The time transformation may equivalently be part of a regressor set.

While there are few or no good reasons to use regressor splines in hazard models, they provide a flexible tool in systems of equations in which a hazard model is combined with some other model type. For example, suppose the decision of an elderly person to move in with his adult children (a probit model) is in part determined by his log-hazard of dying at the current time. A hazard equation for mortality identifies an age pattern and other determinants; that same age pattern now affects a probit propensity. The age pattern is estimated off the mortality process as a duration spline; the same functional form now enters the probit as a regressor spline.

13.10. Binomial Models

This section documents binomial models. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
binomial model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = varname;
   exposure = varname;
   probability ={linear | logistic | normprob}(<building blocks>);
```

where both the outcome and the exposure may also be specified in terms of an expression rather than a single variable. For example:

```
exposure = exp(varname);
```

Unique Options and Features

Binomial models are used to specify the probability of an observed count value, specified in the outcome statement, as a function of the probability of an occurrence and the number of exposures. The count outcome may be 0, 1, …, *n*, where *n* is the exposure.

The model is specified as follows. Let *Y* denote the non-negative integer-valued outcome, *N* the non-negative integer-valued number of trials or exposures ($0 \le Y \le N$), and *p* the probability of a "success," then the probability of the observed outcome (number of successes in *N* trials) is given by

$$\Pr(Y = y) = \binom{N}{y} p^{y} (1-p)^{N-y} \quad \text{where} \quad \binom{N}{y} = \frac{N!}{y!(N-y)!}.$$

The probability *p* may be a (possibly nonlinear) function of data variables and integrated residuals.

outcome = *varname*;

The outcome (number of successes), *Y*, may be specified as a variable or expression that evaluates to a non-negative integer.

exposure = *varname*;

The exposure (number of trials), *N*, may be specified as a variable or expression that evaluates to a strictly positive integer. It should be at least as large as the outcome.

probability = {linear | logistic | normprob}(<*building blocks*>);

The probability of a success may be specified as a function of building blocks. At a minimum, it must contain a regressor set, parameter, or regressor spline. It may also contain integrated residuals of normal or finite mixture distributions. Non-integrated residuals and duration splines are not allowed.

You may specify the probability as equal to the sum of the building blocks, or as a logistic or cumulative normal probability transformation. To illustrate, suppose the only building block is regressor set `BetaX`, mathematically represented by $\beta' x$. The three options are:

- "`probability = linear(regset BetaX)`" corresponds to $p = \beta' x$. In other words, the linear "transformation" does not transform the building blocks.
- "`probability = logistic(regset BetaX)`" corresponds to the logistic transformation $p = \left(1 + \exp\{-\beta' x\}\right)^{-1}$.
- "`probability = normprob(regset BetaX)`" corresponds to the cumulative normal probability transformation $p = \Phi\left(\beta' x\right) = \int_{u=-\infty}^{\beta' x} \left(2\pi\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}u^2\right\} \mathrm{d}u$. This transformation is also used in probit models, but other than that, there is no relationship with probit models.

Both the logistic and normal probability transformations guarantee that the probability is between zero and one. By contrast, the linear "transformation" carries the risk that the probability becomes less than zero or larger than one during the search, for some observations. If the probability is not a function of parameters, you could define a parameter and use it as the probability:

```
define parameter lambda; range=(0,1);
binomial model;  ...;  probability = linear(par lambda);
```

For probabilities that are functions of data variables or variance components, the logistic or probit transformations are better suited.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models and heteroskedasticity in variance components; see Section 13.3.5.

13.11. Poisson Models

This section documents Poisson models. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
poisson model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = varname;
   [exposure = varname;]
   incidence = exp(<building blocks>);
```

where both the outcome and the (optional) exposure may also be specified in terms of an expression rather than a single variable.

Unique Options and Features

Poisson count models are used to specify the probability of an observed count value, specified in the outcome statement, as a function of the incidence of occurrences and, optionally, the level of exposure. The model was first derived by Poisson (1837) as the limit of a sequence of binomial distributions. (Define a binomial distribution with probability $p = \lambda/n$; draw n independent values from this distribution. As n approaches infinity, the number of successes follows a Poisson distribution.) From entirely different starting points, several other scholars arrived at the same distribution. Johnson, Kotz, and Kemp (1992, Chapter 4) provide an interesting historical perspective.

A random variable Y is said to have a Poisson distribution if

$$\Pr(Y = y) = \frac{e^{-\lambda}\lambda^{y}}{y!}, \quad y = 0, 1, 2, \ldots$$

Parameter λ is the incidence rate, i.e., the expected number of occurrences for any one outcome. Oftentimes, the incidence rate is conceptualized as the expected number of occurrences per *period*, and an outcome is the result of exposure to multiple periods. To accommodate this interpretation, we parameterize:

$$\lambda = E \cdot \exp(\beta' x),$$

where E is the exposure and $\exp(\beta' x)$ the (period) incidence rate. For multilevel extensions, aML support residuals. For example:

$$\lambda = E \cdot \exp(\beta' x + \eta).$$

Both the mean and the variance of the Poisson distribution are equal to λ. This is often violated by the data. In particular, the variance often exceeds the mean. In such cases, the outcome is overdispersed and the negative binomial model may be more appropriate. See Section 13.12.

outcome = *varname;*

The outcome (number of successes), *Y*, may be specified as a variable or expression that evaluates to a non-negative integer. There is no upper bound on *Y*.

exposure = *varname;*

Optionally, the user may specify an exposure variable or expression. This is *E* in the equations above. It must be a strictly positive real number. Omitting the exposure statement is equivalent to setting *E*=1.

Some software packages support an offset variable rather than an exposure variable. The offset is simply the natural logarithm of exposure. To see this, re-write:

$$\lambda = E \cdot \exp(\beta' x) = \exp(\ln E + \beta' x) = \exp(offset + \beta' x).$$

In other words, the log-exposure (offset) enters the incidence equation with a parameter that is fixed to one. Indeed, suppose the data contain both exposure variable "`exposure`" and its logarithm, "`offset`". The usual specification:

```
poisson model;
   outcome = varname;
   exposure = exposure;
   incidence = exp(regset BetaX);
```

is equivalent to:

```
poisson model;
   outcome = varname;
   incidence = exp(offset + regset BetaX);
```

incidence = exp(*<building blocks>*);

The (observable part of the period) incidence rate, $\exp(\beta' x)$, must also be strictly positive. To ensure that it is strictly positive, aML requires that you specify it as an exponentiated function of building blocks. In other words, aML uses the log-link function. The building blocks may

consist of regressor sets, parameters, regressor splines, and variables. It may also contain integrated residuals of normal or finite mixture distributions, $\exp(\beta' x + \eta)$. Non-integrated residuals and duration splines are not allowed.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models; see Section 13.3.5.

13.12. Negative Binomial Models

Attention former users of aML Version 1:

There are many ways to parameterize negative binomial models. Versions 1 and 2 follow different parameterizations. See Section 13.12.1 for backward compatibility of Version 2.

This section documents negative binomial models. Only aspects that are specific to these types of models are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
negative binomial model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = varname;
   [exposure = varname;]
   dispersion = {<building blocks> | exp(<building blocks>)};
   incidence = exp(<building blocks>);
```

where both the outcome and the exposure may also be specified in terms of an expression rather than a single variable.

Unique Options and Features

Much like Poisson models, negative binomial count models are used to specify the probability of an observed count value that may take any non-negative value. While the Poisson distribution implies that the mean and the variance are equal, the negative binomial distribution allows the variance to be greater than the mean, i.e., allows for overdispersion. The literature has developed many parameterizations of negative binomial models. A very intuitive derivation builds on a Poisson model with outcome-specific heterogeneity. Consider a Poisson distribution with incidence rate λ. The incidence rate is, in part, unobserved:

$$\lambda = \exp(\beta' x + u) = \exp(\beta' x)\exp(u)$$

If $\exp(u)$ follows a gamma distribution, the resulting distribution is negative binomial (e.g., Greene 2000, Section 19.9.4). The basic parameterization in aML follows from that derivation. The probability distribution of negative binomial outcome Y is:

$$\Pr(Y=y)=\frac{\Gamma\left(y+\frac{1}{\alpha}\right)}{\Gamma(y+1)\Gamma\left(\frac{1}{\alpha}\right)}\theta^{\frac{1}{\alpha}}(1-\theta)^{y},$$

where $\Gamma(\cdot)$ denotes the Gamma function and

$$\theta=\frac{1}{1+E\alpha\exp(\beta' x)}.$$

E is the exposure, α is the dispersion, and $\exp(\beta' x)$ is the observable part of the (period) incidence rate. An outcome may be the result of exposure to multiple periods. The overall incidence rate is thus $E\exp(\beta' x)$. Heterogeneity is allowed in θ; see below.

outcome = *varname;*

The outcome (number of successes), *Y*, may be specified as a variable or expression that evaluates to a non-negative integer.

exposure = *varname;*

Optionally, the user may specify an exposure variable or expression. This is *E* in the equations above. It must be a strictly positive real number. Omitting the exposure statement is equivalent to setting *E*=1.

Some software packages support an offset variable rather than an exposure variable. The offset is simply the natural logarithm of exposure. To see this, re-write:

$$E\cdot\alpha\cdot\exp(\beta' x)=\alpha\cdot\exp(\ln E+\beta' x)=\alpha\cdot\exp(offset+\beta' x).$$

In other words, the log-exposure (offset) may be seen as entering the incidence equation with a parameter that is fixed to one. Indeed, suppose the data contain both exposure variable "`exposure`" and its logarithm, "`offset`". The usual specification:

```
negative binomial model;
   outcome = varname;
   exposure = exposure;   /*  or:  exposure = exp(offset);  */
   dispersion = exp(regset AlphaX);
   incidence = exp(regset BetaX);
```

is equivalent to:

```
negative binomial model;
```

```
outcome = varname;
dispersion = exp(regset AlphaX);
incidence = exp(offset + regset BetaX);
```

dispersion = {<*building blocks*> | exp(<*building blocks*>)};

This statement specifies the dispersion, α in the equations above. The greater α, the greater the variance relative to the mean.

Dispersion may be specified directly (`dispersion=<building blocks>;`) or in exponentiated form (`dispersion = exp(<building blocks>);`). For example, if dispersion is the same for all outcomes, it may be estimated directly as a parameter:

```
define parameter Alpha; range = (0,Inf);
negative binomial model;
   outcome = varname;
   dispersion = par Alpha;
   incidence = exp(...);
```

We restricted the range of the parameter to be `(0,Inf)`, because dispersion must be positive. Dispersion may also be parameterized. For example:

```
define regressor set lnAlpha; var = <varlist>;
negative binomial model;
   outcome = varname;
   dispersion = exp(regset lnAlpha);
   incidence = exp(...);
```

Here we exponentiated the regressor set to ensure that the dispersion remains positive.

Dispersion may be specified as (exponentiated) regressor sets, parameters, regressor splines, and variables. (Integrated) residuals and duration splines are not allowed.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models; see Section 13.3.5.

incidence = exp(<*building blocks*>);

The (observable part of the period) incidence rate, $\exp(\beta' x)$, must also be strictly positive. To ensure that it is strictly positive, aML requires that you specify it as an exponentiated function of building blocks. In other words, aML uses the log-link function. The building blocks may consist of regressor sets, parameters, regressor splines, and variables. It may also contain integrated residuals of normal or finite mixture distributions, $\exp(\beta' x + \eta)$. Non-integrated residuals and duration splines are not allowed.

Recall that the negative binomial distribution may be derived as a generalization of the Poisson distribution, with a partly unobservable incidence rate $\lambda = \exp(\beta' x + u)$. In principle, there is little difference between implied residual *u* and residual η, except that *u* follows the gamma and η the normal or finite mixture distribution. Neither is identified with just one outcome per draw. For η to be identified, it needs to appear multiple times with the same draw, i.e., it needs to appear at a more aggregate level than the outcome. While *u* represents outcome-specific heterogeneity, η may capture heterogeneity at the person level, or at any other level above the outcome level.

Building blocks may be directly or indirectly referenced; see Sections 13.3.3 and 13.3.4, respectively. Building blocks may also be freely interacted, so that you may specify a wide range of models, including nonlinear models; see Section 13.3.5.

13.12.1. Negative Binomial Model in aML Version 1

The literature has derived the negative binomial model for very diverse types of problems, leading to several different parameterizations. aML's parameterization corresponds to (a generalization of) the parameterization that is most commonly used in the economics literature. However, aML Version 1 was different. Both the syntax and parameterization have changed. To run a Version 1 control file and Version 1's algorithms with aML Version 2, add the following to your control file:

```
option version=1;
```

This will make Version 2 behave as-if it were Version 1, thus ensuring full backward compatibility.

"Option version=1" makes aML behave as-if you are running aML Version 1. Negative binomial models are the only models for which this has material consequences. The parameterizations of other models are unchanged. See Section 13.1.33 for additional (minor) implications of "`option version=1`".

Refer to the Reference Manual of aML Version 1 for details on old-style syntax and algorithms. For comparison, the syntax is:

```
negative binomial model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
```

```
outcome = varname;
scale = exp(<building blocks>);
probability ={linear | logistic | normprob}(<building blocks>);
```

The likelihood function is:

$$\Pr(Y=y)=\frac{\Gamma(A+y)}{\Gamma(A)\Gamma(y+1)}p^{A}(1-p)^{y},$$

where $\Gamma(\cdot)$ denotes the Gamma function, A the "scale" and p the "probability." Similar to the (current) binomial model, Version 1 supported three probability transformations:

$$p=\begin{cases}\beta'x & \text{if linear();}\\ 1/\left(1+\exp(-\beta'x)\right) & \text{if logistic();}\\ \Phi(\beta'x) & \text{if normprob().}\end{cases}$$

Similar to the current incidence function, the old-style probability function may include parameters, regressor sets, et cetera, as well as integrated residuals.

An undesirable feature of the Version 1 parameterization is that higher probabilities make greater count outcomes less likely. In the current parameterization, *higher* $\beta'x$ imply (lower θ and a) higher likelihood of *greater* count outcomes.

There is virtually no risk of running your old control files and finding different results due to the new parameterization. If you try running old control files unchanged, aML will complain about the old syntax and suggest that you add "`option version=1`". It is not allowed to mix old-style with current syntax or algorithms.

13.13. Tobit Models

This section documents tobit models. Only aspects that are specific to this type of model are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
tobit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = expr;
   [lower limit = expr;]
   [upper limit = expr;]
   model = <building blocks>;
```

The outcome is typically just the name of a variable, but it may be an expression. For example:

```
outcome = income;
outcome = log(income);
outcome = log(income+sqrt(income^2+1));
outcome = weight/(height^2);
outcome = min(income, 100000);
```

The tobit lower and/or upper limit are typically just scalars, but they, too, may be expressions. While the syntax indicates that both the lower limit and the upper limit statements are optional, aML requires that at least one be present. If only a lower limit statement is specified, the left-censored tobit applies; only an upper limit statement results in a right-censored tobit; and if both lower and upper limit statements are present, the tobit outcome is both left- and right-censored. The most common tobit model is left-censored at zero:

```
lower limit = 0;
```

The right-hand-side of the equation may include parameters (including elements of matrices and inverse matrices), regressor sets and regressor splines, integrated residuals, and non-integrated residuals.

Unique Options and Features

Tobit models must have at least one (non-integrated) residual from a normal distribution with independent draws for every outcome. They may have multiple residuals (variance components) from normal or finite mixture distributions. Residuals from normal distributions may be

integrated out ("intres") or enter directly ("res"). Residuals from finite mixtures must be integrated out.

Discussion

Tobit models are appropriate when an outcome that is inherently continuous is censored from below (left-censored) or above (right-censored). In other words, one only observes a continuous outcome if that outcome falls in a certain range. If the outcome is outside that range, there are two possibilities. First, one may not observe anything about those cases, in which event the truncated normal model is appropriate (Section 5.3). Second, one may observe everything about those cases, except for the outcome. In that event, the censored normal density model, better known as the Tobit model (Tobin 1958; Judge et al., 1988) applies.

Tobit outcomes may be left-censored, right-censored, or both. The general formulation is:

$$y^* = \beta' x + v, \;\; y = \begin{cases} \tau_L & \text{if } y^* \le \tau_L; \\ y^* & \text{if } \tau_L < y^* < \tau_U; \\ \tau_U & \text{if } y^* \ge \tau_U, \end{cases}$$

where τ_L and τ_U are known thresholds, y^* is some latent continuous concept, and y its observed counterpart. The likelihood function is:

$$L = \begin{cases} \Phi\left(\dfrac{\tau_L - \beta' x}{\sigma_v}\right) & \text{if } y^* \le \tau_L; \\ \phi\left(\dfrac{y - \beta' x}{\sigma_v}\right) & \text{if } \tau_L < y^* < \tau_U; \\ 1 - \Phi\left(\dfrac{\tau_U - \beta' x}{\sigma_v}\right) & \text{if } y^* \ge \tau_U. \end{cases}$$

If the outcome is left-censored but not right-censored, the third branch does not apply; if it is right-censored only, the first branch does not apply.

The lower and upper thresholds are specified with the "lower limit" and "upper limit" statements, respectively. Both may be expressions involving variables and scalars. Naturally, any variable in a lower or upper limit statement must be at at least the level of the outcome variable, i.e., at the same level or a more aggregated level. Limit statements that involve a variable may be useful when the limit is data-dependent. For example, consider a model of annual earnings based on the March 1988 Current Population Survey. In that survey, earnings are top-coded at $99,999, i.e., incomes of $100,000 or more are reported as $99,999. In this case, the model may be:

```
tobit model;
   outcome = earnings;
```

```
upper limit = 99999;
model = ...;
```

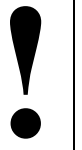
where "earnings" is a variable containing annual earnings. (For simplicity, we ignored possible left-censoring.) Now suppose we estimate a similar model based on earnings data that the U.S. Social Security Administration (SSA) records. Social Security taxes are only levied on earnings up to the so-called contribution and benefit base (maximum taxable earnings). For workers with earnings above the cap, SSA's records show only the cap. The contribution base is changed every year and climbed from $14,100 in 1975 to $84,900 in 2002. Since the maximum amount changes from year to year, an analysis of multiple years needs to account for varying censor levels. Suppose the data contain a variable "cap" with the applicable contribution and benefit base. The model specification then is:

```
tobit model;
   outcome = earnings;
   upper limit = cap;
   model = ...;
```

Perhaps earnings are expressed in real 2000 dollars and you forgot to deflate the cap variable accordingly. If the data contain a variable "cpi" with the Consumer Price Index for the applicable year, with base year 2000, you may write:

```
tobit model;
   outcome = earnings;
   upper limit = cap*100/cpi;
   model = ...;
```

Careful: This specification is risky and may lead to undesired results. Presumably, you converted annual earnings into real 2000 dollars in SAS, Stata, etc., i.e., before creating aML data. You then created an ASCII data file to convert the data with raw2aml into aML format. This may have involved some rounding, and the resulting earnings variable for censored observations may not be exactly equal to cap*100/cpi. If the rounded earnings variable is slightly above the upper limit, all will be fine. But if the rounded earnings variable is slightly below the upper limit, aML will think that this person earned an amount that was just below the cap, and treat it as an uncensored observation.



Be careful with lower and upper limits that are not integer-valued. Rounding error may lead aML to conclude that a certain value is uncensored where it really is censored.

Multivariate Tobit Models

aML supports multivariate tobit models, that is, tobit models of two or more outcomes that are correlated. (aML also supports multilevel tobit models; see below.) For example, suppose we wish to analyze the earnings of married couples, where both his and her earnings are subject to right-censoring at $99,999. Because of assortative mating, shared environments, or other factors, we wish to allow for correlated residuals across spouses. The model may be specified as follows:

```
define normal distribution; dim=2;
   name=u1;
   name=u2;

tobit model;
   outcome = earning1;
   upper limit = 99999;
   model = ... + res(draw=1, ref=u1);

tobit model;
   outcome = earning2;
   upper limit = 99999;
   model = ... + res(draw=1, ref=u2);
```

Since `u1` and `u2` were defined as part of the same distribution and used with the same draw, they will induce correlation in the two tobit models.

For high-earnings couples with both his and her earnings above the top-code, the model reduces to a bivariate probit and the likelihood involves a bivariate cumulative normal distribution. aML knows how to calculate cumulative normal probabilities up to trivariate. If more than three tobit outcomes may be censored, the residual structure needs to be specified such that any correlation arises from integrated residuals only; see the next subsection.

Multilevel Tobit Models

aML supports multilevel tobit models. Mathematically, these are equivalent to the multivariate tobit described in the previous subsection. Suppose we want to analyze a time series of annual earnings of workers using the aforementioned SSA data. The model is:

$$E^*_{it} = \beta' X_{it} + \varepsilon_i + u_{it}, \quad E_{it} = \begin{cases} \beta' X_{it} + \varepsilon_i + u_{it} & \text{if } E^*_{it} < \tau_t \\ \tau_t & \text{if } E^*_{it} \geq \tau_t \end{cases}$$

where E^*_{it} and E_{it} are true and observed earnings, respectively, and τ_t is the cap applicable in year *t*. We could specify:

```
define regset BetaX; var = ...;

define normal distribution; dim=1; name=eps;
```

```
define normal distribution; dim=1; name=u;

tobit model;
   outcome = earnings;
   upper limit = cap;
   model = regset BetaX +
           res(draw=1, ref=eps) +
           res(draw=_iid, ref=u);
```

Residual `eps` enters with `draw=1`, i.e., the same draw applies to all earnings outcomes of a particular observation. Residual `u` enters with `draw=_iid`, i.e., all draws are independent. This formulation is likely to run into troubles. Consider the covariance matrix of $\varepsilon_i + u_{it}$ for a worker who is observed over five years (t=1, …, 5):

$$\Sigma_{\varepsilon+u,\varepsilon+u} = \begin{pmatrix} \sigma_\varepsilon^2+\sigma_u^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2+\sigma_u^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2+\sigma_u^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2+\sigma_u^2 & \sigma_\varepsilon^2 \\ \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2 & \sigma_\varepsilon^2+\sigma_u^2 \end{pmatrix},$$

where σ_ε^2 and σ_u^2 are the variances of ε_i and u_i, respectively. The likelihood for workers with some years of earnings at or above the cap involves cumulative normal probabilities with this type of covariance matrix. As noted above, aML can only evaluate such matrices up to trivariate. More than three years of earnings above the cap will stop the program. It is very easy to circumvent this issue: integrate-out the higher-level residual(s). In other words, the model statement should be:

```
tobit model;
   outcome = earnings;
   upper limit = cap;
   model = regset BetaX +
           intres(draw=1, ref=eps) +  /* intres, not res!  */
           res(draw=_iid, ref=u);
```

The issue is identical to the one resulting from correlated residuals in probit models, discussed in more detail in Section 4.1.

Do-It-Yourself Tobit

The tobit model may be viewed as a switching regression model, where the likelihood switches between a normal density function (for uncensored outcomes) and a cumulative normal probability (for censored outcomes). In other words, a tobit model is simply a combination of a continuous model and a probit model. (Indeed, the word "tobit" is derived from "probit" and the

surname of the scholar who first discussed the tobit model, James Tobin.) Recall the simple tobit model explaining hours worked of Section 2.9:

```
tobit model;
   outcome = hours;
   lower limit=0;
   model = regset BetaX + res(draw=1, ref=v);
```

This model may equivalently be specified as follows:

```
continuous model; keep if hours>0;
   outcome = hours;
   model = regset BetaX + res(draw=1, ref=v);

probit model; keep if hours<=0;
   outcome = hours;
   model = regset BetaX + res(draw=1, ref=v);
```

The continuous model only applies if outcome variable hours is greater than zero, i.e., uncensored. The probit model applies if the outcome is less than or equal to zero, i.e., censored. The rest of the control file is identical to the tobit specification (but see below for integrated residuals). The output will also be identical, except of course for the section that reports model specifications and summary statistics of the outcomes.

You may wonder why we explicitly specified a residual in the probit model, `res(draw=1, ref=v)`. The standard deviation of a residual is not identified in a probit model, and virtually all statistical estimation software packages therefore assume an independent and identically distributed standard normal residual in probit models. In default behavior, aML is no exception (Sections 2.1.4 and 13.5), but aML allows the user to explicitly specify a residual. In the do-it-yourself tobit model, the residual is not standard but has a non-unit standard deviation. This therefore needs to be specified explicitly. By using the same residual `v` in the continuous and probit models, the same standard deviation applies. The standard deviation is identified off uncensored outcomes only.

You may also wonder why we specified "`outcome = hours`" in the probit, where "`outcome = 0`" seems to be equivalent and perhaps more intuitive. In this particular case, either specification would yield the desired result. However, this is only true because the data contain a single level. More generally, aML needs to know how many outcomes there are per observation, i.e., how often a model statement contributes an equation to the likelihood. If an outcome variable is at level 1, there can only be one outcome per observation. If an outcome variable is at a lower (more disaggregated) level, there may be repeated outcomes and thus, somewhat loosely speaking, multiple likelihood modules. If the data contain multiple levels, aML will not accept "`outcome = 0`" because of ambiguity in the number of times that the model statements contributes to the likelihood.

A top-censored tobit model, i.e., a tobit model with an upper limit, may also be re-written as a conditional probit and a continuous model. In that case, the probit outcome must be one. Since it also needs to involve a variable name, one could write "`outcome=`*`varname`*`-`*`varname`*`+1`" or anything else that evaluates to one. A dual-censored tobit with both lower and upper limit may be re-written as two conditional probit models and one conditional continuous model.

The above tobit model had a zero lower limit. For non-zero lower or upper limits, the probit portion of the do-it-yourself specification needs to be adjusted to explicitly state that the threshold is some non-zero number. If the limit does not vary across observations, this is most easily accomplished by the (optional) `threshold` statement. For example,

```
tobit model;
outcome = varname;
lower limit = 40;
model = ...;
```

is equivalent to:

```
continuous model; keep if varname>40;
   outcome = varname;
   model = ...;

define parameter Tau;

probit model; keep if varname<=40;
   outcome = varname-varname;  /* little trick to get zero */
   threshold = Tau;
   model = ...;
```

where parameter `Tau` is initialized to 40 and not estimated. Parameters were introduced above in Section 2.5.1; Section 13.2.1 contains a full description. Section 13.6.1 provides details on the `threshold` statement in probit models.

There is an additional issue with do-it-yourself specifications of multilevel tobit models. As explained above, higher-level residuals must be integrated-out in tobit model specifications because otherwise there may be too many correlated probit modules. Obviously, these integrated residuals must remain integrated-out in the probit branch of the do-it-yourself specification. But what about the continuous branch? Higher-level residuals in continuous models may but need not be integrated-out, because a closed-form solution to the likelihood exists. There is one exception, for technical reasons: higher-level residuals with multiple draws may not be integrated out of continuous modules. This is not an issue with two-level models, because there is only one draw of the higher-level residual (one value per observation). With three or more levels, however, there will be multiple draws for at least one residual (multiple units within one observation and above the lowest level). For that reason, aML does not integrate-out any residuals in the continuous branch of multilevel tobit models. Consider a three-level problem, such as multiple test scores (level 3) by multiple students (level 2), nested in schools (level 1). We want residuals at the

school-level (`eps`), student-level (`eta`), and test-level (`u`). Suppose the data contain variable "`student`" that uniquely identifies students. The following model:

```
tobit model;
   outcome=...;
   lower limit=...;
   model = ... +
           intres(draw=1, ref=eps) +
           intres(draw=student, ref=eta) +
           res(draw=_iid, ref=u);
```

is equivalent to:

```
probit model; keep if ...;
   outcome=...;
   threshold=...;
   model = ... +
           intres(draw=1, ref=eps) +
           intres(draw=student, ref=eta) +
           res(draw=_iid, ref=u);


continuous model; keep if ...;
   outcome=...;
   model = ... +
           res(draw=1, ref=eps) +        /* res, not intres!  */
           res(draw=student, ref=eta) +  /* res, not intres!  */
           res(draw=_iid, ref=u);
```

Again, the noteworthy feature is that integrated residuals in tobit models are not integrated-out in the continuous branch. By its very nature, numerical integration yields an approximation to the likelihood. By not integrating-out any residuals in continuous branches, aML reduces the resulting inaccuracy.[38]

[38] As noted, one could integrate-out higher-level residuals in the continuous branch in problems with only two levels. This is the approach that Stata's `xttobit` command takes. Stata does not support models with three or more levels.

13.14. Multinomial Logit Models

This section documents multinomial logit models. Only aspects that are specific to this type of model are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
multinomial logit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = expr;
   model n1 = <building blocks>;
   model n2 = <building blocks>;
   [model n3 = <building blocks>;]
   [model n4 = <building blocks>;]
   <et cetera>
```

The outcome is typically just a variable, but it may be an expression. Outcomes must be integer-valued.

There should be a model specification for every potential outcome except for the omitted category. Integers *n1*, *n2*, *n3*, *n4*, etc. correspond to the outcomes being distinguished; all other outcome values are grouped in the omitted category. There is no hard limit to the number of categories that may be distinguished.

The right-hand-side of the model equations may include parameters (including elements of matrices and inverse matrices), regressor sets and regressor splines, and integrated residuals. Implicitly, a non-integrated logistic residual is present in each model equation, but this residual should not be specified. Integrated residuals may be distributed normally or as a finite mixture.

Unique Options and Features

Multinomial logit models are appropriate when the outcome of interest may take on a limited number of values that are not ordered. (If they are ordered, the ordered logit or ordered probit model is more appropriate.) For example, someone may choose among various modes of transportation (walking, bicycle, own motorized transportation, public transportation); a household may own a residency, rent a residency, or be otherwise accommodated; a worker may be employed in agriculture, manufacturing, services, or other industry; et cetera. The categories must be mutually exclusive and exhaustive, possibly through the inclusion of a residual category.

Unlike most other model specifications, the multinomial logit model requires multiple model specifications, namely one for each potential outcome, except for the omitted category. If no category is omitted, the model is not identified without additional restrictions on parameters or identification from other parts of a multiprocess model.

Discussion

Consider an outcome that may take *J* distinct values. Denote the probability that choice *j* (*j*=1,…,*J*) is selected by P_j. The likelihood function of the standard multinomial logit model is:

$$P_j = \frac{\exp\{\beta_j' X\}}{\sum_{i=1}^{J} \exp\{\beta_i' X\}}, \quad j = 1,...,J.$$

This model is defined by *J* sets of parameters β_j. However, not all parameters are statistically identified. We typically normalize all parameters associated with a certain category to be zero.[39] Suppose this omitted category is the *J*-th category ($\beta_J = \underline{0}$), so that the likelihood function reduces to:

$$P_j = \begin{cases} \dfrac{\exp\{\beta_j' X\}}{1 + \sum_{i=1}^{J-1} \exp\{\beta_i' X\}} & \text{if } j = 1,...,J-1; \\ \dfrac{1}{1 + \sum_{i=1}^{J-1} \exp\{\beta_i' X\}} & \text{if } j = J \text{ (omitted category).} \end{cases}$$

The choice of omitted category is entirely arbitrary. Also, while the potential choices in our mathematical notation are numbered consecutively from 1 to *J*, no order is implied and any set of integer-valued choices may be specified. Indeed, multinomial models treat all categories on the same footing; any reversal of assigned categories results in equivalent estimates.

Most model estimation software packages require that the same set of explanatory covariates *X* enters in all choice equations. While not expressed in the above likelihood equations, aML is more general in that it accepts different regressors in all choice equations.

Interpretation of multinomial logit parameters is not straightforward. For example, the marginal effect of covariate X_k on the probability that choice *j* will be selected is:

[39] aML does not require that you normalize in this way. You may specify an exhaustive set of choices and impose any number of identifying restrictions in the starting values. Also, in a multiprocess model, you could identify one set of parameters in a related model and estimate the full set of *J* parameters β_j.

$$\frac{\partial P_j}{\partial X_k} = P_j\left(\beta_{jk} - \sum_{i=1}^{J}\beta_{ik}P_i\right),$$

where β_{ik} is the *k*-th element of coefficient vector β_i. This probability depends on the point of evaluation, just like it does in the standard logit model. Furthermore, it depends on all probabilities P_1 through P_J, and can change signs depending on those probabilities. In other words, the multinomial logit model does not share the monotonicity property of standard logit models, where larger values of a covariate with a positive coefficient imply larger values of the probability.

Many people find it easier to interpret multinomial logit parameters through the concept of log-odds ratios. For example, the log-odds ratio of categories *j* and *l* is:

$$\log\left(\frac{P_j}{P_l}\right) = \log\left(\frac{\exp\left(\beta_j'X\right)\Big/\sum_{i=1}^{J}\exp\left\{\beta_i'X\right\}}{\exp\left(\beta_l'X\right)\Big/\sum_{i=1}^{J}\exp\left\{\beta_i'X\right\}}\right) = \left(\beta_j - \beta_l\right)' X .$$

This log-odds ratio depends only on parameters β_j and β_l. If the choice set were to expand or contract from the current *J* options, the log-odds ratio of categories *j* and *l* would not be affected. This property is known as independence from irrelevant alternatives (IIA). It is an undesirable property, because it also applies to highly relevant additional alternatives.[40] The multinomial probit model, by contrast, does not suffer from this property (Section 2.6).

[40] The standard example is the "red bus, blue bus" example. Suppose one chooses among several modes of transportation, say, non-motorized (*n*), own motorized (*o*), and public transportation by bus (*r*), with certain probabilities. Suppose all buses are red. The odds ratio of red bus versus, say, non-motorized transportation, P_r/P_n, reflects the relative preference of riding a red bus versus using non-motorized transportation. Now suppose a new public bus service (*b*) is introduced that is identical to the existing bus service, except that its buses are blue in color. We estimate a new multinomial logit model that includes the new blue bus service as a separate choice. The interpretation of the log-odds ratio of red bus versus non-motorized transportation, P_r/P_n, remains the same as before, namely the relative preference of riding a red bus versus using non-motorized transportation. Its value *should* therefore not change. However, no change in the odds ratio can only occur if the blue buses gain market share by proportional decreases of the market shares of other transportation modes. This is implausible. It is much more likely that blue buses dent the market share of red buses by far more than of other modes. Indeed, in practice, the introduction of a new choice that is similar to an existing choice does affect the odds ratios. This unpatable result is a consequence of the fact that the multinomial probit treats all outcomes on the same footing, without any account of similarities or dissimilarities among potential outcomes. Then why do analysts not worry very much about the undesirable IIA property? Key is to interpret the coefficients in the context of the empirical setting. In other words, instead of assigning a very specific interpretation to an odds ratio ("relative preference of riding a red bus versus using non-motorized transportation"), one should interpret the coefficients as reflective of relative preferences among the alternatives offered. If some alternatives are clearly very similar, consider estimating a multinomial probit model (Section 13.15) or a nested logit model (not supported by aML).

Suppose *l* is the omitted category. The log-odds ratio then simplifies to:

$$\log\left(\frac{P_j}{P_{\text{omitted}}}\right) = \beta_j' X \ .$$

It follows that a one-unit increase in covariate X_k changes odds ratio P_j / P_{omitted} by a factor $\exp\left(\beta_{jk}\right)$. Some people therefore prefer to report exponentiated (multinomial) logit coefficients. It is, of course, straightforward to convert coefficient estimates into their exponentiated counterparts. Note that the standard deviation of $\exp\left(\beta_{jk}\right)$ is $\exp\left(\beta_{jk}\right)\sigma_{\beta_{jk}}$, where $\sigma_{\beta_{jk}}$ is the standard deviation of β_{jk}.

Multilevel Multinomial Logit

aML supports multilevel extensions of multinomial logit models. The outcome variable may be at any level. Repeated outcomes within observations result in a multilevel multinomial logit model. The outcomes may be correlated within observation through residuals (unobserved heterogeneity). For example, the following model specification adds observation-specific unobserved heterogeneity:

```
define normal distribution; dim=1;
   number of integration points=6;
   name=eps;

multinomial logit model;
   outcome = occ;
   model 2 = ... + intres(draw=1, ref=eps);
   model 3 = ... + intres(draw=1, ref=eps);
   model 4 = ... + intres(draw=1, ref=eps);
```

The residual must be integrated-out, since no closed form solution to the likelihood exists. Residuals may be from normal or finite mixture distributions.

The usual rules on residual specifications apply, that is, you may include multiple residuals, multiply them by parameters or regressor sets, etc. There is one restriction: residual draws must be the same in all model specifications of a particular multinomial logit. For example, the following will generate an error message:

```
multinomial logit model;
   outcome = occ;
   model 2 = ... + intres(draw=1, ref=eps);
   model 3 = ... + intres(draw=2, ref=eps);
   model 4 = ... + intres(draw=3, ref=eps);
```

In practice, this is not a restriction; we cannot think of a case in which it would make sense to have multiple draws of residuals belonging to the same distribution in one outcome module. Of course, multiple residual draws may be specified across different outcomes.

Residuals may be correlated with residuals in other model statements, leading to multinomial logit models in a multiprocess setting.

13.15. Multinomial Probit Models

This section documents multinomial probit models. Only aspects that are specific to this type of model are discussed; features that are common to all types of models (such as data structure specification, keep/drop conditions, conditional likelihoods, the use of building blocks, and indirect referencing) are described in Section 13.3.

Model Statement

```
multinomial probit model;
   [data structure = n;]
   [{keep | drop} if condition;]
   [numerator;] [denominator;]
   outcome = expr;
   model n1 = <building blocks>;
   model n2 = <building blocks>;
   [model n3 = <building blocks>;]
```

The outcome is typically just a variable, but it may be an expression. Outcomes must be integer-valued.

There should be a model specification for every potential outcome except for the omitted category. Integers *n1*, *n2*, and *n3* correspond to the outcomes being distinguished; all other outcome values are grouped in the omitted category. There may at most be four distinct categories, i.e., at most three model equation specifications. (The multinomial logit model does not have this limitations; see Section 13.14.)

The right-hand-side of the model equations may include parameters (including elements of matrices and inverse matrices), regressor sets and regressor splines, integrated residuals, and non-integrated normal residuals. At a minimum, there must be a non-integrated, normally distributed residual in each model equation. (In simple and ordered probit models, aML assumes an independent standard normal residual if none is specified. There is no such default behavior in multinomial probit models, in part because correlated residuals are a key feature of multinomial probit models.) These residuals should be different from one another but belong to the same distribution and be specified with the same draw in all model statements of a multinomial probit.[41] Consider a categorical variable that can take on values 0, 1, 2, or 3. We arbitrarily select 0 to be the omitted category. The typical, bare bones multinomial probit model would be:

```
define regset XBeta1; var = ...;
```

[41] Making its usual assumption that the user knows best, aML will not object to multinomial probit specifications with the same residual appearing in multiple model specifications or to uncorrelated residuals. However, it will object to residuals from the same distribution but with different draws—that would not make sense and is not technically feasible.

```
define regset XBeta2; var = ...;  /* same variable list  */
define regset XBeta3; var = ...;  /* same variable list  */
define normal distribution; dim=3;
   name=u1;  name=u2;  name=u3;

multinomial probit model;
   outcome=Y;
   model 1 = regset XBeta1 + res(draw=1, ref=u1);
   model 2 = regset XBeta2 + res(draw=1, ref=u2);
   model 3 = regset XBeta3 + res(draw=1, ref=u3);
```

The variables in the three regressor sets are typically the same, but aML allows different variables. Residuals `u1`, `u2`, and `u3` are correlated because they are defined as part of the same distribution and enter with the same draw. In the starting values, the standard deviations of `u1`, `u2`, and `u3` should be fixed to one for the standard multinomial probit model; aML permits other values and even estimable standard errors, provided they are identified from some other part of a larger model.

Integrated residuals, if any, may be distributed normally or as a finite mixture.

Unique Options and Features

Multinomial probit models are appropriate when the outcome of interest may take on a limited number of values that are not ordered. (If they are ordered, the ordered probit model is more appropriate.) For example, someone may choose among various modes of transportation (walking, bicycle, own motorized transportation, public transportation); a household may own a residency, rent a residency, or be otherwise accommodated; a worker may be employed in agriculture, manufacturing, services, or other industry; et cetera. The categories must be mutually exclusive and exhaustive, possibly through the inclusion of a residual category.

The main difference between a multinomial probit and a multinomial logit model is that the former allows correlation among its residuals. This feature allows it to capture the degree of similarity across alternatives, so that the multinomial logit's undesirable independence of irrelevant alternatives (IIA) property does not apply. Unfortunately, the multinomial probit allows only up to four alternatives (including the omitted category); the multinomial logit does not have this limitation.

Unlike most other model specifications, the multinomial probit model requires multiple model specifications, namely one for each potential outcome, except for the omitted category. If no category is omitted, the model is not identified without additional restrictions on parameters or identification from other parts of a multiprocess model.

Discussion

Consider an outcome that may take *J* distinct values. Each alternative has a pay-off (utility, value, degree of attractiveness). The alternative with the highest pay-off is chosen. We only

observe which alternative is chosen. Nothing is known about the magnitude of the pay-offs, the relative ranking of alternatives that were not chosen, or how much more attractive the chosen alternative was than the others. This implies that two normalizations are needed. Only relative statements can be made, so the pay-off of one alternative needs to be fixed. This is typically done by setting its value to zero. Consider the following *J* pay-off equations:

$$\begin{aligned} y_1^* &= \beta_1' X + u_1 \\ &\vdots \\ y_{J-1}^* &= \beta_{J-1}' X + u_{J-1} \\ y_J^* &= 0 \end{aligned}$$

We arbitrarily normalized the last category, i.e., the last category is the omitted category. Any other omitted category would yield equivalent results.

Since the magnitude of pay-offs is unknown and irrelevant, the scale needs to be normalized. This is typically done by setting the standard deviations of residuals to one: $\sigma_1 = \sigma_2 = \ldots = \sigma_{J-1} = 1$, where σ_j is the standard deviation of u_j. Denote the probability that choice *j* (*j*=1,…,*J*) is selected by P_j. The likelihood function of the standard multinomial probit model is:

$$\begin{cases} P_1 = P\left(y_1^* > y_2^*, y_1^* > y_3^*, \ldots, y_1^* > y_J^*\right) \\ P_2 = P\left(y_2^* > y_1^*, y_2^* > y_3^*, \ldots, y_2^* > y_J^*\right) \\ \quad \vdots \\ P_J = P\left(y_J^* > y_1^*, y_J^* > y_2^*, \ldots, y_J^* > y_{J-1}^*\right) \end{cases}$$

Each probability involves *J*-1 inequalities. In other words, each probability requires the evaluation of a cumulative normal integral of dimension *J*-1. Such integrals become very burdensome for larger values of *J*. aML supports up to trivariate, i.e., there may be at most four alternatives ($J \le 4$).

The coding of alternatives and the choice of omitted category are entirely arbitrary. Regardless of the codes assigned to alternatives, no order is implied. Any set of integer-valued choices may be specified.

Most model estimation software packages that support multinomial probit models require that the same set of explanatory covariates *X* enters in all choice equations. While not expressed in the above pay-off equations, aML is more general in that it accepts different regressors in all pay-off equations.

Multilevel Multinomial Probit

aML supports multilevel extensions of multinomial probit models. The outcome variable may be at any level. Repeated outcomes within observations result in a multilevel multinomial probit

model. The outcomes may be correlated within observation through one or more additional residual (unobserved heterogeneity). For example, consider a model of occupational choice. The unit of analysis is a person; we have panel data with annual information on occupation and other relevant variables. An illustrative model is:

$$\begin{aligned} y^*_{t,1} &= \beta_1' X_t + \varepsilon + u_{t,1} \\ &\vdots \\ y^*_{t,J-1} &= \beta_{J-1}' X_t + \varepsilon + u_{t,J-1} \\ y^*_{t,J} &= 0 \end{aligned}$$

where the *t*–subscript denotes the year of the panel survey. We suppressed the observation (person) subscript. The corresponding data contain two levels. Level 1 is the person, level 2 an annual survey. Key variables at level 2 are `year` (denoting the year of the survey) and `occ` (occupational category in that year). The following model specification estimates the above multilevel multinomial probit model with person-specific unobserved heterogeneity:

```
define normal distribution; dim=3;
   name=u1;  name=u2;  name=u3;
define normal distribution; dim=1;
   number of integration points=6;
   name=eps;

multinomial probit model;
   outcome = occ;
   model 1 = ... +
             intres(draw=1, ref=eps) + res(draw=year, ref=u1);
   model 2 = ... +
             intres(draw=1, ref=eps) + res(draw=year, ref=u2);
   model 3 = ... +
             intres(draw=1, ref=eps) + res(draw=year, ref=u3);
```

Draws of residuals `u1`, `u2`, and `u3` are specified with a variable name (or expression), in this case variable `year`. This ensures that transitory residuals $u_{t,1}$ through $u_{t,J-1}$ are independent across years. The draw of residual `eps` is the same for all replications of the outcome, i.e., the same (person-specific) value of ε applies throughout the panel. Heterogeneity residual `eps` must be integrated-out, because the overall covariance matrix of probit outcomes would otherwise not be block-diagonal any more; see the technical note below. Integrated residuals may be from normal or finite mixture distributions.

The usual rules on residual specifications apply, that is, you may include multiple residuals, multiply them by parameters or regressor sets, etc. There is one restriction: residual draws must be the same in all model specifications of a particular multinomial probit. For example, the following will generate an error message:

```
multinomial probit model;
   outcome = occ;
   model 2 = ... + intres(draw=1, ref=eps) + ...;
   model 3 = ... + intres(draw=2, ref=eps) + ...;
   model 4 = ... + intres(draw=3, ref=eps) + ...;
```

(As noted above, the same restriction applies to non-integrated residuals within a multinomial probit model.) In practice, this is not a restriction; we cannot think of a case in which it would make sense to have multiple draws of residuals belonging to the same distribution in one outcome module. Of course, multiple residual draws may be specified across different outcomes.

Residuals may be correlated with residuals in other model statements, leading to multinomial probit models in a multiprocess setting.

Technical Note: Block-diagonal Covariance Matrices

With a great deal of care, you could estimate multinomial probit models without using the `multinomial probit model` statement. Consider outcome `Y` that may take values 0, 1, or 2. We arbitrarily select 0 as the omitted category. The model is:

$$\begin{cases} y_1^* = \beta_1' X + u_1 \\ y_2^* = \beta_2' X + u_2 \end{cases}$$

$$Y = \begin{cases} 0 & \text{if} \quad y_1^* < 0 \text{ and } y_2^* < 0 \\ 1 & \text{if} \quad y_1^* > 0 \text{ and } y_1^* > y_2^* \\ 2 & \text{if} \quad y_2^* > 0 \text{ and } y_2^* > y_1^* \end{cases}$$

The multinomial probit model specification could be:

```
define regset XBeta1; var=...;
define regset XBeta2; var=...;
define normal distribution; dim=2;
   name=u1;
   name=u2;

multinomial probit model;
   outcome = Y;
   model 1 = regset XBeta1 + res(draw=1, ref=u1);
   model 2 = regset XBeta2 + res(draw=1, ref=u2);
```

Now let's study the likelihood function carefully. For example, alternative 1 is chosen if

$$y_1^* > 0 \text{ and}$$
$$y_1^* > y_2^* \quad \Leftrightarrow \quad y_1^* - y_2^* > 0$$

We can write this as two simple probits:

```
/*  Specify y1>0  */
probit model;  keep if (Y==1);
   outcome = (Y==1);
   model = regset XBeta1 + res(draw=1, ref=u1);

/*  Specify y1-y2>0  */
probit model;  keep if (Y==1);
   outcome = (Y==1);
   model = regset XBeta1 + res(draw=1, ref=u1)
         - regset XBeta2 - res(draw=1, ref=u2);
```

Writing probit models for the other alternatives in a similar manner, and condensing things a little, the entire multinomial probit model could equivalently be specified as:

```
/*  Specify y1<>0  */
probit model;  keep if (Y==0 or Y==1);
   outcome = (Y==1);
   model = regset XBeta1 + res(draw=1, ref=u1);

/*  Specify y2<>0  */
probit model;  keep if (Y==0 or Y==2);
   outcome = (Y==2);
   model = regset XBeta2 + res(draw=1, ref=u2);

/*  Specify y1-y2<>0  */
probit model;  keep if (Y==1 or Y==2);
   outcome = (Y==1);
   model = regset XBeta1 + res(draw=1, ref=u1)
         - regset XBeta2 - res(draw=1, ref=u2);
```

In other words, each multinomial probit outcome with three potential values contributes two correlated probit modules to the likelihood. The likelihood function involves a 2-by-2 covariance matrix of probit residuals. (Similarly, each multinomial probit outcome with four potential values contributes three correlated probit modules; the likelihood function involves a 3-by-3 covariance matrix.)

Now consider a multilevel extension. The above logic carries through, but the likelihood function now involves a covariance matrix of dimension $2T$-by-$2T$, where T is the number of replications (outcomes) in an observation. aML only knows how to compute cumulative normal probabilities of matrices up to 3-by-3, so this spells trouble even for T=2. Fortunately, without additional residuals (heterogeneity), the $2T$-by-$2T$ matrix is block-diagonal with T blocks of 2-by-2 matrices along the diagonal. aML recognizes this block-diagonality and computes the cumulative normal probabilities one block at a time. Adding heterogeneity is not a problem, so long as it is integrated-out. If unobserved heterogeneity is specified as a non-integrated residual, however, the $2T$-by-$2T$ matrix is no longer block-diagonal and its cumulative normal probability cannot be computed. This results in an error message.

13.16. Starting Values

The end of the control file needs to contain initial parameter values for the iterative maximum likelihood search procedure. The syntax is as follows:

```
starting values;
name1   {T|F}   x1
name2   {T|F}   x2
name3   {T|F}   x3
...
;
```

where *name1*, et cetera, are user-selected coefficient names of up to eight characters and *x1*, et cetera, their starting values. Starting values are also referred to as initial values. The coefficient names may consist of any character except blanks, tabs, and semicolons, unless you enclose the name in single or double quotes. For coefficients on data variables in regressor sets, it is common to give coefficient names the same name as the data variables, but this need not be done. It often cannot be done, namely when you transform data variables on-the-fly. For example, a regressor set may contain "`school<12`", flagging a high school drop-out. You may give its coefficient any name, including "`dropout`" and "`educ<HS`".

The order of starting values is determined by the order in which sets of parameters are defined, as described in the sections that document building block definitions.

The starting values *x1*, et cetera, must be real numbers. There is one exception: in the definition of vectors that make up a finite mixture distribution, you may let aML determine starting values. In that case, and that case alone, the starting value may be stated as "`auto`" (without the double quotes).

If the parameter is to be estimated, a "`T`" appears in between its name and starting value; if the parameter is fixed, an "`F`". (Think of these as "True" and "False" relative to the assumption that the parameter needs to be estimated.)

You may instruct aML to search sequentially over alternative sets of free parameters. The program uses optimized values of parameters from the previous round for the next round of estimation. This is done by typing multiple `T`'s and `F`'s between parameter names and their initial values. These `T`'s and `F`'s must not be separated by spaces or other characters. For example, if a parameter needs to be fixed in the first round, and freed up in the second and third rounds, the following needs to be specified:

```
name    FTT    x
```

For example, you may want to first estimate the intercept(s) only (fixing all other parameters), then free up regression parameters, and finally parameters related to residual distributions. The starting values may look as follows:

```
starting values;

Constant    TTT    0
X1          FTT    0
X2          FTT    0
Sigma       FFT    1
;
```

Multiple `T`'s and `F`'s can be very convenient if one is building up starting values for a model. As a general rule, it is a good strategy to build up such models by first estimating individual equations separately, then putting them together with (say) only intercepts and simultaneity parameters estimated, and as a third step freeing up all parameters. While it is always preferable to check the results of intermediary steps, it may save time to specify the step procedure in advance and let the program run through the sequence automatically.

The program will only execute subsequent rounds of estimation if the preceding round converged successfully. The same number of `T`/`F` values must be specified for each and every parameter.

13.17. Expressions

In most cases where aML's control file requires a variable name, you may instead specify an expression involving zero or more variables. With zero variables, you would specify a constant: "`1`" to denote an intercept in a regressor set, "`origin=0`" in a duration spline specification, et cetera. In outcome specifications, constants are not allowed, because aML would not know at what level (how often) the outcome needs to be evaluated. In other words, expressions that specify an outcome must involve one or more variables. If there is only one level in the data, no confusion is possible and no variable is required.

The following table lists the operators that aML supports.

`exp(x)`	exponent
`int(x)`	integer portion (truncation)
`log(x)`	natural logarithm
`abs(x)`	absolute value
`sqrt(x)`	square root
`spline(x, nodes)`	piecewise-linear spline transformation (see below)
`min(x,y[,...])`	minimum
`max(x,y[,...])`	maximum
`x^y`	x to the power y
`x*y`	x times y
`x/y`	x divided by y
`x+y`	x plus y
`x-y`	x minus y
`x==y`	x equals y (evaluates to 0 if false, to 1 if true)
`x<y`	x is less than y (evaluates to 0 if false, to 1 if true)
`x<=y`	x is less than or equal to y (evaluates to 0 if false, to 1 if true)
`x>y`	x is greater than y (evaluates to 0 if false, to 1 if true)
`x>=y`	x is greater or equal to than y (evaluates to 0 if false, to 1 if true)
`x!=y`	x is not equal to y (evaluates to 0 if false, to 1 if true)
`x and y`	Boolean and (evaluates to 1 if and only if both x and y equal 1)
`x or y`	Boolean or (evaluates to 1 if x, y, or both equal 1)
`not x`	Boolean not (evaluates to 1 if x equals 0)

In addition to `x!=y`, aML accepts `x~=y` to denote inequality.

With one exception, all expressions evaluate to a scalar. The exception is the spline transformation, which evaluates to an array, i.e., to multiple scalars. It only enters as part of a regressor set. For example, suppose you wish to estimate the effect of age on some outcome of interest, and allow for different slopes under age 20; between age 20 and 50; and over age 50:

```
spline(age, 20 50)
```

Formally, the spline transformation is given by:

$$\text{spline}(x,\ \nu_1\ \nu_2 \ldots \nu_n) = \begin{pmatrix} \min[x, \nu_1] \\ \max[0, \min[x-\nu_1, \nu_2-\nu_1]] \\ \vdots \\ \max[0, x-\nu_n] \end{pmatrix},$$

where $\nu_1\ \nu_2\ \ldots \nu_n$, denote the nodes. Note that each spline segment connects to the other at the node that separates them. In other words, the piecewise-linear spline transformation is a continuous function. Also see Panis (1994).

There is no limit to combining transformations and expressions. For example,

```
min(age^3-12, sqrt(income), exp((survey-birthdt)/365.25))
```

is perfectly fine, should that make sense.

Variables that appear in expressions need not be at the same data level. Their result is at the lowest level of any of the variables in the expression. For example, if `x` is a level 3 variable and `y` a level 4 variable, then `x*y` will behave as-if it is a level 4 variable.

14. aML Output

By default, aML writes output of its estimation to standard output and to an output file. The output file typically has extension "`.out`", but you may specify other output file names using option "`-o`" on the command line (page 264). This chapter helps interpret the output that aML produces.

The output may be distinguished into five sections. Section 14.1 documents general output; Section 14.2 discusses aML's feedback on the building blocks that you specified; Section 14.3 does the same on model specifications; Section 14.4 explains information on the search process; Section 14.5 describes the results of estimation; and Section 14.6, finally, discusses warnings and error messages.

Note that you may not see only part of the output illustrated below depending on the level of output information; see "`option screen info level`" and "`option file info level`". By default, these are set to 3 and 5, respectively. The output below is based on level 5, the highest level.

14.1. General

The first part of aML's output states the title of the run; displays license information; the date and time that the run started; the names of the control file and the data file; the data the data file was created; the version number of raw2aml that created it; convergence criteria; a frequency distribution of data structures in the data, and the numbers of outcomes that they generate; and information pertaining to weighted optimization. Most of the information is self-explanatory. For example:

```
 1  ============================================================================
 2  =                  This is the title of the run                            =
 3  ============================================================================
 4
 5                   +-----------------------------------+
 6                   |  aML version:    1.00             |
 7                   |  Serial number:  D1021020         |
 8                   |  Licensed to:    User Name        |
 9                   |                  Affiliation      |
10                   |  Academic license, single user    |
11                   +-----------------------------------+
12
13  Start of program:  Sun Jan  9 11:07:01 2000
14  Control file:      filename.aml
15  Input data file:   \projects\data\mydata.dat
16       Created on:   Sun Jan  9 10:58:57 2000
17       Created by:   raw2aml version 1.00
18
19  Converge if wgn < 1.0
```

```
20
21  Frequency distribution of data structures:
22   STRUCTURE |   Freq.  Percent
23  -----------+------------------
24         100 |   7202    11.96   (7202 outcomes)
25         200 |   4860     8.07   (4860 outcomes)
26         300 |  10243    17.02   (10243 outcomes)
27         400 |   9426    15.66   (9426 outcomes)
28         510 |  21326    35.43   (21326 outcomes)
29         520 |   7141    11.86   (7141 outcomes)
30  -----------+------------------
31       Total |  60198   100.00   (60198 outcomes)
32
33  Note: the number of observations is 5825, the number of level 2 branches
34  60198; they generated 60198 outcomes.
35
36  Observations are weighted by normalized variable 'nsampwt'; the sum of
37  weights is 5697.6167, the number of observations that are used in the
38  estimation 5825, so weights are scaled by 1.02236.
```

The frequency distribution of data structures (lines 21-31) not only shows how many level 2 branches there are in the data, but also how many outcomes each data structure generates. In the sample file, each level 2 branch generated one outcome. If some outcomes were at level 3 or lower, each level 2 branch would generate potentially multiple outcomes.

Lines 36-38 provide information on weighted optimization. In this case, the control file contained "`option normweight=nsampwt`", where `nsampwt` is a level 1 variable. As indicated by the output, its average value is slightly less than one. It is used as a normalized weight, so aML inflates each weight by 1.02236 such that the sum of weights is equal to the number of observations.

14.2. Building Block Definitions and Summary Statistics

Next, aML repeats definitions of building blocks. It also writes out the name(s) that you assigned to their coefficients, which may help you determine whether you lined up the starting values in the right order. aML also provides some relevant summary statistics.

14.2.1. Parameters, Vectors, and Matrices

Suppose you defined and initialized a vector as follows:

```
define vector VectorName;  ref = 10 20 30;
   dim=4;
   range=(0,1);
   increasing=no;

<...>

starting values;
Vec(1)   T   .2
Vec(2)   T   .4
Vec(3)   T   .6
Vec(4)   T   .8
<...
;
```

This definition would be repeated as follows:

```
define vector VectorName;  /* coefficient names 'Vec(1)'-'Vec(4)' */
   ref=10 20 30;
   dim=4;
   increasing=no;
   range=(0,1);
```

Make sure that the definition is repeated correctly. Carefully check that the names that you assigned to the vector's coefficients, `Vec(1)` through `Vec(4)`, are matched correctly in the "comment" that aML wrote out. This helps protect against misalignment of starting values. Definitions and coefficient names of parameters and matrices are repeated similarly.

14.2.2. Regressor Sets

aML repeats regressor set definitions in the output file and provides summary statistics on its regressors. For example:

```
define regressor set Reg_u;
   var = (nummar>1) (2<=parity<=10) (inschool==2 or inschool==3)
         (inschool==0)*(educyr<12) (inschool==0)*(educyr>12)
         (inschool==0)*(educyr==12) dadlths dadgths momlths momgths
         (fam14g!=1);

   ---------+-----------------  Summary statistics  ----------------
   name     |       #         Mean     Std Dev         Min          Max
   ---------+-----------------------------------------------------------
   PrevMar  |  84768    .0150411    .1217169         0.0          1.0
   Parity2+ |  84768    0.259119    .4381536         0.0          1.0
   Inschool |  84768    .1914166    .3934185         0.0          1.0
   Educ<hs  |  84768    0.097525    .2966731         0.0          1.0
   Educ=hs  |  84768    .1570404    0.363841         0.0          1.0
   Educ>hs  |  84768    .0326302    .1776679         0.0          1.0
   dad<hs   |  84768    .4195195    .4538405         0.0          1.0
   dad>hs   |  84768    .2413783    .3956354         0.0          1.0
   mom<hs   |  84768     0.44169    .4827995         0.0          1.0
   mom>hs   |  84768    .1761318    .3711942         0.0          1.0
   famnot1  |  84768    .3487637    .4765818         0.0          1.0
```

All transformations in the regressor set are repeated. The summary statistics list coefficient names; this helps protect against misalignment of starting values. They also list the number of times variables enter any model, their means, standard deviations, minimum and maximum values. The means and standard deviations are computed unweighted, even if optimization is weighted.

14.2.3. Splines

aML repeats spline definitions in the output file and provides summary statistics on the (origin) variables to which they are applied. For example:

```
define spline AgePattern;
   intercept;     /* intercept coefficient Constant */
   nodes= 16 20 25;  /* slope coefficients age0-16 - age25+ */

   ---------+-----------------  Summary statistics  ----------------
            |       #         Mean     Std Dev         Min          Max
   ---------+-----------------------------------------------------------
   origin   |    150    12.35693    19.54654         0.0       80.835
```

This spline was defined with an intercept and three nodes, i.e., with a total of five coefficients. aML states that the names of these coefficients, in the starting values, are "`Constant`" (for the intercept) and "`age0-16`" through "`age25+`" (for the four slopes); this helps protect against misalignment of starting values.

It also provides summary statistics of "`origin`". This is not necessarily a variable name. It summarizes all origin variables (where the spline is used as a duration spline in hazard models) and to-be-transformed variables (where the spline is used as a regressor spline in any model).

14.2.4. Distributions

aML supports four types of distributions: normal, ARMA(p,q), CAR(1), and finite mixture. It reproduces the definition in the output and indicates which coefficient names match the definitions. For example, for a four-variate normal distribution, output may be:

```
define normal distribution;  dim=4;
name=u1;
name=u2;
name=u3;
name=u4;
search=cholesky;
/* NOTE: no restrictions are placed on standard deviations or    */
/* correlation coefficients.  The following matrix shows names   */
/* of standard deviations (on the diagonal) and correlation      */
/* coefficients (off the diagonal) in the starting values.       */
/*    sigma_u1                                                   */
/*    rho_u2u1  sigma_u2                                         */
/*    rho_u3u1  rho_u3u2  sigma_u3                               */
/*    rho_u4u1  rho_u4u2  rho_u4u3  sigma_u4                     */
```

Carefully study the note in comment-style as it tells the names of coefficients in starting values that form the correlation matrix. No restrictions were placed on standard deviations and correlations of this distribution. The order of starting values gets a bit trickier when there are restrictions. Consider the following definition:

```
define normal distribution; dim=4;
   name=eps1;
   name=eps2;
   name=eps3;
   name=eps4;
   restrictions sigma(1)=sigma(2)=sigma(3)=sigma(4)
                rho(1,2)=rho(2,3)=rho(3,4)
                rho(1,3)=rho(2,4);
<...>
starting values;

sigma    T    1
rho1     T    .6
rho2     T    .4
rho3     T    .2
;
```

Notice that this distribution's covariance matrix is restricted to be band-diagonal. The output contains the following:

```
define normal distribution;  dim=4;
   name=eps1;
   name=eps2;
   name=eps3;
   name=eps4;
   restriction sigma(1)=sigma(2);
   restriction sigma(1)=sigma(3);
   restriction sigma(1)=sigma(4);
   restriction rho(2,1)=rho(3,2);
   restriction rho(3,1)=rho(4,2);
   restriction rho(2,1)=rho(4,3);
   /* NOTE: restrictions are placed on standard deviations or       */
   /* or correlation coefficients.  The following matrix shows the  */
   /* names of standard deviations (on the diagonal) and            */
   /* correlation coefficients (off the diagonal) in the starting   */
   /* values.  Please check this matrix carefully:                  */
   /*    sigma                                                      */
   /*    rho1      sigma                                            */
   /*    rho2      rho1      sigma                                  */
   /*    rho3      rho2      rho1      sigma                        */
```

aML writes the restrictions pairwise, but equivalently. It is very important that you carefully check that your starting values correspond to the standard deviations and correlations that you intend them to be.

Similarly, aML repeats the definitions of finite mixture, ARMA, and CAR(1) distributions, including the starting value coefficients that correspond to their parameters.

14.3. Model Specifications and Summary Statistics

The output repeats each model specification, followed by summary statistics of the outcome(s) and of other important variables. For example:

```
continuous model;
   data structure = 2000;
   keep if 18<=age<=70;
   outcome = wage;
   model = regset continuous +
      intres(draw=1, ref=d1) +
      res(draw=match, ref=u1)
      ;

   Summary statistics of the outcome and selected variables:

            |      #         Mean     Std Dev          Min          Max
   ---------+-------------------------------------------------------------
   outcome  |   4417     127.505    28.98207    34.31187    197.4787


      match     |       Freq.     Percent
   -------------+------------------------
               1 |        724       16.39
               2 |        722       16.35
               3 |        722       16.35
               4 |        727       16.46
               5 |        751       17.00
               6 |        771       17.46
   -------------+------------------------
         Total   |       4417      100.00

            |      #         Mean     Std Dev          Min          Max
   ---------+-------------------------------------------------------------
   age      |   4417    48.37831    12.08324           15           70
```

For continuous outcomes, aML writes out summary statistics of the outcome and of any variable that was used in keep/drop, draw, or indirect referencing expressions. If the number of distinct values is fewer than or equal to the maximum number of frequency categories, aML presents a frequency table; otherwise, a table with mean, standard deviation, minimum, and maximum values. Also see "`option maximum number of frequency categories`" on page 279; the default is 20 categories.

An example of output of an ordered probit model:

```
ordered probit model;
   threshold vars = T1m (-Inf=-99999) T2m (Inf=99999);
   model = regset X +
      intres(draw=1, ref=eps) +
      res(draw=_iid, ref=N(0,1))
      ;
```

```
   Summary statistics of the outcome and selected variables:

            |      #         Mean     Std Dev         Min          Max
   ---------+-----------------------------------------------------------
   outcome1 |   1000    -63199.6    48249.22      -99999     1.987932
   outcome2 |   1000     14800.5    35527.12    .0011356       99999
```

Note that aML includes "`res(draw=_iid, ref=N(0,1))`" in the model specifications, even though this was not specified in the control file (not shown). It serves as a reminder of aML's default behavior to insert an *iid* standard normal residual into (ordered) probit models, unless you explicitly specify a non-integrated residual.

Also note that summary statistics of the outcome (threshold) variables are labeled "`outcome1`" and "`outcome2`", even though the names of the data variables are "`T1m`" and "`T2m`". The reason is that threshold variables may be expressions, and you should be mostly interested in the outcomes themselves.

An example of a hazard model:

```
hazard model;
   keep if marnum==1;
   censor = censor;
   duration = lower upper;
   timemarks = time;
   model = durspline(origin=0, ref=FemaleAge) +
      durspline(origin=time1980, ref=FemaleTime) +
      regset FemaleGetmar
      ;

   Summary statistics of the outcome and selected variables:

   Hazard spell durations (for noncensored spells, lower and upper
   duration variables and their difference; for censored spells,
   spell duration):

       censor |      #         Mean     Std Dev         Min          Max
   -----------+-----------------------------------------------------------
     / lower  |  13610     22.00632    5.092896      12.961       76.964
   0 - upper  |  13610     22.08666    5.093879       13.04       77.046
     \ window |  13610     .0805665    .0022844    .0519981    .0830002
   1 - spell  |   1157     44.17905    14.42554      28.504       84.999


      marnum   |       Freq.     Percent
   ------------+------------------------
             1 |      14767      100.00
   ------------+------------------------
         Total |      14767      100.00
```

```
           |      #         Mean      Std Dev          Min          Max
---------+---------------------------------------------------------
time1980 |  14767    -36.9581    16.00037     -68.961     -14.045
```

First note the table with summary statistics of spell durations. For non-censored cases, summary statistics of the lower and upper bound of the event window are presented, as well as on their difference, i.e., the width of the window. For censored cases, aML just shows summary statistics of the spell duration.

Tabulations and summary statistics are furthermore presented of other key variables; here "`marnum`" and "`time1980`".

Finally, an example of a negative binomial model specification:

```
 41  negative binomial model;
 42     outcome = count;
 43     exposure = exposure;
 44     dispersion = par Alpha;
 45     incidence = exp(regset BetaX);
 46
 47     Summary statistics of the outcome and selected variables:
 48
 49               |      #         Mean      Std Dev          Min          Max
 50     ---------+---------------------------------------------------------
 51     outcome   |   1000     178.463    315.2044            0         3762
 52
 53               |      #         Mean      Std Dev          Min          Max
 54     ---------+---------------------------------------------------------
 55     exposure  |   1000    71.50072    28.28182            1           96
```

As before, the name of the outcome variable is "`counts`", but a frequency table or summary statistics are given under the "`outcome`" label.

14.4. Search Process

The next part of the output file provides feedback on the search process. For example (line numbers added):

```
119  Number of parameters in model:       15
120  Number of parameters estimated:      15
121
122  Starting values:
123       Name      Est?     Value
124     1  Constant   T    -22.450061
125     2  age0-16    T      1.210481
126     3  age16-20   T       .359682
127     4  age20-25   T      -.030738
128     5  age25+     T      -.075655
129     6  time       T      -.006536
130     7  Intercpt   T       .000000
131     8  black      T      -.512751
132     9  native     T       .169042
133    10  asian      T      -.229725
134    11  hispanic   T      -.226453
135    12  dropout    T       .088895
136    13  college    T      -.376332
137    14  perminc    T      -.025333
138    15  sigma      T       .600000    (must be strictly positive)
139
140  ================================================================================
141  =                       RESULTS OF OPTIMIZATION                                =
142  ================================================================================
143
144
145  ITERATION 1            LOG-LIKELIHOOD:  -76093.694498
146
147  PARAMETER        VALUE     GRADIENT   SEARCH DIR
148  Constant     -22.45006     493.9406    -.1661642
149  age0-16       1.210481     8001.586    -.0262199
150  age16-20      .3596821     3699.932      .153883
151  age20-25     -.0307377     5432.061     .2181848
152  age25+        -.075655     2711.092    -.0037534
153  time         -.0065356    -3401.718    -.0035644
154  Intercpt            .0     493.9406           .0
155  black         -.512751     75.02762    -.2231791
156  native        .1690424     6.275706     .0196938
157  asian        -.2297253    -21.36751    -.2668793
158  hispanic     -.2264531     75.08633     -.032642
159  dropout       .0888951     375.8768     .1567529
160  college      -.3763322    -233.0915     -.557282
161  perminc      -.0253332    -372.9549    -.0099529
162  sigma               .6    -1501.509     .8320498
163
164  SMALLEST EIGENVALUES:
165      -3.4E-12    3.219836    64.26053    117.9607    170.7055
166
```

```
167  REL PARAM CHG:  7.098269      WGTD GRAD NORM:   29.3267
168      GRAD NORM:  11368.06       REL LN-L IMPR:       N/A
169
170  SEARCHING TO IMPROVE FUNCTION VALUE (OLD LN-L =   -76093.694498):
171                    STEPSIZE:  1       NEW LN-L =   -76532.209608
172                    STEPSIZE:  1/2     NEW LN-L =   -75922.091156
173
174  ------------------------------------------------------------------------
175
176  ITERATION 2            LOG-LIKELIHOOD:  -75922.091156
177                         ABSOLUTE IMPROVEMENT:  171.603342
178
179  PARAMETER        VALUE     GRADIENT   SEARCH DIR
180  Constant     -22.53314     128.5995    -.0479446
181  age0-16       1.197371     2129.599     .0009946
182  age16-20      .4366236     1930.517     .0044825
183  age20-25      .0783547     3913.216     .0225655
184  age25+       -.0775317     1773.486    -.0150967
185  time         -.0083178     1842.724      .000258
186  Intercpt            .0     128.5995           .0
187  black        -.6243405     61.29071     .0883843
188  native        .1788893     6.082661     .0279626
189  asian         -.363165    -19.90757     -.080802
190  hispanic     -.2427741     32.28349     .0327518
191  dropout       .1672716      231.346     .0833646
192  college      -.6549732    -173.6912    -.2235022
193  perminc      -.0303096    -293.8676     -.004868
194  sigma         .9269465    -1691.306    -.1613431
195
196  SMALLEST EIGENVALUES:
197      -2.5E-13    3.181893    61.10427    90.34634    378.3696
198
199  REL PARAM CHG:   .498379      WGTD GRAD NORM:  20.83758
200      GRAD NORM:  5760.651       REL LN-L IMPR:  .0022552
201
202  SEARCHING TO IMPROVE FUNCTION VALUE (OLD LN-L =   -75922.091156):
203                    STEPSIZE:  1       NEW LN-L =   -75742.815607
204                    STEPSIZE:  2       NEW LN-L =   -75992.649674
205
206  ------------------------------------------------------------------------
207
208  ITERATION 3            LOG-LIKELIHOOD:  -75742.815607

et cetera...
```

Lines 119 and 120 report the total number of parameters in the model and the number over which the likelihood is maximized. In the example, these numbers are the same, but one often fixes one or more parameters to some constant value.

Lines 122-138 repeat the starting values that you specified. Note that sigma must be strictly positive. It is the standard deviation of a distribution (not shown), and aML automatically imposed a range restriction.

Subsequent lines show the iterative search process. For example, lines 147-162 show the current iteration's parameter values, their derivatives, and search direction. The latter two are in terms of the untransformed parameters, not in terms of internal transformations that ensure that parameters remain in legitimate ranges (such as sigma in the example).

Line 165 shows the five smallest eigenvalues of the Hessian matrix (matrix of second derivatives). (Strictly speaking, they are the opposites of eigenvalues, as eigenvalues of the Hessian are negative, except possibly for very small values due to numerical imprecision). These eigenvalues are extremely important!

Be sure to keep an eye on the smallest eigenvalues of the Hessian matrix as the maximum likelihood search process unfolds. Eigenvalues that are zero or close to zero are indicative of underidentification of your model.

In the example, we deliberately included a duration spline with an intercept and a regressor set with an intercept (definitions not shown). These intercepts are perfectly collinear, which shows up as a near-zero eigenvalue. (The reported smallest eigenvalue, -3.4E-12, is not quite zero because of numerical imprecision.) Fortunately, aML's search algorithm is pretty smart: the search direction on the second intercept is zero (line 154).

Line 167-168 report the current values of four potential convergence criteria. By default, convergence is achieved when the weighted gradient norm is less than 0.1 (page 276).

Line 171 shows the results of stepping out one times the search direction. The likelihood is worse, so the search continues at one-half the search direction. This likelihood is better, so aML accepts those new parameters and moves to the next iteration and recomputes the gradient, Hessian, and search direction (lines 176-194). Note that the new parameters result in a smaller weighted gradient norm. Lines 202-204 indicate that stepping out one times the search direction improves the likelihood, but twice the search direction does not.

You may be tempted to lower the level of output information and suppress much of the intermediate results during the search process. We recommend that you resist this temptation. The intermediate results include at least three important items:

- Eigenvalues of the Hessian provide very useful information on model identification. If they are close to zero from the onset, there is probably a multicollinearity in your model, for example because you specified too many intercepts. If the smallest eigenvalue moves toward zero during the search, chances are that some standard deviation became very small, or some correlation close to -1 or +1. This indicates that the residual is not identified, for example because you specified independent draws where the same draw should apply to multiple modules.
- The weighted gradient norm should steadily decrease throughout the search process. Nothing is necessarily wrong with your model if it increases from iteration to iteration,

but the search method is probably inefficient. It is not uncommon for the weighted gradient norm to increase when the parameter values step out two or more times the search direction. In that case, you may achieve faster convergence by limiting the number of steps the search is allowed to take at any iteration. For example, you may want to restrict "`option step range = -10 to 0`" (page 275) so that aML never steps out more than one ($=2^0$) times the search direction. Taking small steps, as small as 1/1024 ($=2^{-10}$) is rarely a problem.

- If parameters are close to their optimal value, the largest likelihood improvement tends to be achieved with a stepsize equal to one. If larger or smaller steps improve the likelihood more, the parameters may not be close to their optimum. (If the log-likelihood were perfectly parabolic, one step would jump straight to the maximum value, with just one iteration. The log-likelihood is not parabolic, but for most problems the approximation is fairly good once the search process reaches the top.) Non-unit stepsizes are not a problem, just an indication that the search has some way to go. In particular, for continuous models, it is very common to see large steps early in the search. (Also see "`option save step`" on page 276.)

14.5. Results of Estimation

The last part of the output file contains the results of estimation. For example (line numbers added):

```
427  ================================================================================
428  =                  ESTIMATION CONVERGED SUCCESSFULLY                           =
429  =                        RESULTS OF ESTIMATION                                 =
430  ================================================================================
431
432  Convergence based on:
433     Weighted gradient norm:          .0480711 < .1
434     Relative function improvement:  1.23E-06
435     Gradient norm:                  22.54625
436     Relative parameter change:       .0035144
437
438  ================================================================================
439
440  Log Likelihood:  -75720.0124
441                                              BHHH-based, non-corrected
442    Parameter    Free?     Estimate          Std Err        T-statistic
443
444     1  Constant    T    -22.591371052      .78449806968      -28.7972
445     2  age0-16     T      1.197949328      .04988006075       24.0166
446     3  age16-20    T      .43723030037      .01227691256       35.6140
447     4  age20-25    T      .12465562729      .01297718019        9.6058
448     5  age25+      T    -.09206057295      .00329222907      -27.9630
449     6  time        T    -.00764218713      .00069488763      -10.9977
450     7  Intercpt    T               .0                .0        ------
451     8  black       T    -.59885711048      .03589285179      -16.6846
452     9  native      T      .2113552647      .10338393444        2.0444
453    10  asian       T    -.43549499468      .09280273712       -4.6927
454    11  hispanic    T    -.25576485619      .04284996174       -5.9688
455    12  dropout     T     .25185460741      .02589089049        9.7275
456    13  college     T    -.74556603336      .04187967317      -17.8026
457    14  perminc     T     -.0361925048      .00897502062       -4.0326
458    15  sigma       T     .80455265366      .03772444049       21.3271
459
460  ================================================================================
461
462  Elapsed clock time is 170 seconds.
```

Line 428 states that the search process converged. Alternatively, you may see messages indicating that convergence was not achieved because the function could not be improved or because the number of iterations was insufficiently low. Despite non-convergence, aML will report parameter estimates, standard errors, et cetera, based on the last iteration. Use this information for subsequent attempts only. In particular, the standard errors are based on (an approximation to) the Hessian matrix, which is only valid at the optimum parameter values.

Lines 432-436 report the values of measures that may be used to establish convergence. In this case, as reported, convergence is based on the default criterion, namely a weighted gradient norm of less than 0.1. See "option converge" on page 276.

Lines 440-458 present the parameter estimates, their asymptotic standard errors, and t-statistics. Note the qualification of standard errors as "`BHHH-based, non-corrected`". Asymptotically, standard errors are related to the inverse of the Hessian matrix. By default, aML approximates this matrix as minus the sum over observations of the outerproduct of first derivatives. This "BHHH" approximation is itself based on asymptotic theory (Berndt, Hall, Hall, and Hausman, 1973). For smaller samples, we recommend that you specify "`option numerical standard errors`" (Section 13.1.5) to compute the Hessian matrix numerically. Another set of standard errors, so-called Huber-corrected or "robust" standard errors, may be computed with "`option huber`" (Section 13.1.6). If applicable, aML reports all three sets of standard errors.

Additional detail on the results of estimation may be obtained by specifying, for example, "option variance-covariance matrix" (for a variance-covariance matrix of parameter estimates; Section 13.1.7), "option correlation matrix (for the corresponding correlation matrix; Section 13.1.8), "option table format" (for easier-to-read table format; Section 13.1.10); and "option starting value format" (for output in the form of starting values in control files; Section 13.1.11). The latter is useful for updating control files with converged values, but this is more capably handled with `update`, a separate program that is bundled with aML; see Section 15.1. Furthermore, table formats are more capably created with bundled program `mktab`; see Section 15.2.

The last output line states how many clock-time seconds it took for the run to complete. The output reproduced here was created on a PC. On UNIX, it would distinguish between clock-time and CPU-time.

14.6. Error Messages and Warnings

aML's error messages are designed to provide helpful feedback on the nature of the problem. There are many hundreds different error messages; they should be self-explanatory and are not documented here. We would very much appreciate hearing from you if any error message was unclear or may otherwise be improved; please contact <support@applied-ml.com>.

Most warnings are also self-explanatory. We elaborate on a small subset.

***** WARNING: likelihood underflows to zero for ID = *id* *****

This warning may appear during the search procedure. It indicates that the likelihood for a particular observation was so small that it could not be distinguished from zero due to numerical limitations. This occurs when the current model parameters provide an extremely poor fit to the data of the observation.

You may see this message when the search process is stepping out many steps (number of search directions) because the likelihood keeps on improving. At some point, the parameters wander out in a very unlikely region, and underflows may result. This is no cause for concern; aML will realize that the current stepsize is too large, and back up.

If there are many underflows early on in the search process, your starting values are extremely poor. Start over and build up your model in small steps, such that the converged values of each run are good starting values for the next step. Even if there are only a few underflows early on in the search process, there is some cause for concern. Your starting values are apparently quite poor. aML may find its way to the maximum likelihood despite poor starting values. However, if the likelihood function is not globally concave, it may be unable to find the maximum. Indeed, it is not uncommon that likelihood functions for complex multiprocess, multilevel models are not concave. Try to build up your model in small steps, such that the converged values of each run are good starting values for the next step.

If there are observations whose likelihood underflows to zero in the final iteration, you should be very concerned. It indicates that even the final, perhaps converged, parameter values are a very poor fit to some of the data's observations. This should really not happen, and you should re-evaluate your choice of model.

WARNING: the following correlation matrix is not positive definite:

This warning may pop up during the search procedure, either because you initialized correlations of a trivariate or higher-dimensional distribution such that its covariance matrix is not positive definite (and thus illegitimate), or because aML's search process stepped into an illegitimate region. By default, aML will reduce all correlations

proportionally until the matrix is positive definite. You may turn that default behavior off by specifying "`option ensure positive definite=no`"; see page 280.

A more elegant solution that prevents non-positive definite covariance matrices is to specify "search=cholesky" as part of the distribution's definition (page 304). This option makes aML internally transform covariance matrices into Cholesky-decomposed parameters, and search on that transformation. The Cholesky search is only available when you did not specify any equality restrictions across standard deviations or correlations.

aML makes you initialize standard deviations and correlations, rather than variances and covariances. This ensures that univariate and bivariate distributions always have a positive definite covariance matrix. Only correlations of trivariate and higher-dimensional distributions may wander into illegitimate territory.

`WARNING: one or more standard deviations are (near) zero:`

This warning is typically the result of a search process that tried to push a standard deviation toward zero.

It may be the case that the standard deviation is significantly different from zero, and that aML's search algorithm went the wrong direction. This sometimes occurs when you initialize a standard deviation at a small value, perhaps because that specification is close to a previous run without the residual. We therefore recommend that you always initialize standard deviations well above zero. (In the order of 0.6 for hazard and categorical outcome models; at about the same level as transitory residuals in continuous outcome models.)

It may also be the case that the standard deviation truly is zero. If you intend to capture unobserved heterogeneity with this residual, and you have good reasons to believe that there really should be heterogeneity, then check that your draw variables are in order. With rare exceptions, identification of heterogeneity requires that the same draw enters in multiple equations. If instead you drew heterogeneity residuals independently across repeated observations, heterogeneity cannot be distinguished from transitory variation, and the standard deviation will go to zero.

15. Auxiliary Utility Programs

The aML package consists of six executable files: `aml`, `raw2aml`, `update`, `mktab`, `amltest`, and `points`. The latter four are utilities that are not central to model estimation, but they improve the efficiency of using aML. This chapter documents each utility in turn.

15.1. update

The update utility updates starting values in a control file with converged parameter estimates in the corresponding output file. The syntax is:

```
update [-f] file.aml [file.aml...file.aml]
```

where "`file.aml`" is an aML control file. If no extension is specified, update assumes a "`.aml`" extension. You may specify and update multiple control files at once.

The control file will only be updated if the output file indicates that the estimation process converged. Option "`-f`" forces an update of starting values, even if no convergence was achieved.

In the aML control file, "`option starting value format`" generates a listing of parameter estimates in starting value format (page 273). Update does not require that the output file contain this format. In fact, it was written to make updating starting values more efficient than the otherwise required copy-and-paste action.

15.2. mktab

The mktab utility reads one or more output files and writes out parameter estimates and standard errors (or t-statistics) in easy-to-read table-format. The syntax is:

```
mktab [-option] file.out [file.out...[+]...file.out]
```

where "*`file.out`*" is the name of an aML output file that contains estimation results. If no extension is specified, mktab assumes extension "`.out`".

You may specify multiple output files and tabulate the results in multiple columns. The results of two or more output files may be tabulated in a single column by specifying a "+" between the file names. For example, suppose you estimate a two-equation model by first estimating the two separate equations in two control files (`eqn1.aml` and `eqn2.aml`, say) and then put them together with a correlation coefficient (and/or other parameters which account for simultaneity across the equations) in a third control file (`eqn3.aml`, say). The results may be tabulated and compared by

```
mktab eqn1 + eqn2 eqn3
```

or:

```
mktab eqn3 eqn1 + eqn2
```

The order in which output files are listed determines the order in which parameters are listed along the rows. The output tends to look best when you list the most comprehensive specification first, so that, for example, standard deviations and correlations of a distribution are listed together. Duplicate parameter names are listed in the order in which they are encountered. If there are duplicates, mktab issues a warning.

The following optional switches are supported:

`-c` places commas between columns and eliminates spaces. This option is convenient for importing the table into spreadsheet and word processor packages. Direct mktab's output into a file (`mktab -c file.out > temp`). To insert into a spreadsheet package, directly import the file (`temp`) and instruct the package that fields are delimited by commas. To insert into a word processor document, open the file (`temp`) with any text editor, copy its contents, paste it into a word processor document, and convert the text into a table, where text fields are separated by commas. To neatly line up all parameter estimates and standard errors or t-statistics, insert a tab mark that aligns decimal points in every cell.

`-s` uses strict criteria for significance asterisks. By default, mktab places one asterisk ('`*`') next to parameters that are asymptotically significant at 10 percent, two ('`**`') for 5 percent, and three ('`***`') for 1 percent significance. Option "-s" instead uses '`*`' at 5 percent, '`**`' at 1 percent, and '`***`' at 0.1 percent.

`-t` lists absolute t-statistics in parentheses, instead of the standard errors are that are listed by default. May not be combined with `-p`.

`-p` lists p-values in parentheses, instead of the standard errors are that are listed by default. May not be combined with `-t`.

`-ns` suppresses standard errors (but not significance asterisks);

`-NS` suppresses standard errors and significance asterisks;

`-`*n* specifies the number of digits after the decimal point with which parameter estimates and standard errors (or t-statistics or p-values) are written out. The default is 4.

Options need to be specified separately. For example, options `-c` and `-t` may not be combined into `-ct`.

15.3. amltest

The amltest utility is a convenient way to conduct a likelihood ratio test on nested specifications. The syntax is:

```
amltest file1.out file2.out
```

where the two arguments are names of aML output files. If no extension is specified, extension "`.out`" is assumed.

The output files should contain estimates of two nested models. The order in which the two output files are specified does not matter, i.e., it does not matter whether the first model is nested in the second or vice versa. The likelihood ratio test provides a joint significance test of the parameters that are fixed in the nested model. Suppose the nested model has `n1` parameters over which the likelihood was maximized and the nesting model has `n2` free parameters (`n1<n2`). Their respective log-likelihood are `lnl1` and `lnl2`, say. Twice the difference in their log-likelihoods is distributed as a Chi-square distribution with `n2-n1` degrees of freedom. Amltest computes the probability that the chi-square distribution exceeds twice the difference in the output files, i.e., it computes the level of significance at which the `n2-n1` additional parameters are jointly different from zero.

There is no check that the models are indeed nested.

15.4. points

The points utility computes Gauss-Hermite support points and weights. The syntax is:

```
points n
```

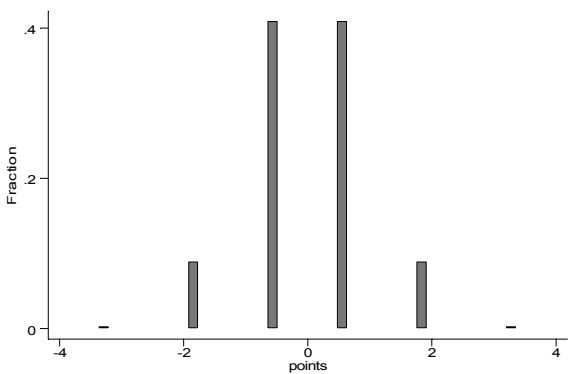
aML uses Gauss-Hermite quadrature to compute the support points and weights that approximate normal distributions that are integrated-out (see Section 13.2.6, in particular page 305 and following). The points utility computes those points and weights for *n* number of integration points. For example:

```
points 6
```

produces the following output:

```
          points       weights
1   -3.324257E+00  2.555784E-03
2   -1.889176E+00  8.861575E-02
3   -6.167066E-01  4.088285E-01
4    6.167066E-01  4.088285E-01
5    1.889176E+00  8.861575E-02
6    3.324257E+00  2.555784E-03
```

The graph on the right visualizes the six-point Gauss-Hermite approximation to the standard normal distribution. (The first and sixth points are barely visible because their weights are only 0.002555784.) For normal distributions with standard



deviation σ, the points are scaled by σ. For *k*-variate distributions, each *k*-vector of 6^k combinations of points is premultiplied by the Cholesky decomposition of the covariance matrix.

16. Miscellaneous Features

16.1. Control File Comments

Raw2aml and aML control files may include user comments. These comments must begin with a slash-asterisk combination (`/*`) and terminate with a asterisk-slash (`*/`). Comments may be nested.

Comments may not be inserted in strings. For example, the following title includes every character that is typed inside the double quotes, including the word "`contains`":

```
option title = "This title /* contains */ a comment-like string";
```

Similarly, if you would like to include a slash-asterisk combination in a coefficient name, you may do so by enclosing the entire name in double quotes:

```
starting values;
"/*name*/"   T    0
<et cetera>
;
```

Raw2aml and aML preprocess their control files and strip out comments before parsing the statements. By default, the stripped-down version of the control file is removed after parsing is done, but you may save it permanently to disk by using the "`-m`" option that both raw2aml and aML support. See pages 222 and 264.

16.2. Macro Language

Raw2aml and aML support a simple macro language. It may be used to avoid repetition of control file statements. A macro must be defined before it may be used. The definition syntax is:

```
%macro macroname;
   <statements or other text>
%mend;
```

where the "`macroname`" may be up to twelve characters in length. It may contain any printable character except blanks and semicolons. It may not contain any arguments. Macros may be invoked as:

```
%macroname
```

Note that this invocation does not terminate with a semicolon. It need not even terminate with a blank space or line feed (carriage return) character. You may therefore use macros for portions of names or strings. For example:

```
%macro sex;male%mend;
option title = "This model refers to %sexs only";
```

The title resolves to "`This model refers to males only`". This strict resolution implies that the first characters of the name of a macro may not be equal to the full name of another macro. For example, if you define two macros by the names of "`temp`" and "`temp1`", you would not be able to use "`temp1`", because "`%temp1`" evaluates to the definition of "`temp`", followed by the number "`1`".

Macros may be nested. For example:

```
%MACRO covariates;
   thisvar
   thatvar
   %race
%MEND;
%MACRO race;  (race==1)  %MEND;
```

macro "`%covariates`" now expands to:

```
   thisvar
   thatvar
     (race==1)
```

Raw2aml and aML preprocess their control files and resolve macros before parsing the statements. By default, the version with resolved macros is removed after parsing is done, but you may save it permanently to disk by using the "`-m`" option that both raw2aml and aML support (pages 222 and 264, respectively). This can be quite helpful, especially for debugging nested macros.

Glossary

Branch — Multilevel data may be thought of as consisting of a top level (level 1) with zero or more level 2 branches, zero or more level 3 subbranches, zero or more level 4 subbranches, et cetera. A branch is thus a data unit at a specific level. We try to be consistent and use "branch" for level 2 units and "subbranch" for levels 3 and lower. Also see Section 3.1.1.

Building block — Anything that may appear on the right-hand-side of model equations. The most common building blocks are regressor sets, distributions with residuals, and (for hazard models) splines that make up the baseline hazard. Other building blocks include parameters, vectors, and matrices. aML requires that a building block is first defined and then used to specify (potentially multiple) models. Its definition introduces parameters. By defining a building block only once, the same parameters may be used in multiple model statements, thereby restricting their values across models. See Section 13.2 for building block definitions.

Control variable — Variable in the data set that is used for control purposes. For example, variables specifying a data structure number, an observation ID, and a number of subbranches are control variables. Control variables are typically created by the researcher, rather than a response of a survey subject.

Data structure — A data structure is a subset of variables in the data. It contains both outcome variables and explanatory variables. For a number of reasons it may be convenient or necessary to create subsets of variables into distinct data structures. This is almost always only applicable to multiprocess models in which covariates of one process are not readily placed in the same records as covariates of another process. Data structures are optional and tend to be used by advanced users only. Also see page 319.

Data variable — Any variable in the data set that is not a control variable. Data variables may be dependent or independent variables in statistical models. They tend to be survey responses or transformations of survey responses. Data variables are always listed as part of the *varlist* in a raw2aml control file: "`level n var = varlist;`".

Draw — By their very nature, residuals are unobserved to the analyst. However, each residual has a certain value in each realization. Each realization is the result of a draw. In multilevel models, there is only one realized value of the residual corresponding to the highest (most aggregated) level, one draw. At lower levels, there will be more than one draw. In model specifications, residuals are specified with a "`draw=expression`"; all outcomes with the same residual

draw are correlated. Draws are only unique within distribution, i.e., the same draw values for residuals from different distributions do not result in correlation. Residuals are correlated if and only if they belong to the same distribution and they have the same draw.

Hessian matrix — Matrix of second derivatives of the log-likelihood with respect to the model parameters. This matrix plays an important role in the maximum likelihood search algorithm and the computation of standard errors. By default, it is approximated as minus the sum over all observations of the outerproduct of first derivatives (Berndt, Hall, Hall, Hausman, 1974). Also see "`option numerical search`" (Section 13.1.4) and "`option numerical standard errors`" (Section 13.1.5).

Observation — A set of observed outcomes and explanatory variables that is statistically independent from other sets. Two annual wage records of the same person are statistically dependent, need to be modeled jointly, and are thus part of the same observation. Wage records of another person (who is not related to the first person) are independent from those of the first person, and thus part of another observation. An observation in aML always corresponds to the highest level, the most aggregated level, level 1. There may be multiple ASCII records, possibly in multiple ASCII files, pertaining to one observation. Raw2aml and aML will know that they are part of the same observation because they all have the same ID.

Parameter — In generic terms, a parameter is an unknown scalar that is part of one or more model equations. In that interpretation, any regression coefficient, standard deviation, et cetera, is a parameter. Specifically, aML recognizes a parameter building block (Section 13.2.1). It is a scalar parameter which, once defined, you may use in any model specification. For example, you may interact other model elements with parameters.

Regressor set — A regressor set is a vector of variables, say X, forming a regression equation, $\beta'X$, for which coefficient vector β is to be estimated (or assigned a fixed value). The variables are typically just variables in the data set, but may be formed as transformations or interactions of variables. The resulting regression equation may be used in any model. Think of a regressor set as a $\beta'X$, but realize that X may be an expression involving zero or more variables.

References

Abramowitz, M. and I.A. Stegun. 1972. *Handbook of mathematical functions*, Dover Publications, Inc., New York.

Berndt, E.R., B.H. Hall, R.E. Hall, and J.A. Hausman. 1974. "Estimation and Inference in Nonlinear Structural Models." *Annals of Economic and Social Measurement* 3: 653-665.

De Leeuw, Jan and Ita Kreft. 1986. "Random Coefficient Models." *Journal of Educational Statistics* 11(1): 55-85.

Greene, William H. 2000. *Econometric Analysis* (fourth edition). Prentice Hall, Upper Saddle River, New Jersey.

Heckman, James J. 1979. "Sample Selection Bias as a Specification Error." *Econometrica* 47: 153-161.

Heckman, James and Burt Singer. 1984. "A Method for Minimizing the Impact of Distributional Assumptions in Econometric Models for Duration Data." *Econometrica* 52(2): 271-320.

Hildreth, C. and J. Houck. 1968. "Some Estimators for a Linear Model with Random Coefficients." *Journal of the American Statistical Association* 67: 584-595.

Hox, J. 1995. *Applied Multilevel Analysis*. Amsterdam: TT-Publikaties.

Huber, P.J. 1967. "The Behavior of Maximum Likelihood Estimates under Non-standard Conditions." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability.* Berkeley, California: University of California Press, Vol. 1, 221-233.

Johnson, Norman L. and Samuel Kotz. 1970. *Distributions in Statistics: Continuous Univariate Distributions.* (Volume 2). John Wiley and Sons, New York.

Johnson, Norman L., Samuel Kotz, and Adrienne W. Kemp. 1992. *Univariate Discrete Distributions.* (Second edition). John Wiley and Sons, New York.

Judge, George D., R. Carter Hill, William E. Griffiths, Helmut Lütkepohl, and Tsoung-Chao Lee. 1988. *Introduction to the Theory and Practice of Econometrics.* John Wiley and Sons, New York.

Judge, George D., William E. Griffiths, R. Carter Hill, Helmut Lütkepohl, and Tsoung-Chao Lee. 1985. *The Theory and Practice of Econometrics.* John Wiley and Sons, New York.

Kalbfleish, John D. and Ross L. Prentice. 1980. *The Statistical Analysis of Failure Time Data.* John Wiley and Sons, New York.

Lillard, Lee A. 1993. "Simultaneous Equations for Hazards: Marriage Duration and Fertility Timing." *Journal of Econometrics* 56: 189-217.

Lillard, Lee A, Constantijn W.A. Panis, and Dawn M. Upchurch. 1996. "Interdependencies over the Life Course: Women's Fertility, Marital, and Educational Experiences." Mimeo, RAND.

Longford, N.T. 1993. *Random Coefficient Models.* Oxford: Clarendon Press.

McCullagh, P. and J. Nelder. 1983. *Generalized Linear Models*. Chapman and Hall, New York.

Panis, Constantijn and Lee Lillard. 1993. "Timing of Child Replacement Effects on Fertility in Malaysia." Mimeo, RAND.

Panis, Constantijn W.A. 1994. "The Piecewise Linear Spline Transformation with an Application to Age at First Marriage," *Stata Technical Bulletin* 18: 27-29.

Pascal, Blaise. 1679. *Varia opera Mathematica D. Petri de Fermat*: Tolossae.

Pindyck, Robert S. and Daniel L. Rubinfeld. 1991. *Econometric Models and Economic Forecasts*. McGraw-Hill, Inc, New York, NY.

Poisson, Siméon-Denis. 1837. *Recherches sur la Probabilité des Jugements en Matière Criminelle et en Matière Civile, Précédées des Regles Générales du Calcul des Probabilités*. Paris: Bachelier, Imprimeur-Libraire pour les Mathematiques, la Physique, etc.

Tobin, James. 1958. "Estimation of Relationships for Limited Dependent Variables." *Econometrica* 26: 24-36

White, Hall. 1980. "A Heteroskedasticity-consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." *Econometrica* 48: 817-830.

White, Hall. 1982. "Maximum Likelihood Estimation of Misspecified Models." *Econometrica* 50: 1-25.

Index