STATA
TECHNICAL
BULLETIN

January 2000
STB-53

A publication to promote communication among Stata users

Subscriptions are available from Stata Corporation, email stata@stata.com, telephone 979-696-4600 or 800-STATAPC, fax 979-696-4601. Current subscription prices are posted at www.stata.com/bookstore/stb.html.

Previous Issues are available individually from StataCorp. See www.stata.com/bookstore/stbj.html for details.

Submissions to the STB, including submissions to the supporting files (programs, datasets, and help files), are on a nonexclusive, free-use basis. In particular, the author grants to StataCorp the nonexclusive right to copyright and distribute the material in accordance with the Copyright Statement below. The author also grants to StataCorp the right to freely use the ideas, including communication of the ideas to other parties, even if the material is never published in the STB. Submissions should be addressed to the Editor. Submission guidelines can be obtained from either the editor or StataCorp.

The *Stata Technical Bulletin* (ISSN 1097-8879) is published six times per year by Stata Corporation. Stata is a registered trademark of Stata Corporation.

Contents of this issue

		page
an71.	Spring NetCourse schedule announced	2
dm63.1.	A new version of winshow for Stata 6	3
dm75.	Safe and easy matched merging	6
sg35.2.	Robust tests for the equality of variances update to Stata 6	17
sg120.1.	Two new options added to rocfit command	18
sg124.	Interpreting logistic regression in all its forms	19
sg125.	Automatic estimation of interaction effects and their confidence intervals	29
sg126.	Two-parameter log-gamma and log-inverse Gaussian models	31
sg127.	Summary statistics for estimation sample	32
sg128.	Some programs for growth estimation in fisheries biology	35
sg129.	Generalized linear latent and mixed models	47

an71 Spring NetCourse schedule announced

NetCourse 101. An Introduction to Stata

6 weeks (4 lectures)	
Course dates:	February 25 through April 7
Deadline for enrollment:	February 21
Cost:	$ 85
Course leaders:	Shana Carter, David Reichel, and Jeremy Wernow
Prerequisites:	Stata 6, installed and working
Schedule:	
Lecture 1	February 25
Lecture 2	March 3
One-week break	March 9 through March 15
Lecture 3	March 17
Lecture 4	March 24
Closing discussion	
Course ends	April 7

NetCourse 151. An Introduction to Stata Programming

6 weeks (4 lectures)	
Course dates:	April 7 through May 19
Deadline for enrollment:	April 3
Cost:	$100
Course leaders:	David Cowart, Roberto Gutierrez, and Ken Higbee
Prerequisites:	Stata 6, installed and working
	Basic knowledge of using Stata interactively
Schedule:	
Lecture 1	April 7
Lecture 2	April 14
One-week break	April 20 through April 26
Lecture 3	April 28
Lecture 4	May 5
Closing discussion	
Course ends	May 19

NetCourse 152. Advanced Stata programming

6 weeks (4 lectures)	
Course dates:	March 3 through April 14
Deadline for enrollment:	February 28
Cost:	$100
Course leaders:	David Drukker, Ken Higbee, and Allen McDowell
Prerequisites:	Stata 6, installed and working
	NetCourse 151 or equivalent knowledge
Schedule:	
Lecture 1	March 3
Lecture 2	March 10
One-week break	March 16 through March 22
Lecture 3	March 24
Lecture 4	March 31
Closing discussion	
Course ends	April 14

More information, including an outline of each course, can be obtained by pointing your browser to *http://www.stata.com*; clicking on the Headline *Spring NetCourse schedule announced*; and emailing *stata@stata.com* for enrollment forms.

dm63.1 A new version of winshow for Stata 6

Tony Brady, Imperial College School of Medicine, UK, t.brady@ic.ac.uk

The `winshow` command (Brady 1998) has been updated and improved for Stata 6. The major new features are the ability to display and enter data into a variety of controls (edit boxes, drop-down lists, check boxes and radio buttons), and the ability to search a dataset for particular observations.

Syntax

`winshow` [*varlist*] [`if` *exp*] [`in` *range*] [, `maxdisp(`*#*`)` `edit` `new` `del` `find` `strict`

`dateord(`*str*`)` `log(`*varlist*`)` `call(`*program_name*`)` `caption(`*text*`)` `header(`*var*`)`

`lspace(`*#*`)` `drop(`*#*`)` `nonum` `novar` `nodesc` `notype` `nopreserve`]

`winset` [*varlist*] [, `list` `do`]

New options

There are four new options in the new version of `winshow` (see Brady 1998 for a complete description of all options):

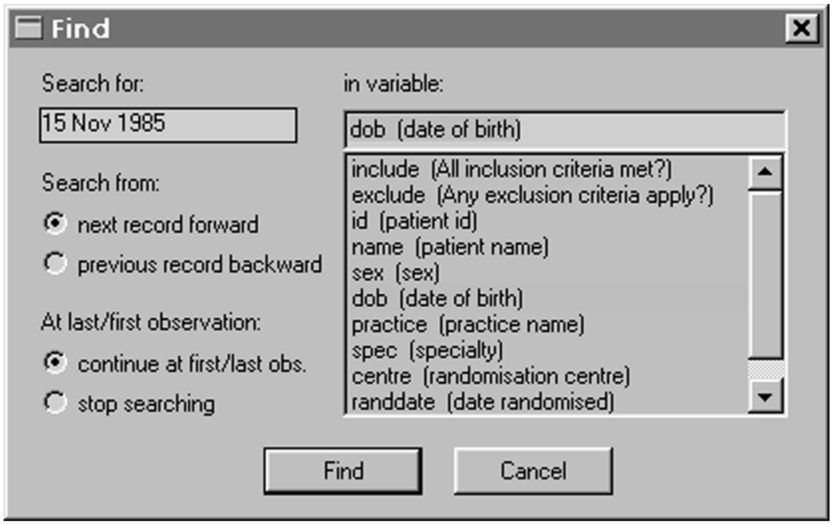
`find` adds a find button to the `winshow` dialog box allowing searches for a particular value in one of the variables (see Figure 1). By default, searches start at the next observation and go forward through the dataset, cycling around to the first observation and finishing at the current observation. Backward searches are allowed and the cycling round can be disabled if required. Dates can be entered naturally into the "search-for" field; they will be converted into elapsed dates before the search starts. Searching for text within a string variable will be deemed successful even if the search text is only part of the searched text and regardless of case. For example, searching for 'good' in a surname variable would find observations with surnames 'Goodman', 'MR GOOD', 'Holgoods', and so on.

`header(`*var*`)` adds a header to the dialog box displaying the current value of *var*. This is useful when an observation is spread over several pages.

`lspace(`*#*`)` determines the horizontal spacing between lines (the default is 11).

`drop(`*#*`)` defines the maximum depth of drop-down lists (the default is 60).

`do` in `winset` displays the `char` commands necessary for creating the current `winshow` settings for *varlist*. Programmers might find it useful to log these commands to a do-file.



Search for: 15 Nov 1985

in variable: dob (date of birth)

include (All inclusion criteria met?)
exclude (Any exclusion criteria apply?)
id (patient id)
name (patient name)
sex (sex)
dob (date of birth)
practice (practice name)
spec (specialty)
centre (randomisation centre)
randdate (date randomised)

Search from:
● next record forward
○ previous record backward

At last/first observation:
● continue at first/last obs.
○ stop searching

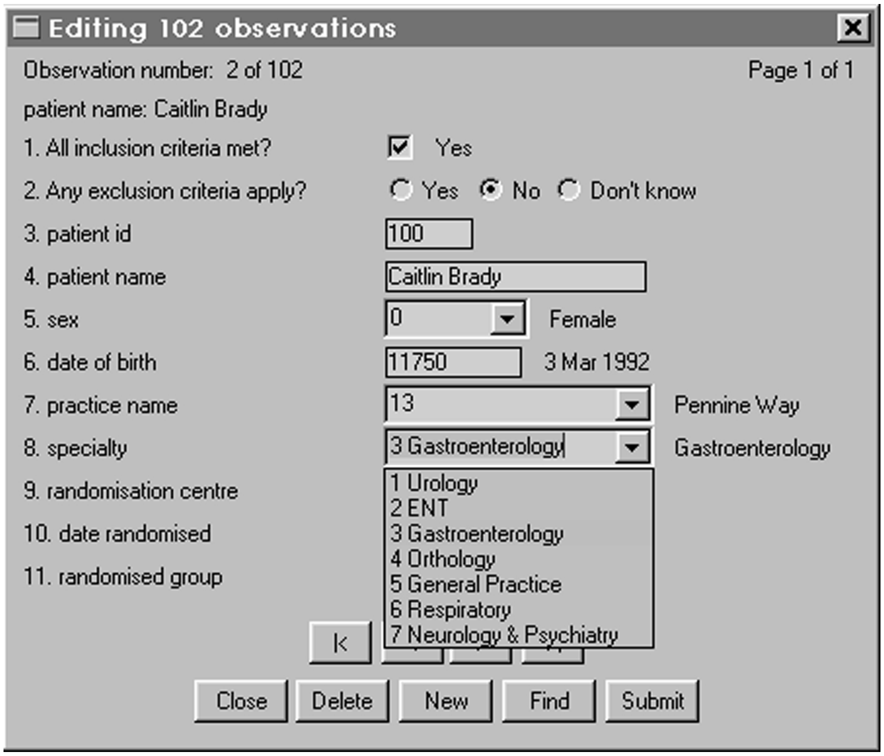
Figure 1. Example of search facility.

Example

The `winshow` dialog box shown in Figure 2 was generated with the command:

```
. winshow, edit new del find header(name) novar notype
```

The controls used to display information are determined by the `contype` characteristic of each variable. These can be set manually with the `char` command or through the `winset` interface (see below). Edit boxes can be used to display any type of variable but all other control types may only be used with labeled categorical variables. Radio buttons allow variables with two, three, or four categories defined by the first and one or more contiguous values in the set d$\{1, 2, 3, 4\}$. Check boxes only allow variables with two categories which must be coded 0 (unchecked) and 1 (checked). By their nature, radio buttons and check boxes do not allow missing values. There is no limit on the number of categories allowed with drop-down boxes, and missing values are allowed unless otherwise specified (the `specialty` variable shown in Figure 2 is an example of missing not allowed).



Editing 102 observations

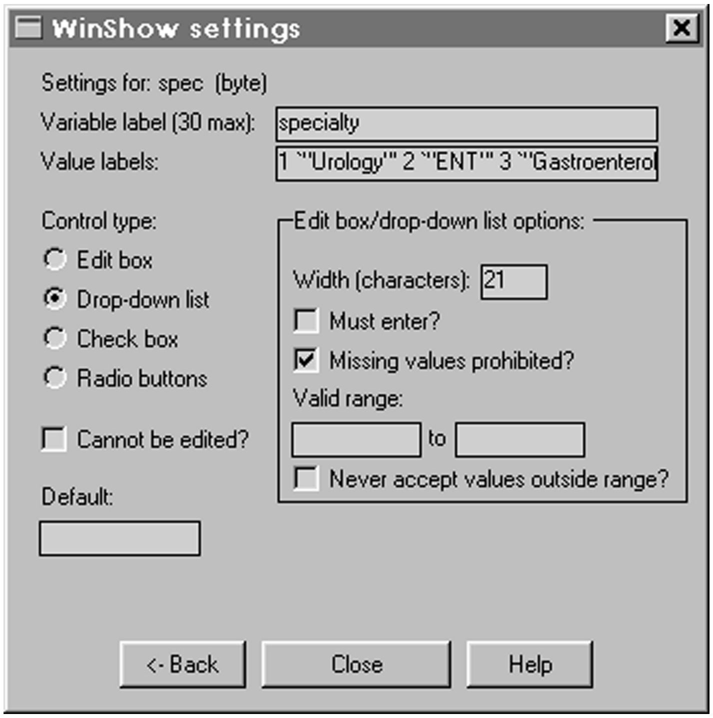
Observation number: 2 of 102 Page 1 of 1

patient name: Caitlin Brady

1. All inclusion criteria met? ☑ Yes
2. Any exclusion criteria apply? ○ Yes ● No ○ Don't know
3. patient id 100
4. patient name Caitlin Brady
5. sex 0 Female
6. date of birth 11750 3 Mar 1992
7. practice name 13 Pennine Way
8. specialty 3 Gastroenterology Gastroenterology
9. randomisation centre
10. date randomised
11. randomised group

Figure 2. Example of new control types and header.

`winshow` characteristics are used by `winshow` to determine how variables are to be displayed and dealt with during data entry. These characteristics are most easily manipulated with the `winset` program (see Figure 3).



Settings for: spec (byte)

Variable label (30 max): specialty

Value labels: 1 `"Urology"' 2 `"ENT"' 3 `"Gastroenterol

Control type:

○ Edit box

● Drop-down list

○ Check box

○ Radio buttons

□ Cannot be edited?

Default:

Edit box/drop-down list options:

Width (characters): 21

□ Must enter?

☑ Missing values prohibited?

Valid range: to

□ Never accept values outside range?

Figure 3. Reviewing the variable characteristics.

`winset` also allows the user to edit variable and value labels. For programmers, the relevant characteristics are

Characteristic	Value	Description
`contype`	#	Controltype: 1=edit box, 2=drop-down list, 3=check box, 4=radio buttons
`default`	# or *string* or “*todays date*”	Specifies a default value the variable is to take when entering a new observation. A special default value for date variables is allowed; namely, *todays date*. This allows observations to be date-stamped according to when they were entered.
`len`	#	The length in characters of the edit box used to display values of the variable. The default length is either the length of the maximum value of the variable (for numeric) or the space allocated to a string variable (e.g., 12 for `str12`).
`noedit`	#	A value of 1 prohibits editing of the variable.
`nomiss`	#	A value of 1 prevents the user from leaving the variable blank or entering missing (a stricter version of `req`).
`range`	# # [*strict*]	The valid range of input values is defined by the two numbers (lower and upper limit, respectively). Optionally, *strict* may be specified which prohibits entry of values outside the valid range (including missing).
`req`	#	A value of 1 means that the variable cannot be left blank (although missing is allowed).

Acknowledgment

This work was supported by the British Heart Foundation (grant number DSVC 3358).

References

Brady, T. 1998. dm63: Dialog box window for browsing, editing, and entering observations. *Stata Technical Bulletin* 46: 2–6. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 28–34.

dm75 Safe and easy matched merging

Jeroen Weesie, Utrecht University, Netherlands, j.weesie@fss.uu.nl

In this insert I describe the new command `mmerge` for “merging on matching variables” of two datasets. `mmerge` is both safer and easier to use than the powerful standard Stata command `merge`. The safety of `mmerge` is best explained in terms of an important property of matched variable(s): namely, whether or not it (they) form a key in the two datasets. A key comprises one or more variables that uniquely define observations and is never missing. Whether or not one or more variables form a key is primarily a theoretical issue; it depends on the properties of the objects represented by the data. If a dataset of individuals contains a person’s social security number (SSN), SSN provides a good example of a key. There cannot be two persons with the same social security number and it is never (should never be) missing. Of course, a dataset may violate these properties, but this means there are errors in the dataset that need to be fixed. `mmerge` helps identify such problems.

I want to stress that key-ness is a theoretical property of the data that is derived from one’s understanding of the data—it is not just an empirical issue. The claim that a (list of) variables forms a key, however, can be falsified on the data, e.g., it can be verified that the SSN of all respondents is not missing, and that no two persons have the same SSN. It is also possible that one or more variables uniquely identifies observations “by coincidence.” For instance, all persons in a dataset may have a different date of birth, but it is ill-advised to say that such a variable is a key. If observations are added, there now may be two persons with the same date of birth, and date of birth no longer may serve as a key, and `mmerge` would produce an error message.

Adopting the labels “master” and “using” for the data in memory and the data file to be match-merged to the master, we can clearly distinguish four types of match-merging.

type	master	using
1:1	key	key
n:1	—	key
1:n	key	—
n:n	—	—

The simplest case is when the match variables form a key in both the master and the using data. For example, the two datasets contain characteristics of a set of cars, identified by the make of the car. `make` will (should be) a key in both datasets. 1:1 matching on `make` creates a dataset on cars, with all characteristics of the cars found in the two datasets.

If the matching variables form a key in the master or in the using data, matched merging is sometimes called “table lookup” or “spreading.” Consider type n:1. Conceptually, the master data file contains information about units of type A (e.g., persons), that contain a property of type B (e.g., the city in which the person lives), while the using file contains properties of objects of type B (e.g., properties of cities). The type B dataset has a (simple or composite) key, KeyB (e.g., cityid), and the master contains a variable (e.g., city) that takes values in the set of values of KeyB. Match-merging means bringing properties of B into the dataset on A (properties of the city of which a person lives are “imported” into the data on persons). In my experience, table lookup is the most frequent application of matched merging.

Finally, in n:n matching, the match variables do not form a key in the master and using data. In this case, the standard Stata command `merge` performs a one-to-one merge between the observations of the master and using data that tie on the matching values, copying the last record “on the shorter side”. Thus, the *physical* order of the data within ties determines the result of the matched merge; a very undesirable and dangerous “feature” indeed. This implementation of matched merging when the matching variables do not form a key on either side is somewhat odd behavior; the main reason being that SAS, SPSS, and Statistica implement matched merging similarly. `mmerge` takes another approach, also followed by relational database systems, and also, for instance, by S-Plus; namely, a matched merge is defined as the set of all pairs of master and using observations that have the same values on the match variables. Note that this definition generalizes 1:1, n:1, and 1:n matched merging quite neatly. The resulting dataset is clearly independent of the order of observations of the master and using data.

This form of matched merging is not possible with the `merge` command. However, it is available in Stata, namely via the command `joinby`. `joinby` can also be used to perform 1:1, 1:n, and n:1 matched merging, but is less efficient than `mmerge`, and also lacks a substantial number of safety and convenience features offered by `mmerge`. For n:n matched merging, `mmerge` indeed invokes `joinby`.

It should be noted that a new version of `joinby` has become available as of September 28, 1999. The new `joinby` has additional features required by `mmerge`. The user should ensure they have a recent version of `joinby` installed.

Main features of mmerge

`mmerge` makes matched merging both easier and safer than using `merge` directly. This is due to the features described below.

- mmerge verifies whether the match variables are keys in the two datasets as specified by the option type. For instance, type(1:n) asserts that the match variables are a key in the master dataset. mmerge verifies whether this is (may be) true, and otherwise produces an error message, listing the duplicate key values. With type(1:n) it is *not* verified that the match variables are *not* a key in the using file; the match variables may by coincidence uniquely identify observations. mmerge, however, displays a message about key-ness also for datasets for which the match variables are not declared to be a key.

  mmerge also supports the following two derived matching types.

  - type(spread) is for people who cannot remember how to distinguish 1:n versus n:1 matching. mmerge will verify that you have a key in at least one of the two datasets.

  - type(auto) specifies that you don't know whether the match variables form a key. This is easy but circumvents the safety features that depend on the key-ness of matching variables, and should be avoided whenever possible. You should think about your data and tell mmerge whether matching variables are (should be) a key in some dataset.

- mmerge can treat missing values in the match variables three different ways:

  i) You may specify that you are convinced that there are no missing values. mmerge will verify this claim. This is the default behavior.

  ii) You may specify that missing values are to be treated like any other value. This is the strategy followed by many of the Stata data manipulation commands, but I think that this seldomly makes good sense in the context of merging datasets.

  iii) You may specify that missing values may occur in the match variables of the master and the using data, but that missing values in the master should *not* match missing values in the using data.

- mmerge has a simple way to specify whether unmatched observations from the master and using are to be retained in the result. For example, unmatched(master) specifies that unmatched observations from the master are retained.

  Like merge, mmerge generates a variable (with default name _merge) that describes "where an observation in the merge result comes from." The value 3 means that the observations have "two parents." merge uses the values 1 and 2 to identify observations that only occur in the master and only in the using data, respectively. mmerge extends this scheme by differentiating between two sources of mismatch; values −1 and −2 are used to mark unmatched observations that originate in the master and using data with missing values in the match variables. The values 1 and 2 are reserved for unmatched observations that are nonmissing in the match variables. Thus, in a merge of a dataset of person and a dataset of cities on the city of residence, _merge==-1 marks persons for whom the city of residence is unknown (missing), while _merge==1 identifies persons who live in a city not described in the database of cities.

  mmerge displays an understandable table describing the _merge variable.

- mmerge displays the names of variables in common to the master (data in memory) and using data; these will be included in the merged data only once, and a user may be confused as to the origin (and hence the interpretation) of such variables. Following mmerge, the common variable contains the values of the master data, unless options replace and update specify otherwise. Currently, mmerge does not inform the user whether common variables take the same values in the master and using data. It should in a future release of mmerge.

Convenience features of mmerge

mmerge provides a number of features to make merging convenient to the user:

- mmerge provides options for two standard merge files operations:

  - The option simple should be specified if you are convinced that the master and using data describe the same objects, identified by the match variables. simple is equivalent to specifying type(1:1) and unmatched(both), and typing assert _merge==3 upon the completion of mmerge.

  - table specifies table-lookup merging in which the master data describing objects of type A (e.g., persons) with an embedded type B object (e.g., city of residence), is enlarged with the properties of cities from a dataset of objects of type B (e.g., cities). This is equivalent to specifying type(n:1) and unmatched(master).

- While merge requires you to sort the master and using data explicitly by hand, mmerge does this for you. Third, mmerge offers a number of features for the manipulation of the using data file. (For the master data, one can do these easily before invoking `mmerge`.) These features are requested via a series of options that start with `u` to indicate that they manipulate the using data. One can:

- Select observations (via option `uif(`*exp*`)`).
- Select variables (options `ukeep(`*varlist*`)` and `udrop(`*varlist*`)`).
- Rename the matching variables (option `umatch(`*varlist*`)`),
- Prefix the names of variables or fully rename variables (options `uname(`*stub*`)` and `urename(`*stub*`)`).
- Prefix a string to the variable labels (`ulabel(`*stub*`)`).

Of course, one can do without these features, and do everything by hand. This would, however, often mean that one would have to modify the type-B file each time one would want to merge this file into some type A file. A large fraction of this is standard work, and hence could and should be made automatic by a declarative command such as `mmerge`. Also, ideally, if all is done by hand, one should do this within a do-file in which the merge is performed, and one should remove the modified type-B file immediately afterwards. However, I am not in an ideal world, or more accurately I am sloppy, and very similar data files too often clutter my hard disk, and it is sometimes hard to decide which of the files can be deleted. I doubt that this is peculiar to me.

Merging with implicit matching

The Stata command `merge` also supports merging without matching variables. In the Stata Reference Manual this is described as *one-to-one merging*. This should not be confused with *one-to-one matching* as supported by `mmerge`. One-to-one merging involves the "horizontal concatenation" of variables from two datasets. A link between the ith record in dataset 1 and the ith record in dataset 2 is only implicit, and this is the real danger of one-to-one merging. Thus the merge result is crucially dependent on the *physical order* of the observations in the master data and the using data. It is too easy to make mistakes, e.g., one of the datasets is in the wrong order, with awful consequences. Thus, one-to-one merging should be avoided unless you really know what you are doing. In the Stata reference manual a firm warning is issued against one-to-one merging, but the `merge` command fails to warn the user that one-to-one merging is being performed; I would welcome an option that has to be specified if one-to-one merging is to be performed, that tells `merge`: "Leave me alone, I know what I am doing." At times, I performed merging without matching variables because I accidentally omitted the planned match variable; only finding out my mistake much later. `mmerge` does not support one-to-one merging; there is no safe way to play with dynamite, and the unsafe should not be (made) easy!

Example of 1:1 matched merging

This example is adapted from the section on `merge` in the Stata Reference Manual. For this example, I have split the variables in the automobile data into three datasets `autotech`, `autocost`, and `automore`:

```
. describe using autotech

Contains data                                 1978 Automobile Data
  obs:            74                          6 Oct 1999 12:17
 vars:             4
 size:         2,072
-------------------------------------------------------------------------------
   1. make      str18  %-18s                  Make and Model
   2. mpg       int    %8.0g                  Mileage (mpg)
   3. weight    int    %8.0gc                 Weight (lbs.)
   4. length    int    %8.0g                  Length (in.)
-------------------------------------------------------------------------------
Sorted by:

. describe using autocost

Contains data                                 1978 Automobile Data
  obs:            74                          6 Oct 1999 12:17
 vars:             3
 size:         1,924
-------------------------------------------------------------------------------
   1. make      str18  %-18s                  Make and Model
   2. price     int    %8.0gc                 Price
   3. rep78     int    %8.0g                  Repair Record 1978
-------------------------------------------------------------------------------
Sorted by:

. describe using automore
```

```
Contains data                                 1978 Automobile Data
  obs:            74                          6 Oct 1999 12:17
 vars:             2
 size:         1,776
-------------------------------------------------------------------------------
   1. make      str18  %-18s                  Make and Model
   2. displ     int    %8.0g                  Displacement (cu. in.)
-------------------------------------------------------------------------------
Sorted by:
```

I want to merge the datasets into one. As `mmerge` can only merge a file “into” the master data, i.e., the data in memory, we have to take two merge steps. We start with `autotech` and `autocost`. As both datasets describe the same objects, I have to specify the option `type(1:1)`.

```
. use autotech, clear
(1978 Automobile Data)

. mmerge make using autocost, type(1:1)

-------------------+-----------------------------------------------------------
     matching type | 1:1
mv´s on match vars | none
unmatched obs from | both
-------------------+-----------------------------------------------------------
master        file | autotech.dta
               obs |    74
              vars |     4
        match vars | make  (key)
-------------------+-----------------------------------------------------------
using         file | autocost.dta
               obs |    74
              vars |     3
        match vars | make  (key)
-------------------+-----------------------------------------------------------
  common variables | none
-------------------+-----------------------------------------------------------
result        file | autotech.dta
               obs |    74
              vars |     7  (including _merge)
-------------------+-----------------------------------------------------------

all observations come from master and using data (_merge==3)
```

Compared to `merge`, `mmerge` provides an extensive report describing what has happened (the report may actually be suppressed with the option `noshow`). `mmerge` has verified that `make` is a key in both `autotech` and `autocost`, and has found out that the master and using data have no variables in common apart from the matching variables. Then `mmerge` sorted these datasets, merged them, and displayed a report on the result. Since `_merge==3` for all observations, we conclude that `autotech` and `autocost` describe the same cars.

We proceed to a one-to-one merge of the third dataset `automore` with additional properties of the cars.

```
. mmerge make using automore, type(1:1)

-------------------+-----------------------------------------------------------
     matching type | 1:1
mv´s on match vars | none
unmatched obs from | both
-------------------+-----------------------------------------------------------
master        file | autotech.dta
               obs |    74
              vars |     6
        match vars | make  (key)
-------------------+-----------------------------------------------------------
using         file | automore.dta
               obs |    74
              vars |     2
        match vars | make  (key)
-------------------+-----------------------------------------------------------
  common variables | none
-------------------+-----------------------------------------------------------
result        file | autotech.dta
               obs |    75
              vars |     8  (including _merge)
-------------------+-----------------------------------------------------------
```

```
_merge                            code  |   freq
----------------------------------------+--------
          only in master data        1  |      1
           only in using data        2  |      1
both in master and using data        3  |     73
----------------------------------------+--------
                                Total   |     75
```

Everything seems to work fine. But is it? There is a problem: I had expected that the datasets described the same cars, and hence _merge== 3 in all observations. This is not the case. There appears to be one observation that originates in the master data (the previous merge), and one that originates in the using data (automore). What is wrong? Somehow, the matching on make is not as expected. Let's take a look at the values of make in the unmatched observations:

```
. list make if _merge!=3

     make
 72. VW Rabbit
 75. Volkswagen Rabbit
```

Now it is obvious. In one dataset I entered the abbreviation 'VW', and in the other I spelled out 'Volkswagen.' After fixing this problem in automore.dta, and rerunning the above two mmerge commands again, everything is correct.

Next, I want to include the merging commands and subsequent analyses in a do-file. At this stage, I am sure that one-to-one matching works fine. In fact, I would like to *assert* that this is true, so that if I later rerun the do-file, and make other modifications to the datasets, I can be sure that the same cars are described in the three datasets. mmerge's type has a value simple to simplify this. It says in effect: "Both datasets describe the same objects. If this is not the case, please tell me so, and do not continue."

```
. use autotech
. mmerge make using autocost, type(simple)
  (output omitted )
. mmerge make using automore, type(simple)
  (output omitted )
. (analyses)
```

Example of n:1 matched merging

In this example, I focus on table-lookup merging using a number of data files. In a household data file (hn95data), I have variables that specify for 1,535 two-partner households the cities in which the couple lives and works, some household variables such as the number of children living in the household, and individual variables (the race of the husband and the wife). The file cities contains variables describing properties of 640 cities.

```
. describe using hn95data

Contains data                                 Households in the Netherlands
                                                survey 1995
  obs:         1,534                          14 Sep 1999 14:18
 vars:            10
 size:        27,612
-------------------------------------------------------------------------------
   1. hhid      int    %9.0g                  household id
   2. city      int    %9.0g                  city of residence
   3. hhinc     byte   %9.0g                  household income
   4. nkids     byte   %9.0g                  nr of kids
   5. hrace     byte   %9.0g                  husband: race
   6. hwcity    int    %9.0g                  work husband: city
   7. wrace     byte   %9.0g                  wife: race
   8. wwcity    int    %9.0g                  work wife: city
   9. religion  byte   %9.0g                  religion of couple
  10. vchurch   byte   %9.0g                  frequency church visit couple
-------------------------------------------------------------------------------
Sorted by:

. describe using cities

Contains data                                 database by city
  obs:           640                          14 Sep 1999 14:18
 vars:             5
 size:         5,760
-------------------------------------------------------------------------------
   1. cityid    byte   %9.0g                  city identifier
   2. inhabit   byte   %9.0g                  nr of inhabitants
   3. region    byte   %9.0g                  region
   4. jobs      byte   %9.0g                 nr of jobs
   5. vacant    byte   %9.0g                 nr of vacancies
-------------------------------------------------------------------------------
Sorted by:
```

I want to add information about the city in which the couple lives to the household file that I have in memory as the master file. Using the standard Stata command merge, I would have to proceed as follows:

```
. use cities
. rename cityid city
. sort city
. save TEMPNAME
. use hn95data, clear
. sort city
. merge city using TEMPNAME
. tab _merge
. drop TEMPNAME
```

The merging step can be accomplished far more simply by mmerge, using the option umatch() to specify the name of the matching variable in the using data (cities).

```
. use hn95data, clear
. mmerge city using cities, table umatch(cityid)
match-var in using data should form a key
error: duplicate values in match-var(s)
Did you mix up match types n:1 and 1:n?
Non-unique key values
   cityid     __FREQ
      265          2
      640          4
```

What is happening? mmerge complains that the file cities is not suitable for table lookup (n:1 matching) since the match variable cityid in cities is not a key; it does not uniquely define observations. It also lists the duplicate key values, in this case 265 and 640. In the code fragment using merge, this problem probably would not have become clear immediately, if at all. mmerge has located the problem for you. After fixing the problem with the key, I reissue the mmerge command:

```
. mmerge city using cities, table umatch(cityid)
-------------------+-----------------------------------------------------------
     matching type | n:1
mv´s on match vars | none
unmatched obs from | master
-------------------+-----------------------------------------------------------
master        file | hn95data.dta
               obs |  1534
              vars |    10
        match vars | city  (not a key)
-------------------+-----------------------------------------------------------
using         file | cities.dta
               obs |   640
              vars |     5
        match vars | cityid  (key)
-------------------+-----------------------------------------------------------
  common variables | none
-------------------+-----------------------------------------------------------
result        file | hn95data.dta
               obs |  1534
              vars |    15  (including _merge)
-------------------+-----------------------------------------------------------
_merge                               code  |   freq
-------------------------------------------+--------
matchvar==missing in master data      -1   |     71
             only in master data       1   |    115
   both in master and using data       3   |   1348
-------------------------------------------+--------
                                    Total  |   1534
```

This time mmerge performed as planned. It verified that cityid is now a key and sorted the data. It also states that city is not a key in the master data, i.e., city either has missing values and/or has duplicate values in the matching variables. This is no surprise; survey data have missing values, and there is no reason to believe that all sampled households live in different cities. Next, mmerge extracted relevant information from the using data, and performed the matched merge. Also, it provided details about the merging process. Note that mmerge has analyzed whether the master and using data have no variables in common apart from matching variables. Had there been variable names in common, mmerge would not have imported such variables from the using data into the master data in memory, which is usually an unexpected and undesirable affair in table-lookup matching.

Finally, mmerge reports the status of the observations in the master data after merging was completed. As explained above, mmerge has expanded the set of possible _merge values with a value −1 that is generated (only in case of table lookup merging) if the key variable(s) in the master is missing. In our example, there are 75 households for which the city of residence is missing. Next, mmerge reports that there are 113 households for which the city of residence is nonmissing, but for which no information is available in the cities file. In a serious analysis, one would need to seek to extend the cities file with extra records on cities currently not covered. Here I simply put the problem aside.

A careful reader may have noticed that there are no observations in the merged file with _merge==2. One can expect that the cities dataset contains records on cities that do not occur in the household data file (this is indeed the case). In contrast with merge, mmerge ignores these records by default in case of table lookup merging, i.e., mmerge sets unmatched(master).

I move one step further, and add the region, number of vacancies, and number of jobs in the cities in which the husband and wife work to the household data. A naming conflict is facing us, as we can't have two variables named vacant in one dataset. Moreover, we already have a variable region in the master dataset, namely the region in which the couple lives, and we can't import the region variable from the cities file for the regions in which the husband and wife work without some extra effort. Performing a straightforward merge would leave variables such as region unchanged, i.e., the variables will be associated with the city of residence, and so fail to import information about the cities in which the couple work. mmerge provides via the option uname(*stub*) a simple mechanism to deal with this common problem, namely prefixing a "stub" (string) to the names of variables to be imported. One clearly has to be careful that the prefixed variable names, truncated to eight characters, are still unique, so generally the stubs should not exceed one or two characters; mmerge will notice the problem.

Personally, I like to use variable labels abundantly. In this case, I like to have informative labels for the added variables. This is facilitated by the ulabel option of mmerge that specifies a prefix to the variables labels of the variables in the using data.

To do the relevant table lookup merging, one could issue the commands

```
. mmerge hwcity using cities, table umatch(cityid) uname(HW) ulabel(work husband)
-------------------+----------------------------------------------------------
     matching type | n:1
mv´s on match vars | none
unmatched obs from | master
-------------------+----------------------------------------------------------
master        file | hn95data.dta
               obs |  1534
              vars |    14
        match vars | hwcity  (not a key)
-------------------+----------------------------------------------------------
using         file | cities.dta
               obs |   640
              vars |     5
        match vars | cityid  (key)
-------------------+----------------------------------------------------------
  common variables | none
-------------------+----------------------------------------------------------
result        file | hn95data.dta
               obs |  1534
              vars |    19  (including _merge)
-------------------+----------------------------------------------------------
_merge                               code  |  freq
-------------------------------------------+--------
matchvar==missing in master data      -1   |   143
             only in master data       1   |   122
   both in master and using data       3   |  1269
-------------------------------------------+--------
                                   Total   |  1534
```

and, with some abbreviation of option names:

```
. mmerge wwcity using cities, ta um(cityid) un(WW) ul(work wife)
```

```
-------------------+-----------------------------------------------------------
     matching type | n:1
mv´s on match vars | none
unmatched obs from | master
-------------------+-----------------------------------------------------------
master        file | hn95data.dta
               obs |  1534
              vars |    18
        match vars | wwcity  (not a key)
-------------------+-----------------------------------------------------------
using         file | cities.dta
               obs |   640
              vars |     5
        match vars | cityid  (key)
-------------------+-----------------------------------------------------------
  common variables | none
-------------------+-----------------------------------------------------------
result        file | hn95data.dta
               obs |  1534
              vars |    23  (including _merge)
-------------------+-----------------------------------------------------------

_merge                               code  |   freq
-------------------------------------------+--------
matchvar==missing in master data      -1   |    314
             only in master data       1   |    103
   both in master and using data       3   |   1117
-------------------------------------------+--------
                                   Total   |   1534
```

Let us look at the variable description of the data in memory:

```
. describe

Contains data from hn95data.dta
  obs:         1,534                          Households in the Netherlands
                                                survey 1995
 vars:            23                          6 Oct 1999 14:00
 size:        47,554 (99.8% of memory free)
-------------------------------------------------------------------------------
   1. hhid      int    %9.0g                  household id
   2. city      int    %9.0g                  city of residence
   3. hhinc     byte   %9.0g                  household income
   4. nkids     byte   %9.0g                  nr of kids
   5. hrace     byte   %9.0g                  husband: race
   6. hwcity    int    %9.0g                  work husband: city
   7. wrace     byte   %9.0g                  wife: race
   8. wwcity    int    %9.0g                  work wife: city
   9. religion  byte   %9.0g                  religion of couple
  10. vchurch   byte   %9.0g                  frequency church visit couple
  11. inhabit   byte   %9.0g                  nr of inhabitants
  12. region    byte   %9.0g                  region
  13. jobs      byte   %9.0g                  nr of jobs
  14. vacant    byte   %9.0g                  nr of vacancies
  15. HWinhabi  byte   %9.0g                  work husband: nr of inhabitants
  16. HWregion  byte   %9.0g                  work husband: region
  17. HWjobs    byte   %9.0g                  work husband: nr of jobs
  18. HWvacant  byte   %9.0g                  work husband: nr of vacancies
  19. WWinhabi  byte   %9.0g                  work wife: nr of inhabitants
  20. WWregion  byte   %9.0g                  work wife: region
  21. WWjobs    byte   %9.0g                  work wife: nr of jobs
  22. WWvacant  byte   %9.0g                  work wife: nr of vacancies
  23. _merge    byte   %32.0g      __MERGE
-------------------------------------------------------------------------------
```

You may have noticed that `describe` informs us that the variable `_merge` has an attached variable label `__MERGE`. This label is produced by `mmerge` to improve the usefulness of `_merge`. (The value labels depend on whether `update` matching is performed. `mmerge` only produces the label if there is no variable label in the data named `__MERGE`, unless it was produced by `mmerge` itself.)

In the rest of this example, I demonstrate two further applications of `mmerge`. First, I want to look up regional information about the region in which the husband works, contained in the variable `HWregion`. Note that `HWregion` did not occur in the original household data (`hn95data`), but was created by a previous table-lookup merge. Characteristics (actually, only an indicator for regional economic development) of the 10 regions in the Netherlands are described in dataset `regions`.

```
. describe using regions
Contains data                                 database by region
  obs:            10                          6 Oct 1999 14:00
 vars:             2
 size:            60
-------------------------------------------------------------------------------
   1. regid     byte   %9.0g                  region identifier
   2. growth    byte   %9.0g                  regional economic growth in 1990
-------------------------------------------------------------------------------
Sorted by:
. mmerge HWregion using regions, table um(regid) un(HW) ul(work husband) noshow
 _merge                                code  |   freq
 -------------------------------------------+--------
 matchvar==missing in master data      -1   |    265
    both in master and using data       3   |   1269
 -------------------------------------------+--------
                                    Total   |   1534
```

In the final step in this example, I illustrate that `mmerge` supports table lookup with compound keys, i.e., keys that consist of more than one variable. The dataset `cregfile` describes the number of churches of 10 different denominations in each of the 640 cities in 1970, 1980, and 1990.

```
. describe using cityregi
Contains data                                 database by city and religion
  obs:         6,400                          6 Oct 1999 14:00
 vars:             5
 size:        64,000
-------------------------------------------------------------------------------
   1. idcity    int    %9.0g                  city identifier
   2. idrelig   byte   %9.0g                  religion identifier
   3. nchurch7  byte   %9.0g                  nr of churches in 1970
   4. nchurch8  byte   %9.0g                  nr of churches in 1980
   5. nchurch9  byte   %9.0g                  nr of churches in 1990
-------------------------------------------------------------------------------
Sorted by:
```

I now `mmerge` in the data on 1990, renaming the variable at the same time, (multiple rename clauses are to be separated by a backslash):

```
. mmerge city religion using cityregi, table umatch(idcity idrelig)
>  ukeep(nchurch9) urename(nchurch9 nch1990)
-------------------+-----------------------------------------------------------
     matching type | n:1
mv´s on match vars | none
unmatched obs from | master
-------------------+-----------------------------------------------------------
master        file | hn95data.dta
               obs |  1534
              vars |    23
        match vars | city religion  (not a key)
-------------------+-----------------------------------------------------------
using         file | cityregi.dta
               obs |  6400
              vars |     3  (selection via udrop/ukeep)
        match vars | idcity idrelig  (key)
-------------------+-----------------------------------------------------------
  common variables | none
-------------------+-----------------------------------------------------------
result        file | hn95data.dta
               obs |  1534
              vars |    25  (including _merge)
-------------------+-----------------------------------------------------------
_merge                                code  |   freq
-------------------------------------------+--------
matchvar==missing in master data      -1   |     76
             only in master data       1   |    257
   both in master and using data       3   |   1201
-------------------------------------------+--------
                                   Total   |   1534
```

Syntax for important special cases

`mmerge` *match_variable(s)* `using` *filename* [`,` { `simple` | `table` } `umatch(`*varlist*`)` `ukeep(`*varlist*`)`]

Full syntax

`mmerge` *match_variable(s)* `using` *filename* [`,` `type(`*type_value*`)` `unmatched(`*unmatched_value*`)`

{ `simple` | `table` } `missing(`*missing_value*`)` `nolabel` `replace` `update` `_merge(`*varname*`)`

`noshow` `ukeep(`*varlist*`)` `udrop(`*varlist*`)` `uif(`*exp*`)` `umatch(`*varlist*`)` `uname(`*stub*`)`

`urename(`*rename_specs*`)` `ulabel(`*stub*`)`]

where *type_value* is of the form

{ `auto` | `1:1` | `1:n` | `n:1` | `n:n` | `spread` }

unmatched_value is of the form

{ `both` | `none` | `master` | `using` }

missing_value is of the form

{ `none` | `value` | `nomatch` }

and *rename_specs* is of the form

oldname newname \ *oldname newname* \ . . .

Options to specify special cases

`simple` specifies matched merging in which the master and using data supposedly describe the same objects, fully identified by the match variables, i.e., the match variables form a key in the master and using data. `simple` is equivalent to specifying options `type(1:1)` and `unmatched(both)`, and invoking `assert _merge==3` upon the completion of `mmerge`.

`table` specifies a "table lookup merge", in which the master dataset contains data on objects of type A (e.g., households) that contain an object of type B (e.g., a city), identified by the match-variables (e.g., the city of residence), while the using data contains descriptions of type B (cities). Thus, the match variables should form a key in the using data. `table` is equivalent to specifying options `type(n:1)` and `unmatched(master)`.

In a future version of `mmerge`, I hope to provide additional "standard types of matched merging".

Options for matching

`type(`*type_value*`)` specifies whether the match variables are keys in the master and using data. Valid values are

`auto`	`mmerge` determines the match-type. (default)
`1:1`	key in the master and using data
`1:n`	key in master data
`n:1`	key in using data
`n:n`	no keys; `mmerge` performs a relational join via `joinby`
`spread`	`mmerge` determines which of `1:n` or `n:1` holds

While `auto` is easy, you are strongly advised to specify your knowledge of the relationship between the master and using data via one of the other values. This allows `mmerge` to test whether your understanding is consistent with the data.

`missing(`*missing_value*`)` specifies how missing values in the match variables are treated. Valid values are

`none`	missing values not allowed in the match vars (default)
`value`	missing values are treated as ordinary values
`nomatch`	missing values of the match vars in the master should not match missing values in the match-vars of the using data

Options for merging

unmatched(*unmatched_value*) specifies whether nonmatching observations in the master and using data are included in the merge result. Valid values are

both	nonmatching observations from master and using are included (default)
none	only completely matching observations are retained
master	nonmatching observations from master are included
using	nonmatching observations from using are included

nolabel prevents Stata from copying the value label definitions from the disk dataset. Even if you do not specify this option, in no event do label definitions from disk replace those already in memory.

update varies the action mmerge takes when an observation is matched. By default, the master data is held inviolate; values from the master data are retained when the same variables are found in both datasets. If update is specified, however, the values from the using data are retained in cases where the master data contains missing.

replace, allowed with update only, specifies that even in the case when the master dataset contains nonmissing values, they are to be replaced with corresponding values from the using data when corresponding data are not equal. A nonmissing value, however, will never be replaced with a missing value.

_merge(*varname*) specifies the name of the variable that will mark the source of the resulting observation. The default is _merge(_merge).

noshow specifies that the report on the files and the contents of _merge is suppressed.

Options for manipulating the using data (u options)

ukeep(*varlist*) and udrop(*varlist*) specify a *varlist* in the using data that has to be kept (dropped) before being merged into the master data. It is not valid to specify both ukeep and udrop. If neither is specified, all variables of the using data are used. The match variable(s) need not be specified in ukeep; they are automatically included in ukeep (excluded from udrop).

uif(*exp*) specifies that only the observations in the using data that meet expression *exp* are to be used. Properness of the key in the using data is determined *after* uif is processed.

umatch(*varlist*) specifies the names of the match variables in the using data. The umatch variables are associated with the match variables in the specified order. Clearly, the number of match variables in umatch should be the same as the number of matching variables in the master.

urename(*rename_specs*) specifies a list of *oldname newname* clauses to be applied to the using data, separated by a backslash. Note that urename is applied *after* ukeep and udrop, and hence ukeep and udrop should use the original names. It is not allowed to rename the match variables here. Use umatch instead. mmerge renames the umatch variables to the master match variable names after ukeep and udrop have been processed, but before urename is processed. An error occurs if there are naming conflicts.

uname(*stub*) specifies a stub prefixed to the names of the variables imported from the using data, truncated at eight characters. uname may not be combined with urename.

ulabel(*stub*) specifies a stub to be prefixed to the variable labels of the imported variables from the using data.

Miscellaneous commands

The command mmerge depends on two utility commands that may be of independent interest to some readers. The command tabl displays a one-variable tabulation with value labels as well as numeric codes. mmerge uses tabl to tabulate the contents of the _merge variable. tabl can also be invoked directly. See help tabl.

enumopt is a programmer's utility that facilitates the handling of options that are specified to take one of a series of values. enumopt support the conventional abbreviation rules of Stata, and allows one to set defaults. I wrote this utility for mmerge, but also believe that many Stata commands *seem* to have a rather complicated syntax because exclusive options are spelled out sequentially instead of as possible values of a single option. Given the fate of earlier parsing utilities that I contributed to the STB, I hope that options of type "enumerate" will be supported in a future release of Stata, so that enumopt can become obsolete as soon as possible.

Acknowledgments

mmerge significantly extends the contribution mergein by R. M. Farmer in STB-29. This project was supported by the Netherlands Organization for Scientific Research under grant PGS 370–50. I want to acknowledge the frequent and supportive

feedback from Bill Gould, Chris Snijders, and Fred Wolfe on early versions of this command. Comments and suggestions on the further development of `mmerge` are welcome indeed.

sg35.2 Robust tests for the equality of variances update to Stata 6

Mario Cleves, Stata Corporation, mcleves@stata.com

`robvar` originally published in STB-25 (Cleves 1995) computes Levene's robust test statistic for the equality of variances and two reformulations suggested by Brown and Forsythe based on robust estimators of central tendency.

`robvar` has been updated to Stata 6. It is now faster and all calculations are performed in double precision. Additionally, the command now accepts the `if` and `in` modifiers, and saves results in `r()`.

Syntax

`robvar` *varname* [`if` *exp*] [`in` *range*] , `by(`*groupvar*`)`

Description

Both the traditional F test for the homogeneity of variances and Bartlett's generalization of this test to K samples are highly sensitive to the assumption that the data are drawn from an underlying Gaussian distribution. Levene (1960) proposed a test statistic for equality of variance that is robust under non-normality. Subsequently Brown and Forsythe (1974) proposed alternative formulations of Levene's test statistic using more robust estimators of central tendency in place of the mean. These reformulations were demonstrated to be more robust than Levene's test when dealing with skewed populations.

`robvar` reports Levene's statistic (W_0) and two statistics proposed by Brown and Forsythe that replace the mean in Levene's formula with alternative location estimators. The first alternative (W_{50}) replaces the mean with the median. The second alternative replaces the mean with the 10 percent trimmed mean (W_{10}).

See Cleves (1995) for a complete description of this command.

Examples

We wish to test whether the standard deviation of the length of stay for patients hospitalized for a given medical procedure differs by gender. Our data consists of observations on the length of hospital stay for 1778 patients; 884 males and 894 females. Length of stay, `lgthstay`, is highly skewed (Skewness coefficient=4.912591) and thus violates Bartlett's normality assumption. We therefore use `robvar` to compare the variances.

```
. robvar lgthstay, by(sex)

   1:FEMALE |        Mean   Std. Dev.       Freq.
------------+-----------------------------------
          0 |   9.0874434   9.7884747         884
          1 |    8.800671   9.1081478         894
------------+-----------------------------------
      Total |   8.9432508   9.4509466        1778

W0=   .55505315   df(1, 1776)     Pr > F = .45635888

W50=  .42714734   df(1, 1776)     Pr > F = .51347664

W10=  .44577674   df(1, 1776)     Pr > F = .50443411
```

For his data we cannot reject the null hypothesis that the variances are equal, however, Bartlett's test yields a significance probability of 0.0319 due to the pronounced skewness of the data.

`robvar` saves in `r()`:

Scalars

`r(N)`	number of observations
`r(df1)`	numerator degrees of freedom
`r(df2)`	denominator degrees of freedom
`r(w0)`	Levene's F statistic
`r(p_w0)`	Levene's p-value
`r(w50)`	Brown and Forsythe's F statistic (median)
`r(p_w50)`	Brown and Forsythe's p-value (median)
`r(w10)`	Brown and Forsythe's F statistic (trimmed mean)
`r(p_w10)`	Brown and Forsythe's p-value (trimmed mean)

References

Brown, M. B. and A. B. Forsythe. 1974. Robust test for the equality of variances. *Journal of the American Statistical Association* 69: 364–367.

Cleves, M. 1995. sg35: Robust tests for the equality of variances. *Stata Technical Bulletin* 25: 13–15. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 91-93.

Levene, H. 1960. Robust tests for equality of variances. *Contributions to Probability and Statistics.* ed. I. Olkin, 278–292. California: Stanford University Press.

sg120.1 Two new options added to rocfit command

Mario Cleves, Stata Corporation, mcleves@stata.com

In STB-52 (Cleves, 1999), under the heading `roctab`, I introduced a series of commands for performing Receiver Operating Characteristic (ROC) analysis on rating and discrete classification data.

Although the nonparametric `roctab` command can be used with a continuous classification variable, the parametric `rocfit` model, an implementation of the popular approach developed by Dorfman and Alf (1969) for obtaining maximum likelihood estimates of the parameters of a smooth fitting ROC curve, can be difficult or impossible to estimate with a continuous classification variable. The reason for this difficulty is that the Dorfman and Alf approach requires that in addition to two regression parameters, $k-1$ fixed boundary parameters be simultaneously estimated, where k is the number of distinct values of the classification variable. A new option, `cont()`, allows us to group the continuous variable by specifying the number of groups to be created, and the `generate()` option generates a variable containing values indicating the groups.

Syntax

`rocfit` now has syntax

`rocfit` *refvar classvar* [*weight*] [`if` *exp*] [`in` *range*]

[`, cont(`*#*`)` `generate(`*newvar*`)` `level(`*#*`)` `nolog` *maximize_options*]

The `cont()` and `generate()` options are new. See Cleves (1999) for a description of the other options.

New options

`cont(`*#*`)` specifies that the continuous *classvar* be divided into # groups approximately of equal length. The option is required when *classvar* takes more than 20 distinct values.

`cont(.)` may be specified to indicate that *classvar* is to be used as it is, even though it could take more than 20 distinct values.

`generate(`*newvar*`)` specifies the name of the new variable to contain values indicating the groups produced by `cont(`*#*`)`. `generate(`*newvar*`)` is not valid without specifying `cont(`*#*`)` or with `cont(.)`.

Examples

If *classvar* has less than 20 distinct values then we can specify

```
. rocfit disease rating
```

This works whether `rating` is categorical or continuous.

If *classvar* has more than 20 distinct values, for example 25, and we wish to use the variable as it is, then we can specify

```
. rocfit disease rating, cont(.)
```

This also works whether `rating` is categorical or continuous, although presumably `rating` is categorical.

If *classvar* takes a large number of values, for example 100, and we wish to fit the model by creating 10 groups, then we can specify

```
. rocfit disease rating, cont(10)
```

This is appropriate only if `rating` is continuous. `cont(10)` specifies that `rating` should be categorized into 10 groups.

The `generate()` option can be used to generate a new variable containing the values indicating the groups created by `cont(10)`.

```
. rocfit disease rating, cont(10) generate(group)
```

Of course we could have grouped our data prior to analysis and used our own group variable as the *classvar*, omitting `cont(10)`.

References

Cleves, M. 1999. sg120: Receiver Operating Characteristic (ROC) analysis. *Stata Technical Bulletin* 52: 19–31.

Dorfman D. D. and E. Alf. 1969. Maximum likelihood estimation of parameters of signal detection theory and determination of confidence intervals-rating method data. *Journal of Mathematical Psychology*. 6: 487–496.

sg124 Interpreting logistic regression in all its forms

William Gould, Stata Corporation, wgould@stata.com

Abstract: The interpretation of logistic regression in all of its forms—ordinary, conditional, ordered, and multinomial—is explained. In all cases, exponentiated coefficients can be interpreted as some form of odds ratio. Guidance is provided on the accuracy of interpreting odds ratios as risk ratios.

Keywords: logit, logistic regression, conditional logistic regression, MacFadden choice model, fixed-effects logistic regression, ordered logistic regression, multinomial logistic regression; case–control, matched case–control; odds, odds ratios, conditional odds ratios, risk ratios.

Contents:

1. Ordinary (binary-outcome) logistic regression
   1.1 Odds
   1.2 Odds ratios
   1.3 Constancy of the odds ratios
   1.4 Interpreting logit output
   1.5 Actions of predict after logit and logistic
   1.6 Demonstration
2. Conditional logistic regression
   2.1 Derivation of model
   2.2 Interpreting clogit output
   2.3 Actions of predict after clogit
   2.4 Equivalency of conditional and ordinary logistic regression
3. Ordered logistic regression
   3.1 Odds ratios
   3.2 Demonstration
   3.3 Calculating confidence intervals for the odds ratios
   3.4 Actions of predict after ologit
   3.5 Equivalency of ordered and ordinary logistic regression
4. Multinomial logistic regression
   4.1 Relative Risk Ratio (RRR) interpretation
   4.2 Conditional Odds Ratio (COR) interpretation
   4.3 Demonstrations
   4.4 Interpreting mlogit output
   4.5 Actions of predict after mlogit
   4.6 Equivalency of multinomial and ordinary logistic regression
5. References

1. Ordinary (binary-outcome) logistic regression

The logistic regression or logit model is

$$\text{odds}(y_j \neq 0) = \exp(\mathbf{x}_j\mathbf{b} + b_0)$$

The Stata commands `logit` and `logistic` both report binary-outcome (ordinary) logistic regression estimates. Exponentiated coefficients have the interpretation of odds ratios (ORs). `logit` reports coefficients and `logistic` reports the exponentiated coefficients. For instance, `logit` might report a coefficient of .5 and `logistic` would correspondingly report $\exp(.5) = 1.6487$, labeling that result an odds ratio.

`logit` will report exponentiated coefficients (specify option `or`) and `logistic` will report unexponentiated coefficients (specify option `coef`), which command you use to estimate your model makes no difference.

1.1 Odds

Let p be the probability of an event. $o = p/(1-p)$ is called the odds of the event. Either way of expressing likeliness works equally well:

Probability p	Corresponding odds o
.0	.00
.1	.11
.2	.25
.3	.42
.4	.67
.5	1.00
.6	1.50
.7	2.33
.8	4.00
.9	9.00
1.0	∞

When probabilities are small, $p/(1-p)$ approximately equals p because $1-p$ is approximately 1. For unlikely events, epidemiologists often ignore that formal definition of the odds and talk about "risk" as if $o = p$. That works reasonably well for $p < .1$:

Probability p	Corresponding odds o
.1	.11...
.01	.0101...
.001	.001001...
.0001	.00010001...

1.2 Odds ratios

The exponentiated coefficient in an ordinary logistic regression has the interpretation

$$\frac{\text{odds(if the corresponding variable is incremented by } 1)}{\text{odds(if variable not incremented)}}$$

or, equivalently,

$$\frac{\text{P(event} \,|\, x+1) \,/\, \bigl(1 - \text{P(event} \,|\, x+1)\bigr)}{\text{P(event} \,|\, x) \,/\, \bigl(1 - \text{P(event} \,|\, x)\bigr)}$$

For instance, consider the model

```
. logit outcome female age
```

If the exponentiated coefficient on female is 1.5, then the odds of the event are 50 percent greater when female $==$ 1 than when female $==$ 0.

If the exponentiated coefficient on age is .5, then the odds of the event halve as age increases by 1 and they halve at every age. If age is measured in years, the odds halve for each yearly increase in age. If age is measured in 5-year spans (variable age is true age divided by 5), then the odds halve for each 5-year increment in age.

For unlikely events, epidemiologists will sometimes speak about odds ratios as if they were relative risks (risk ratios), just as they sometimes speak about odds as if they were risks. The approximation is reasonably accurate for $p < .1$:

Table of actual Risk Ratios given
Odds Ratios and P(event)

P(event$\|x$)	.25	.50	.75	1.00	1.50	2.00	4.00
.2	.2941	.5556	.7895	1.000	1.364	1.667	2.500
.1	.2702	.5263	.7692	1.000	1.429	1.818	3.077
.01	.2519	.5025	.7519	1.000	1.493	1.980	3.883
.001	.2502	.5003	.7502	1.000	1.499	1.998	3.988
.0001	.2500	.5000	.7500	1.000	1.500	2.000	3.999

Note: Let $p = \text{P(event}|x)$ and $q = \text{P(event}|x+1)$.
The odds ratio is then $o = (q/(1-q))/(p/(1-p))$.
Thus, given p and o, the value of q can be solved for:
$q = c/(1+c)$ where $c = op/(1-p)$.
The risk ratio is then q/p.

1.3 Constancy of the odds ratios

It is a remarkable property of logistic regression that the odds ratio of an effect is constant regardless of the values of the covariates. For instance, say you estimate the following logistic regression model:

$$-13.70837 + .1685\,x_1 + .0039\,x_2$$

The effect on the odds of a 1-unit increase in x_1 is $\exp(.1685) = 1.18$, meaning the odds increase by 18 percent. Incrementing x_1 increases the odds by 18 percent regardless of the value of x_2—it does not matter whether $x_2 = 0$ or $x_2 = 1000$. For every observation in the dataset, incrementing x_1 has the same multiplicative effect on the odds.

1.4 Interpreting logit output

Do not confuse coefficients with exponentiated coefficients. Look at the headers above the coefficient table. If it says `Coef`, then coefficients are being reported:

```
Logit estimates                                   Number of obs   =         74
                                                  LR chi2(2)      =      35.72
                                                  Prob > chi2     =     0.0000
Log likelihood = -27.175156                       Pseudo R2       =     0.3966

------------------------------------------------------------------------------
     dom |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |   .1685869   .0919174      1.834   0.067       -.011568    .3487418
  weight |   .0039067   .0010116      3.862   0.000        .001924    .0058894
   _cons |  -13.70837   4.518707     -3.034   0.002      -22.56487   -4.851864
------------------------------------------------------------------------------
```

If it says `Odds Ratio`, then exponentiated coefficients are being reported:

```
Logit estimates                                   Number of obs   =         74
                                                  LR chi2(2)      =      35.72
                                                  Prob > chi2     =     0.0000
Log likelihood = -27.175156                       Pseudo R2       =     0.3966

------------------------------------------------------------------------------
     dom | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |   1.183631   .1087963      1.834   0.067       .9884987    1.417283
  weight |   1.003914   .0010156      3.862   0.000       1.001926    1.005907
------------------------------------------------------------------------------
```

When exponentiated coefficients are reported, Stata does not report the intercept (labeled `_cons` in the prior output). A coefficient for the intercept is nonetheless estimated.

After seeing the output one way, you can see it the other if you wish. Learn about `logit`'s `or` and `logistic`'s `coef` options.

1.5 Actions of predict after logit and logistic

If you use `predict` after `logit` or `logistic`, you obtain probabilities, not odds. If you want the odds, you can calculate them for yourself:

```
. predict p
. gen odds = p/(1-p)
```

1.6 Demonstration

It is well worth demonstrating that you can obtain the same odds ratios that Stata reports. Try the following experiment:

```
. use auto, clear
. logistic foreign mpg weight
. predict double p
. replace mpg = mpg + 1
. predict double q
. gen double or = (q/(1-q)) / (p/(1-p))
. summarize or
```

You will observe that `or` has the same value reported by `logistic` as the odds ratio for `mpg`, namely .8448578. You will observe that the variance of variable `or` is zero (except for roundoff error), meaning `or` is constant regardless of the value of variable `weight`.

It is also instructive to compare the odds ratio to the risk ratio:

```
. gen double rr = q/p
. summarize or rr
```

The risk ratio will be .8879504, a little larger than the odds ratio of .8448578. Moreover, the risk ratio will not be constant, varying in this dataset between .8450 and .9816. In this dataset, the average probability of the event is .2973, which you can obtain by typing `summarize foreign`.

2. Conditional logistic regression

The conditional logistic regression model is

$$\text{odds}(y_j) \quad \textit{are proportional to} \quad \exp(\mathbf{x}_j\mathbf{b})$$

and notice that this definition includes, as a special case, the definition of ordinary logistic regression,

$$\text{odds}(y_j) = \exp(\mathbf{x}_j\mathbf{b} + b_0)$$

The exponentiated conditional logistic regression coefficients have the same odds-ratio interpretation as ordinary logistic estimates.

Stata's `clogit` command estimates conditional logistic regressions. Specify option `or` to obtain the exponentiated coefficients.

Conditional logistic regression differs from ordinary logistic regression in that the data are divided into groups and, within each group, the observed probability of positive outcome is either predetermined due to the data construction (such as matched case–control) or in part determined because of unobserved differences across the groups. Thus, the likelihood of the data depends on the conditional probabilities—the probability of the observed pattern of positive and negative responses within group conditional on that number of positive outcomes being observed. Terms that have a constant within-group effect on the unconditional probabilities—such as intercepts and variables that do not vary—cancel in the formation of these conditional probabilities and so remain unestimated.

2.1 Derivation of model

Models are typically asserted and it is their properties that are derived, but conditional logistic regression really is ordinary logistic regression applied to a particular data problem.

In the conditional logistic problem, the data occur in groups:

```
group 1:
        obs. 1  outcome=1   x1 = ...  x2 = ...
        obs. 2  outcome=0   x1 = ...  x2 = ...
group 2:
        obs. 3  outcome=1   x1 = ...  x2 = ...
        obs. 4  outcome=0   x1 = ...  x2 = ...
        obs. 5  outcome=0   x1 = ...  x2 = ...
group 3:
        ...
.
.
Group k:
        ...
```

We wish to condition on the number of positive outcomes within group. That is, we seek to fit a logistic model that explains why observation 1 had a positive outcome in group 1 *conditional on* one of the observations in the group having a positive outcome.

In biostatistical applications, this arises, for example, because researchers collect data on the sick and infected (the so-called "positive" outcomes) and then match those cases with controls who are not sick and infected. Thus, the number of positive outcomes is not a random variable. Within each group, there had to be the observed number of positive outcomes because that is how the data were constructed.

Economists refer to this same model as the McFadden choice model. In this model, an individual is faced with an array of choices and must choose one.

The estimator is also known as the fixed-effects logistic regression estimator for reasons that will become more obvious shortly.

Regardless of the justification, we are seeking to fit a model that explains why observation 1 had a positive outcome in group 1, observation 3 in group 2, and so on.

We assume the unconditional probability of a positive outcome is given by the standard logistic equation,

$$\text{odds}(y_j) = \exp(\mathbf{x}_j\mathbf{b} + b_0)$$

or equivalently,

$$\text{P(positive outcome)} = \exp(\mathbf{x}_j\mathbf{b} + b_0)/(1 + \exp(\mathbf{x}_j\mathbf{b} + b_0)) \tag{1}$$

Equation (1) is not the appropriate probability for our data because it does not account for the conditioning. In the first group, for instance, we want

$$\text{P(obs. 1 positive and obs. 2 negative} \,|\, \text{one positive outcome)}$$

and that is easy enough to write down in terms of the unconditional probabilities. It is

$$\frac{\text{P(1 positive) P(2 negative)}}{\text{P(1 positive) P(2 negative)} + \text{P(1 negative) P(2 positive)}} \tag{2}$$

From now on, when we write P(1 positive) and P(2 negative), etc., we will mean the probability that observation 1 had a positive outcome, the probability that observation 2 had a negative outcome, and so on.

Substituting equation (1) into (2), we obtain

$$\text{P(1 positive and 2 negative} \,|\, \text{one positive outcome)} = \frac{\exp(\mathbf{x}_1\mathbf{b})}{\exp(\mathbf{x}_1\mathbf{b}) + \exp(\mathbf{x}_2\mathbf{b})} \tag{3}$$

So that is the model we seek to fit or, at least, that is the term for group 1 and there are similar terms for all the other groups. (We have ignored the possibility of multiple positive outcomes within group, but that just adds complication.)

What is important to note in comparing equations (1) with (3)—in comparing ordinary logistic regression with conditional logistic regression—is that the logistic intercept b_0 cancelled. Whatever the value of b_0, it makes no difference in terms of the conditional outcomes that were observed and so cannot be estimated. Also note that b_0 could vary by group and it would still cancel. Thus, the conditional logistic estimator is often used to estimate the fixed-effects logistic model.

Finally note that, in equation (3), any variable that is constant within group will similarly cancel from both the numerator and denominator and so its effect cannot be estimated.

For a more thorough discussion of the conditional logistic derivation and its implications, see Gould (1999).

Groups that contain all-positive or all-negative outcomes provide no information because the conditional probability of observing such groups is 1 regardless of the values of the parameters **b**. Thus, when Stata encounters such groups, it reports that so many groups were dropped “due to all positive or negative outcomes”.

2.2 Interpreting clogit output

By default, `clogit` reports coefficients. Specify option `or` if you want exponentiated coefficients (odds ratios) reported. Do not confuse coefficients with exponentiated coefficients. Look at the headers above the coefficient table. If it says `Coef.`, then coefficients are being reported. If it says `Odds Ratio`, then exponentiated coefficients are being reported. In neither case is an intercept reported because the intercept remains unestimated.

2.3 Actions of predict after clogit

The default calculation by `predict` following `clogit` estimation is the conditional probability of a positive outcome given a single positive outcome within group. This is not the same probability that `predict` calculates following estimation by `logit` or `logistic`. The overall probability of a positive outcome cannot be calculated because the intercepts of the logit model remain unestimated.

2.4 Equivalency of conditional and ordinary logistic regression

Ordinary and conditional logistic regression produce the same result when there is only one group, however, conditional logistic regression still leaves the intercept unestimated.

Try the following experiment:

```
. use auto, clear
. gen grp = 1

. logit foreign mpg weight
. clogit foreign mpg weight, group(grp)
```

The coefficient reported by `logit` and `clogit` will be the same. `logit`, however, will report an intercept and `clogit` will not.

Results are similarly the same in the exponentiated coefficient (odds ratio) metric:

```
. logistic foreign mpg weight
. clogit foreign mpg weight, group(grp) or
```

3. Ordered logistic regression

In ordered logistic regression, there are multiple outcomes and we will label them 1, 2, 3, $\ldots$, k. The outcomes are ordered from weak to strong, mild to severe, etc. The model is

$$\text{odds}(\text{outcome more severe than } i) = \exp(\mathbf{x}_j\mathbf{b} + b_{0i})$$

Exponentiated ordered-logistic regression coefficients can be interpreted as odds ratios. In the ordered logistic case, it is the ratio, given a one-unit increase in the covariate, of the odds of being in a higher rather than a lower category.

Stata's `ologit` command estimates ordered-logistic regression. There is currently no option to report exponentiated coefficients; you must exponentiate the coefficients for yourself.

3.1 Odds ratios

Let there be k ordered outcomes, numbered 1, 2, 3, $\ldots$, k.

In ordered logistic regression, the exponentiated coefficients are the ratios, for a one-unit increase in the covariate, of the odds of outcome k to outcomes below k, and simultaneously for outcomes $k-1$ and above to outcomes below $k-1$, and simultaneously for outcomes $k-2$ and above to outcomes below $k-2$, and so on.

That is, you have ordered outcomes

```
1        2       3        4       5     ...
```

and, just to fix ideas, let's pretend we have exactly 5 outcomes. If you were to calculate the particular odds ratio comparing outcome 5 to all the outcomes below it,

```
 1        2       3        4       5
-----------------------------     ---
             |                     |
             +---------------------+
                        |
                        or = odds ratio for a 1-unit increase in x
```

you would obtain the same value as if you calculated the odds ratio comparing outcomes 4 and 5 to all outcomes below them,

```
 1        2       3        4       5
--------------------      -----------
          |                    |
          +--------------------+
                     |
                     or = odds ratio for a 1-unit increase in x
```

which would be the same value as the odds ratio comparing outcomes 3, 4, and 5 to the outcomes below that,

```
 1        2       3        4       5
------------     --------------------
     |                     |
     +---------------------+
                |
                or = odds ratio for a 1-unit increase in x
```

and so on.

3.2 Demonstration

To demonstrate this, use the auto data and estimate the model

```
. ologit rep78 mpg weight
```

Variable `rep78` takes on 5 (ordered) outcomes, 1 = Poor through 5 = Excellent.

Try the following:

```
. use auto, clear
. keep if rep78!=.

. ologit rep78 mpg weight

. predict double(p1 p2 p3 p4 p5)

. replace mpg = mpg + 1
. predict double(q1 q2 q3 q4 q5)

. gen double o5 = (q5/(q1+q2+q3+q4))/(p5/(p1+p2+p3+p4))
. gen double o4 = ((q5+q4)/(q1+q2+q3))/((p5+p4)/(p1+p2+p3))
. gen double o3 = ((q5+q4+q3)/(q1+q2))/((p5+p4+p3)/(p1+p2))
. gen double o2 = ((q5+q4+q3+q2)/(q1))/((p5+p4+p3+p2)/(p1))

. summarize o5 o4 o3 o2

. display exp(_b[mpg])
```

To explain,

1. First, we estimate the model `ologit rep78 mpg weight`.
2. We `predict p1 p2 p3 p4 p5`, obtaining the 5 predicted probabilities, observation by observation, of being in each of the `rep78` categories. Of course, $\texttt{p1} + \texttt{p2} + \texttt{p3} + \texttt{p4} + \texttt{p5} = 1$.
3. We add 1 to `mpg` and `predict q1 q2 q3 q4 q5`, thus obtaining the predicted probabilities when mpg is incremented by 1.
4. We `gen o5 = (q5/(q1+q2+q3+q4)) / (p5/(p1+p2+p3+p4))`. The numerator, `q5/(q1+q2+q3+q4)`, are the odds of being in the group $\texttt{rep78} == 5$ given `mpg` is incremented by 1. The denominator, `p5/(p1+p2+p3+p4)`, are the same odds when `mpg` is not incremented. New variable `o5` is the odds ratio.
5. We `gen o4 = ((q5+q4)/(q1+q2+q3))/((p5+p4)/(p1+p2+p3))`. This is just like the calculation above except now we are taking ratios for outcomes $\texttt{rep78} \geq 4$.
6. We `gen o3` and `gen o2`, taking ratios of outcomes $\texttt{rep78} \geq 3$ and $\texttt{rep78} \geq 2$.
7. We `summarize` all the `o` variables. We will discover that each is a constant and that they are all equal to each other!

We will also find that they are equal to $\exp(\texttt{_b[mpg]})$ from the `ologit` output. In this case, we will have to calculate $\exp()$ for ourselves because `ologit` does not have an odds-ratio option (although it should).

3.3 Calculating confidence intervals for the odds ratios

`ologit` does not report exponentiated coefficients. In the demonstration above, we obtain the following `ologit` output:

```
. ologit rep78 mpg weight

Iteration 0:   log likelihood = -93.692061
Iteration 1:   log likelihood = -86.794936
Iteration 2:   log likelihood = -86.513907
Iteration 3:   log likelihood = -86.513267
Iteration 4:   log likelihood = -86.513267

Ordered logit estimates                           Number of obs   =         69
                                                  LR chi2(2)      =      14.36
                                                  Prob > chi2     =     0.0008
Log likelihood = -86.513267                       Pseudo R2       =     0.0766

------------------------------------------------------------------------------
   rep78 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |   .1170693   .0712216      1.644   0.100      -.0225224    .2566611
  weight |  -.0003287   .0004849     -0.678   0.498       -.001279    .0006217
---------+--------------------------------------------------------------------
   _cut1 |  -2.300255   2.918117            (Ancillary parameters)
   _cut2 |  -.5423416    2.85169
   _cut3 |   1.794106   2.830686
   _cut4 |   3.410261   2.872927
------------------------------------------------------------------------------
```

Obtaining the odds ratio is easy enough:

For `mpg`, $\text{OR} = \exp(.1170693) = 1.124$.

For `weight`, $\text{OR} = \exp(-.0003287) = .99967$.

One way to obtain the 95% confidence intervals is to exponentiate the reported coefficient confidence intervals:

For `mpg`, $\text{OR} = \exp(.1170693) = 1.124$, and the 95% confidence interval is $[\exp(-.0225224), \exp(.2566611)] = [\,.978, 1.293\,]$.

For `weight`, OR $= \exp(-.0003287) = .99967$, and the 95% confidence interval is $[\exp(-.001279), \exp(.0006217)] = [\,.99872, 1.0006\,]$.

We could also obtain a standard error for the odds ratio using the delta rule—see Sribney and Wiggins (1999) for a description—which in this case results in SE(OR) $= \exp(b) \times$ SE(b):

For `mpg`, OR $= \exp(.1170693) = 1.124$, and the standard error is $1.124 \times .071226 = .0800$.

For `weight`, OR $= \exp(-.0003287) = .99967$, and the standard error is $.99967 \times .0004849 = .0004847$.

An easy way to obtain these results is to use `lincom` with the `or` option:

```
. lincom mpg, or
 ( 1)  mpg = 0.0
------------------------------------------------------------------------------
   rep78 | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     (1) |   1.124197   .0800671      1.644   0.100       .9777293    1.292607
------------------------------------------------------------------------------

. lincom weight, or
 ( 1)  weight = 0.0
------------------------------------------------------------------------------
   rep78 | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     (1) |   .9996714   .0004847     -0.678   0.498       .9987218    1.000622
------------------------------------------------------------------------------
```

3.4 Actions of predict after ologit

`predict` after `ologit` calculates probabilities, not odds. If you want the odds, you can calculate them for yourself:

```
. predict p1 p2 p3 p4 p5
. gen odds5 = p5/(p1+p2+p3+p4)
. gen odds4 = (p5+p4)/(p1+p2+p3)
. gen odds3 = (p5+p4+p3)/(p1+p2)
. gen odds2 = (p5+p4+p3+p2)/(p1)
```

3.5 Equivalency of ordered and ordinary logistic regression

When there are two outcomes, both ordered and ordinary logistic regression will produce the same results except for a confusing sign reversal. Try the following experiment:

```
. use auto, clear
. logit foreign mpg weight
. ologit foreign mpg weight
```

The coefficients on `mpg` and `weight` will be the same in both sets of output. `logit`, however, will report a coefficient for `_cons` of 13.70837 while `ologit` will report the ancillary parameter `_cut1` being −13.70837.

This difference is caused by how the parameters are used in their respective models. In `logit`, the interpretation is

$$\mathrm{P}(\texttt{foreign}_j) = \mathrm{P}(\mathbf{x}_j\mathbf{b} + b_0 + \textit{noise} > 0)$$

and in `ologit`, the interpretation is

$$\mathrm{P}(\texttt{foreign}_j) = \mathrm{P}(\mathbf{x}_j\mathbf{b} + \textit{noise} > \texttt{\_cut1})$$

Thus, $\texttt{\_cut1} = -b_0$.

4. Multinomial logistic regression

In multinomial logistic regression there are k outcomes 1, 2, ..., k. One of the outcomes is arbitrarily chosen as the "base outcome" as we will choose outcome k. The model is

$$\begin{aligned}
\mathrm{odds}(y_j = 1 \,|\, y_j = 1 \text{ or } y_j = k) &= \exp(\mathbf{x}_j\mathbf{b}_1 + c_1)\\
\mathrm{odds}(y_j = 2 \,|\, y_j = 2 \text{ or } y_j = k) &= \exp(\mathbf{x}_j\mathbf{b}_2 + c_2)\\
&\ \vdots\\
\mathrm{odds}(y_j = k-1 \,|\, y_j = k-1 \text{ or } y_j = k) &= \exp(\mathbf{x}_j\mathbf{b}_{k-1} + c_{k-1})\\
\mathrm{odds}(y_j = k \,|\, y_j = k \text{ or } y_j = k) &= 1
\end{aligned}$$

Note that if there are $k = 2$ outcomes, the model reduces to ordinary logit.

Exponentiated coefficients have two interpretations, that of (1) relative risk ratios (RRR) and (2) conditional odds ratios (COR). Both interpretations are relative to the base group, which in our case, we arbitrarily set to k. Had we chosen a different base group, the coefficients and their interpretations would change but the predictions of the model would not change.

Stata's `mlogit` command estimates multinomial logistic regressions. `mlogit` chooses a base group on its own—not necessarily k—but its choice may be overridden using its `basecategory()` option.

Specify option `rrr` to obtain the exponentiated coefficients, which will be labeled RRRs.

4.1 Relative Risk Ratio (RRR) interpretation

Relative risk is defined

$$r_1 = \mathrm{P}(y = 1) \,/\, \mathrm{P}(y = \text{base category})$$

$$r_2 = \mathrm{P}(y = 2) \,/\, \mathrm{P}(y = \text{base category})$$

and so on. Remember that odds are defined as

$$o_1 = \mathrm{P}(y = 1) \,/\, (1 - \mathrm{P}(y = 1))$$

$$o_2 = \mathrm{P}(y = 2) \,/\, (1 - \mathrm{P}(y = 2))$$

and so on. In the case of two outcomes, $1 - \mathrm{P}(y = 1) = \mathrm{P}(y = \text{base category})$ and odds and relative risks are equal. If there were more than two categories, however, they would differ. For instance,

Category	Probability	Odds	Rel. Risk rel. to category 3
1	.3	.429	1.5
2	.5	1.000	2.5
3 (base)	.2	.250	1.0

Exponentiated coefficients in ordinary logit are odds ratios—the ratio of the odds for a one-unit increase in x to the odds when x is unchanged:

$$\text{OR} = \frac{\mathrm{P}(y = 1 \,|\, x + 1) \,/\, (1 - \mathrm{P}(y = 1 \,|\, x + 1))}{\mathrm{P}(y = 1 \,|\, x) \,/\, (1 - \mathrm{P}(y = 1 \,|\, x))}$$

Exponentiated coefficients in multinomial logistic regression are relative risk ratios—the ratio of the relative risk for a one-unit increase in x to the relative risk when x is unchanged:

$$\text{RRR} = \frac{\mathrm{P}(y = 1 \,|\, x + 1) \,/\, \mathrm{P}(y = \text{base category} \,|\, x + 1)}{\mathrm{P}(y = 1 \,|\, x) \,/\, \mathrm{P}(y = \text{base category} \,|\, x)}$$

The $\text{RRR} = \text{OR}$ when $\mathrm{P}(\text{base category}) = 1 - \mathrm{P}(y = 1)$, that is, when there are two outcomes. When there are more than two outcomes, ORs and RRRs are different. For instance, let's pretend

Category i	$\mathrm{P}(y = i \vert x)$	$\mathrm{P}(y = i \vert x + 1)$
1	.3	.4
2	.5	.3
3 (base)	.2	.3

Then the ORs and RRRs are

Category	OR	RRR
1	1.56	.89
2	.43	.40
3 (base)	1.71	1.00

Note how difficult RRRs can be to interpret. The probability of $y = 1$ increases and yet the RRR falls because the probability of the base category increases, too, and it increased even more.

4.2 Conditional Odds Ratio (COR) interpretation

(The author of this insert thanks Roland Perfekt of the Southern Swedish Regional Tumour Registry in Lund, Sweden, for pointing out this interpretation.)

The exponentiated coefficients from multinomial logistic regression can just as well be given the interpretation of Conditional Odds Ratios, which are defined as

$$\text{COR}_1 = \frac{\text{odds}(y = 1 \,|\, x+1 \text{ and } (y = 1 \text{ or } y = \text{base category}))}{\text{odds}(y = 1 \,|\, x \text{ and } (y = 1 \text{ or } y = \text{base category}))}$$

$$\text{COR}_2 = \frac{\text{odds}(y = 2 \,|\, x+1 \text{ and } (y = 1 \text{ or } y = \text{base category}))}{\text{odds}(y = 1 \,|\, x \text{ and } (y = 1 \text{ or } y = \text{base category}))}$$

and so on. Although we listed RRR ahead of COR, CORs are perhaps a more natural interpretation. Since CORs and RRRs are both equal to the same exponentiated coefficients, whether one uses CORs or RRRs is just a matter of taste.

In the description of RRRs, we offered the following hypothetical set of results

Category i	P$(y = i \vert x)$	P$(y = i \vert x+1)$
1	.3	.4
2	.5	.3
3 (base)	.2	.3

Category	OR	RRR
1	1.56	.89
2	.43	.40
3 (base)	1.71	1.00

The RRRs in this table could just as well be labeled CORs and, if we do that, the interpretation is easier. We previously noted that the probability of $y = 1$ increases and yet the RRR falls because the probability of the base category increases and the base increased even more.

Said in the COR way, we would merely note the odds of being in 1 versus the base (category 3) fall when x increases.

4.3 Demonstrations

Try the following:

```
. use auto, clear
. drop if rep78==. | rep78==1 | rep78==2
. gen outcome = 1 if rep78==3
. gen outcome = 2 if rep78==4
. gen outcome = 3 if rep78==5
```

So far, we have just created a 3-outcome problem (and we will ignore its ordered nature). Next, we estimate our model:

```
. mlogit outcome mpg foreign, base(3) rrr
```

We can now obtain the probabilities `p1`, `p2`, and `p3`. We will then increment `mpg` by 1 and obtain in `q1`, `q2`, and `q3` the probabilities associated with a 1-unit increase in the mpg.

```
. predict double(p1 p2 p3)
. replace mpg = mpg + 1
. predict double(q1 q2 q3)
```

Now we will obtain the RRR for `outcome` == 1:

```
. gen double rrr = (q1/q3) / (p1/p3)
```

We will next obtain the COR for `outcome` == 1, first obtaining the conditional probabilities:

```
. gen double p1g13 = p1/(p1+p3)
. gen double q1g13 = q1/(q1+q3)
. gen double cor = (q1g13/(1-q1g13)) / (p1g13/(1-p1g13))
```

Finally, we can compare results

```
. summarize rrr cor
```

You will obtain a mean of .8422838 for both RRR and COR, both with standard deviations 0 save for roundoff error. (The standard deviation being zero is important; that's what demonstrates that RRRs and CORs are independent of the other values of the covariates.)

`mlogit` will have reported .8422838 for the RRR in its original output.

4.4 Interpreting mlogit output

`mlogit` reports coefficients or, if you specify option `rrr`, exponentiated coefficients (RRRs or CORs). Do not confuse coefficients with exponentiated coefficients. Look at the headers above the coefficient table. If it says `Coef.`, then coefficients are being reported. If it says `RRR`, then exponentiated coefficients are being reported.

4.5 Actions of predict after mlogit

`predict` after `mlogit` calculates probabilities, not odds or relative risks:

```
. predict p1 p2 p3 p4 p5
```

If you want other values, you can calculate them from the probabilities.

4.6 Equivalency of multinomial and ordinary logistic regression

When there are two outcomes, multinomial and ordinary logistic regression produce the same results. Try the following experiment:

```
. use auto, clear
. logit foreign mpg weight
. mlogit foreign mpg weight
```

If you prefer results exponentiated, type

```
. logistic foreign mpg weight
. mlogit foreign mpg weight, rrr
```

This is perhaps one more reason to prefer the COR to RRR interpretation of exponentiated coefficients in the multinomial logistic model; it is more obvious that the CORs are ORs when there are only two outcomes.

5. References

Gould, W. 1999. Within-group collinearity in conditional logistic regression. In *Stata FAQs*. Available http://www.stata.com/support/faqs/stat/clogitcl.html. College Station, TX: Stata Corporation.

Sribney, W. and V. Wiggins. 1999. Standard errors, confidence intervals, and significance tests for ORs, HRs, IRRs, and RRRs. In *Stata FAQs*. Available http://www.stata.com/support/faqs/stat/2deltameth.html. College Station, TX: Stata Corporation.

sg125 Automatic estimation of interaction effects and their confidence intervals

Jokin de Irala-Estévez, University of Navarre, Pamplona, Spain, jdeirala@unav.es

Miguel Angel Martínez, University of Navarre, Pamplona, Spain

Interaction or effect modification refers to the biological situation where the effect of a putative causal factor under study is modified by another factor; see Hosmer and Lemeshow (1989). Effect modification is identified in multivariate analyses by testing the statistical significance of biologically sound interactions between the variables in a main-effects model. This is performed by including and evaluating the significance of second or higher order terms involving the two or more variables that are postulated to possibly modify their respective effects.

The consequence of the identification of variables as significant-effect modifiers is that the effect on the outcome of one of those variables will depend on the values taken by the other variable(s) involved in the interaction. This implies that the coefficients of the models obtained by any statistical packages cannot be directly interpreted without performing further calculations. The only model coefficients that can be directly used to estimate odds ratios are those not included in interaction terms. The remaining odds ratios and their corresponding confidence intervals have to be estimated across the different levels of the other variables of the interaction term (across different categories if the variable is qualitative or across a series of values, sometimes the minimum, mean, and maximum, if it is quantitative). The major difficulty in this process lies in the correct estimation of the variance of each of these odds ratios.

The variance (or its square root to obtain the standard error) has to be calculated from a linear sum of regression coefficients, variances, and covariances of the estimated coefficients. The values of these variances and covariances are generally obtained from a variance–covariance matrix after fitting the model, and this is not easy with all statistical packages. The estimation of such point estimates and their confidence intervals can be cumbersome and authors such as Hosmer and Lemeshow (1989) and Kleinbaum (1994) have suggested systematic procedures to avoid calculation errors. Most rely on carefully subtracting the logits of the comparisons of interest to identify the relevant coefficients and variables needed for the estimation of the correct and final odds ratio. However, the application of the formula used to estimate the variance remains tedious and deters many researchers from describing interactions in their findings. When investigators fail to consider the testing of interactions they are implicitly, but perhaps incorrectly, assuming that all effects are constant across levels of the other variables in the model.

In this note, we propose to use a combination of traditional procedures and Stata commands to easily estimate such odds ratios. To illustrate the procedure we shall use a hypothetical model to predict coronary heart disease (chd with 0=no, 1=yes) using variables such as gender (sex; 0=females, 1=males), age, and smoking status (smk; 0=no, 1=yes). The following logit would result from the modeling process and we shall assume that coefficients are statistically significant and that the statistical assumptions are correctly met.

$$\log(\text{odds } \texttt{chd}) = \beta_0 + \beta_1 \texttt{*smk} + \beta_2 \texttt{*age} + \beta_3 \texttt{*sex} + \beta_4 \texttt{*}(\texttt{age} \times \texttt{sex})$$

If we were interested in estimating the effect of “being male” in coronary heart disease, we would have to estimate different gender odds ratios (referent category being “females”) for a series of values of age because age and gender are involved in a significant interaction. For example, the effect of being male in 30 year old subjects would traditionally be estimated by subtracting the following two logits in order to identify the correct coefficients and variables to be used in the estimation of the odds ratio:

$$\log(\text{odds } \texttt{males}) = \beta_0 + \beta_1(1) + \beta_2(30) + \beta_3(1) + \beta_4(30 \times 1)$$

$$-$$

$$\log(\text{odds } \texttt{females}) = \beta_0 + \beta_1(1) + \beta_2(30) + \beta_3(0) + \beta_4(30 \times 0)$$

$$=$$

$$\log(\text{odds } \texttt{ratio}) = \beta_3(1) + \beta_4(30 \times 1)$$

The odds ratio of coronary heart disease on 30-year-old males compared to 30-year-old females is therefore the exponential of $\beta_3 + 30\beta_4$ and the confidence interval would have to be subsequently estimated using the appropriate variance.

The procedure we propose to simplify these tasks has three steps:

1. Generate the needed interaction terms and fit the logistic regression model. For our example, we created the variable agexsex by typing gen agexsex = age * sex. We then estimate the model logit chd smk age sex agexsex. Alternatively, we could have fit the logistic regression model using Stata’s xi command. In either case, Stata will produce the coefficients corresponding to the constant, to the variables smk, age, and sex and the coefficient corresponding to the age by sex interaction (the four coefficients represented in the logit above); namely β_1 for smk, β_2 for age, β_3 for sex, β_4 for the age by sex interaction, and β_0 for the constant in the model.

2. Depending on the odds ratios of interest and using the simple procedure of subtracting logits described above, we take note of the relevant information needed for Stata to compute our odds ratios. We shall use a spreadsheet display to obtain the relevant model variables that should be used in further analyses using Stata commands and call this table the “logit table”.

Model variables	logit sex=1	Model variables	logit sex=0	Relevant variables
smk	1	smk	1	0
age	30	age	30	0
sex	1	sex	0	sex
agexsex	30 × 1	agexsex	30 × 0	30 × agexsex
				sex, 30 × agexsex

Model variables that remain constant in compared groups (smk, age) have the same values in both logits (1 and 30 for smk and age, respectively), the subtraction of logits results in the elimination of these variables and so we do not need their coefficients to estimate the odds ratio. The main odds ratio of interest is males versus females and so we replace the values 1 and 0 where appropriate. The subtraction of both logits finally results in identifying the variables sex and 30*agexsex as those relevant for estimating the odds ratio of chd in 30-year-old males compared to 30-year-old females.

3. We finally obtain the correct odds ratios and their confidence intervals with no further calculation using Stata's `lincom` command (`lincom` estimates linear combinations of coefficients after any estimation command such as `logit` or `logistic`). We would type `lincom sex + 30*agexsex`.

Step three can be repeated with as many values of age as the researcher is interested in. As an example and for graphical purposes, one may display how the odds ratio changes with ages that range from 25 to 64 years. Subsequent Stata programming commands could be used to store the results (odds ratios and their standard errors) as they are produced and they can be used for tabulation or graphical purposes (graphs are very useful and recommended to better understand interactions). The `lincom` command saves the point estimate and the estimate of the standard error in `r(estimate)` and `r(se)`, respectively.

The above procedure has the advantage of easily obtaining correct odds ratios even in models with more complex interactions. For example, suppose we also had an interaction between age and smoking status in the example described previously. The logit table would have the following display and we could easily estimate odds ratios of `chd` for males versus females in subjects 30 years of age who are smokers:

Model variables	logit `sex=1`	Model variables	logit `sex=0`	Relevant variables
`smk`	1	`smk`	1	0
`age`	30	`age`	30	0
`sex`	1	`sex`	0	`sex`
`agexsex`	30 × 1	`agexsex`	30 × 0	30 × `agexsex`
`smkxsex`	1 × 1	`smkxsex`	1 × 0	`smkxsex`
				`sex`, 30 × `agexsex`, `smkxsex`

We would then repeat the following command with different values of age (replacing 30 by other values of age):

```
. lincom sex + 30*agexsex + smkxsex
```

The same procedure could be repeated for nonsmokers.

References

Hosmer, D. W., Jr., and S. Lemeshow. 1989. *Applied Logistic Regression*. New York: John Wiley & Sons.

Kleinbaum D. G. 1994. *Logistic Regression: A Self-Learning Text*. New York: Springer-Verlag.

sg126 Two-parameter log-gamma and log-inverse Gaussian models

Joseph Hilbe, Arizona State University, jhilbe@aol.com

This insert describes the new commands `lgamma`, `gammalog`, and `ivglog` which are implementations of a full-information maximum-likelihood version of the two-parameter gamma (`lgamma` and `gammalog`) and inverse Gaussian (`ivglog`) family-log link generalized linear model with standard errors produced using the observed information matrix.

Models estimated within the framework of Generalized Linear Models (GLM) are, strictly speaking, one parameter exponential models. However, several traditional GLM models derive from distributions which have, in their base form, two parameters. When these distributions are considered within the GLM framework, their respective scale parameters are fixed as constants.

Traditional GLM models which are based on distributions having two parameters include the Gaussian, Gamma, inverse Gaussian, and negative binomial models. In addition to the `glm` implementation of the negative binomial, Stata has a full-information maximum likelihood version for the negative binomial called `nbreg`. An alternative exists which allows modeling of data which precludes counts of zero.

Stata's `ml` routine estimates models using a modified Newton–Raphson algorithm which uses the observed information matrix to calculate standard errors. The `glm` command uses an Iteratively Reweighted Least Squares (IRLS) algorithm to calculate estimates. In the process, the `IRLS` method implements the expected information matrix to produce its standard errors. When the canonical link is being modeled, no difference exists between the two methods. The standard errors are identical. However, for noncanonical links, standard errors differ; especially when there are few observations in the model. Tests have demonstrated that the use of the observed information matrix is preferable to the expected information matrix. The results are nearly the same, even when the cases are small, but the former produce more accurate standard errors. Hence, if available, use of algorithms which use the observed information matrix to produce standard errors is to be preferred.

Standard GLM algorithms may be internally adjusted so that noncanonical links still produce standard errors based on the observed information matrix. It typically necessitates two to three additional lines of code. I did this for a GLM-based negative binomial algorithm I wrote; see Hilbe 1994. That particular implementation of the negative binomial also included an iterative point estimation of the ancillary parameter. However, the simple use of Stata's `ml` capability makes the exercise much easier.

With this insert I am providing programs which use Stata's `ml` capabilities to estimate full-information maximum likelihood models for both the two-parameter log-linked gamma and the two-parameter log-linked inverse Gaussian regression models. Parameter estimates should be identical to those produced using Stata's `glm` command, e.g., `glm` with a log link. Standard errors will differ from those produced using `glm` because the log link is not the canonical link for either of the models.

The log-likelihoods for the models are

$$\log L = \frac{w}{\phi}\left(y\left(\frac{-1}{\exp(xB)}\right) - \exp(xB)\right) + \left(\frac{w}{\phi}\log(wy/\phi) - \log y - \log\Gamma(w/\phi)\right)$$

for the log-gamma, and

$$\log L = \frac{w}{\phi}\left(y\left(\frac{-1}{2\exp(xB)^2}\right) + \frac{1}{\exp(xB)}\right) - \frac{1}{2}\left(\log\left(\frac{2\pi\phi y^3}{w}\right) + \frac{w}{\phi y}\right)$$

for the inverse-Gaussian, where w is a prior weight and ϕ is a scale parameter.

Both models estimate data having a continuous response which must be greater than zero, that is, $y > 0$. They may be used to estimate most any type of positive response data. Moreover, they may also be used to estimate count response data in which there are many different values for the response. For example, when dealing with hospital length-of-stay data, one typically regards the counts as discrete. However, if there are many different lengths of stay, and the data are underdispersed or highly peaked at initial stages, then using gamma or inverse Gaussian models may be preferable.

A log-gamma response takes a variety of shapes. It is the continuous correlate of the more familiar discrete negative binomial distribution. Log inverse Gaussian responses typically have a high initial peak with a low long tail. This latter model has a long history in reliability studies, but is rarely used in other domains; perhaps without justification.

Using the new commands

All three commands have standard `ml`-type syntax with the full range of `ml` capabilities and output, see [R] **ml**. Robust standard errors, with or without clustering, may be used, together with the production of scores. The `predict` command yields standard output. An `eform` or `irr` option may be called to transform parameter estimates to incidence rate ratios. Help files are provided.

Three sets of files are associated with this insert. The first two are called **`lgamma.ado`**, **`lgamm_ll.ado`**, **`lgamma.hlp`** and **`ivglog.ado`**, **`ivgln_ll.ado`**, and **`ivglog.hlp`**, and refer to the log-gamma and inverse Gaussian distributions, while the third was written by Bill Sribney and was distributed over Statalist. It is a more sophisticated version of **`lgamma.ado`**, having been based on it. The files are called **`gammalog.ado`**, **`gamlf_lf.ado`**, and **`gammalog.hlp`**.

`lgamma` and **`gammalog`** yield identical output. The latter utilizes an analytic null likelihood. Stata users should feel free to use either of the programs.

References

Hilbe, J. 1994. sg16.5: Negative binomial regression. *Stata Technical Bulletin* 18: 2–5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 84–88.

——. 2000. *Handbook of Generalized Linear Models.* Chapman & Hall/CRC.

sg127 Summary statistics for estimation sample

Jeroen Weesie, Utrecht University, Netherlands, j.weesie@fss.uu.nl

This insert describes the simple post-estimation utility `estsumm` that displays summary statistics of the variables involved in estimating a model. Some of this information can be obtained via

```
. summarize varlist if e(sample)
```

but this requires retyping the relevant *varlist*. Also, information about clustering and weights are not easily shown.

Syntax

`estsumm` [`, eq`]

Options

`eq` specifies that the dependent variables and the independent variables in the equations are displayed equation-wise, repeating the summary information on variables entered in more than one equation.

Example

We illustrate `estsumm` using the automobile data with an outright silly example of a weighted regression analysis, with standard errors modified for clustering within the make of the car (the generation of a variable defined as the first word of a string variable may be of some independent interest to some readers).

```
. gen str10 Make = substr(make,1,index(make+" "," ")-1)
. tab Make
       Make |      Freq.     Percent        Cum.
------------+-----------------------------------
        AMC |          3        4.05        4.05
       Audi |          2        2.70        6.76
        BMW |          1        1.35        8.11
      Buick |          7        9.46       17.57
       Cad. |          3        4.05       21.62
      Chev. |          6        8.11       29.73
     Datsun |          4        5.41       35.14
      Dodge |          4        5.41       40.54
       Fiat |          1        1.35       41.89
       Ford |          2        2.70       44.59
      Honda |          2        2.70       47.30
      Linc. |          3        4.05       51.35
      Mazda |          1        1.35       52.70
      Merc. |          6        8.11       60.81
       Olds |          7        9.46       70.27
    Peugeot |          1        1.35       71.62
      Plym. |          5        6.76       78.38
      Pont. |          6        8.11       86.49
    Renault |          1        1.35       87.84
     Subaru |          1        1.35       89.19
     Toyota |          3        4.05       93.24
         VW |          4        5.41       98.65
      Volvo |          1        1.35      100.00
------------+-----------------------------------
      Total |         74      100.00
. regress price weight trunk rep if foreign [aw=1/mpg], cluster(Make)
(sum of wgt is   8.7740e-01)
Regression with robust standard errors                 Number of obs =      21
                                                       F(  3,    10) =   15.82
                                                       Prob > F      =  0.0004
                                                       R-squared     =  0.7517
Number of clusters (Make) = 11                         Root MSE      =  1275.5
------------------------------------------------------------------------------
         |               Robust
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  weight |   5.427974   .8298632      6.541   0.000       3.578923    7.277025
   trunk |   102.4539   104.6404      0.979   0.351      -130.6995    335.6072
   rep78 |  -508.2667   578.2695     -0.879   0.400      -1796.732     780.198
   _cons |  -5225.804   2274.803     -2.297   0.044      -10294.38    -157.226
------------------------------------------------------------------------------
```

(*Example continued on next page*)

```
. estsumm
Estimation sample regress                       Number of obs        =       21
                                                Number of clusters   =       11
                                                Obs per cluster: min =        1
                                                                 avg =      1.9
                                                                 max =        4

Variable |      Mean   Std. Dev.       Min        Max  Label
---------+------------------------------------------------------------------------
   price |  6070.143   2220.984       3748      11995  Price
  weight |  2263.333   364.7099       1760       3170  Weight (lbs.)
   trunk |  11.28571   3.242574          5         16  trunk space (cu.ft.)
   rep78 |  4.285714   .7171372          3          5  Repair Record 1978
---------+------------------------------------------------------------------------
__weight |  .0417809   .0097754   .0243902   .0588235  weighted by = 1/mpg
```

Since estsumm displays variable labels, the screen may become rather crowded by wrapping of the labels. Therefore, estsumm was written to be sensitive to the display linesize.

```
. set display linesize 100

. estsumm
Estimation sample regress                        Number of obs        =       21
                                                 Number of clusters   =       11
                                                 Obs per cluster: min =        1
                                                                  avg =      1.9
                                                                  max =        4

Variable |      Mean   Std. Dev.       Min        Max  Label
---------+-------------------------------------------------------------------------
   price |  6070.143   2220.984       3748      11995  Price
  weight |  2263.333   364.7099       1760       3170  Weight (lbs.)
   trunk |  11.28571   3.242574          5         16  trunk space (cu.ft.)
   rep78 |  4.285714   .7171372          3          5  Repair Record 1978
---------+-------------------------------------------------------------------------
__weight |  .0417809   .0097754   .0243902   .0588235  weighted by = 1/mpg
```

estsumm has only a single option eq that specifies that the output should be put in the multi-equation panel format. In this format, summary statistics for a variable that is included in more than one equation, is repeated in each of the associated panels. For example, for the seemingly unrelated regression model,

```
. sureg (price length trunk mpg rep) (rep for price)

Seemingly unrelated regression
------------------------------------------------------------------
Equation      Obs  Parms        RMSE    "R-sq"       Chi2        P
------------------------------------------------------------------
price          69      4    2485.445    0.2610   29.66034   0.0000
rep78          69      2    .7979384    0.3407   41.15192   0.0000
------------------------------------------------------------------------------
         |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
price    |
  length |   42.88136   26.49298      1.619   0.106      -9.043919    94.80665
   trunk |  -33.11848   101.5098     -0.326   0.744       -232.074    165.8371
     mpg |  -163.3025   86.59608     -1.886   0.059      -333.0277    6.422679
   rep78 |   1035.671   332.6344      3.114   0.002       383.7195    1687.622
   _cons |  -1517.422   5973.718     -0.254   0.799      -13225.69    10190.85
---------+--------------------------------------------------------------------
rep78    |
 foreign |   1.301522   .2061204      6.314   0.000       .8975334     1.70551
   price |   .0000398   .0000327      1.219   0.223      -.0000242    .0001039
   _cons |   2.764767   .2318625     11.924   0.000       2.310325    3.219209
------------------------------------------------------------------------------
```

we can display summary statistics of the estimation sample as

(*Example continued on next page*)

```
. estsumm, eq
Estimation sample sureg                         Number of obs        =       69
Variable |      Mean   Std. Dev.       Min        Max  Label
---------+---------------------------------------------------------------------
depvar   |
   price |  6146.043    2912.44       3291      15906  Price
   rep78 |  3.405797   .9899323          1          5  Repair Record 1978
---------+---------------------------------------------------------------------
price    |
  length |  188.2899    22.7474        142        233  Length (in.)
   trunk |  13.92754   4.343077          5         23  trunk space (cu.ft.)
     mpg |  21.28986   5.866408         12         41  Mileage (mpg)
   rep78 |  3.405797   .9899323          1          5  Repair Record 1978
 foreign |  .3043478   .4635016          0          1  Car type
---------+---------------------------------------------------------------------
rep78    |
   price |  6146.043    2912.44       3291      15906  Price
```

Remark

The command `estsumm` belongs to the library ICSLib, a collection of (primarily) Stata commands developed at the ICS, an interuniversity graduate school of social science theory and methods, located at the sociology departments of the universities of Groningen, Utrecht, and Nijmegen in the Netherlands. ICSLib is written largely by Jeroen Weesie and the late Albert Verbeek, with contributions by other researchers in the ICS. A large part of the ICSLib is available for downloading at the URL http://www.fss.uu.nl/soc/iscore/stata. Consult this site for updates of `estsumm` and other commands from ICSLib submitted to the STB over the last couple of years.

sg128	Some programs for growth estimation in fisheries biology

Isaías Hazarmabeth Salgado-Ugarte, Juana Martínez-Ramírez, José Luis Gómez-Márquez, and Bertha Peña-Mendoza.
FES Zaragoza, UNAM Biología, Mexico, isalgado@servidor.unam.mx

Introduction

It has become generally accepted that the purpose of fisheries management is to ensure sustainable production over time from fish populations (stocks) through regulatory and enhancement actions without forgetting the promotion of economic and social wellbeing of the fishermen and related industries. To achieve this purpose, management authorities must design, justify politically, and administer (enforce) a collection of restraints on fishing activity (Hilborn and Walters 1992). These decisions must rely on biological knowledge of the exploited populations and this activity is synonymous with stock assessment. It is necessary to consider that once the assessment is complete, the choice among several management options may produce the same biological yield. Finally, according to Hilborn and Walters (1992), stock assessment should provide estimates about the tradeoff between average yield and its variability, making the choice of management alternatives on social and economic grounds.

The knowledge of fish age is one important biological characteristic that provides very useful information during the early development of a commercial fishery. Fish that live a long time normally provide large yields at the beginning of a fishery because of fishing down of the older age classes (Hilborn and Walters 1992).

The study of growth is the determination of the body size as a function of age. To assess the aquatic fisheries resources several methods using age composition data have been developed (Sparre and Venema 1992).

In this insert we present programs to perform several procedures designed to estimate the parameters of the von Bertalanffy growth function. Three of them are linear derivations from the basic growth equation, and the last one uses a nonlinear regression approach. The programs are applied to data coming from hard parts reading and length frequency analysis described below carried out by means of the application of kernel density estimation (KDE). The multimodal distributions are analyzed by means of an updated version of the semigraphical determination of Gaussian components presented in *sg23* in STB-18. This contribution intends to introduce an alternative way to analyze length frequency data (by nonparametric and computing-intensive procedures) in conjunction with automated versions of the traditional methods to obtain growth expressions from the estimated sizes. The same computerized algorithms can be applied to ages estimated by fish hard parts reading.

Age estimation by length frequency analysis

When it is possible to measure the size of a large number of individuals sampled from a fish or invertebrate stock, the distribution of these sizes may be analyzed. Traditionally, the histogram or frequency polygon have been employed for this purpose, but recently, kernel density estimation has been proposed as being the statistical procedure best suited to analyze distributions of animal size (Salgado-Ugarte et al. 1993, 1995a, 1995b, 1997). If the reproductive events of the species is of a discrete nature, it would be possible to follow the dominant size groups or classes as modes in the size distribution. If the reproduction occurs at regular time intervals, it may be feasible to approximate ages to the various size groups (King 1995).

With the powerful set of procedures associated to the KDEs such as those to choose the best bandwidth, such as the practical rules (optimal and oversmoothed), cross-validation (least squares and biased), and the bootstrap test for multimodality, the number of groups can be estimated. Once a multimodal distribution has been obtained, the individual components in this mixed distribution can be determined by using for example Bhattacharya's method. These modes represent groups of fish with similar ages (cohorts). With the means estimated for each cohort, and assuming or knowing the time period separating them, it is possible to estimate the parameters of the VBGF.

Age estimation by hard parts reading

To estimate the age of fish from temperate waters it is possible to count year rings formed on hard parts such as scales and otoliths. The rings are formed due to environmental seasonal fluctuations. In tropical locations, where the seasonal changes are small, it could be very difficult to find seasonal marks in hard parts (Sparre and Venema 1992). Nevertheless, in these environmental uniform areas the hard parts show daily increments, in other cases periodic marks are formed by biological events such as reproduction. The best compromise for stock assessment of tropical species is therefore an analysis of a large number of length-frequency data combined with a small number of age readings on the basis of periodic marks, daily rings or reproduction marks (Sparre and Venema 1992, King 1995).

von Bertalanffy growth function parameter estimation

To describe quantitatively the growth of fish, several researchers developed mathematical expressions. In fisheries biology the main model was proposed by von Bertalanffy (1938). The VBGF has been very useful as it follows closely the observed growth of most fish species. Since then, several growth models have appeared in the literature, for example, Beverton and Holt (1957), Ursin (1968), Ricker (1975) Gulland (1983), Pauly (1984), and Pauly and Morgan (1987). The VBGF has become one of the cornerstones in fishery biology because it is used as a submodel in more complex models describing the dynamics of fish populations (Sparre and Venema 1992).

The von Bertalanffy mathematical model expresses the length L as a function of the age of the fish t:

$$L_t = L_\infty(1 - \exp(-K(t - t_0)))$$

The right-hand side of this equation contains the age t and three parameters L_∞, K, and t_0. Different species would have a particular set of parameters. A biological interpretation for the parameters has been provided; L_∞ is "the mean length of very old (strictly speaking infinitely old) fish." It is also called the "asymptotic length." K is a "curvature parameter" which determines how fast the fish approaches its L_∞. A high value for K indicates a short-lived species almost reaching their L_∞ in a year or two. A low K value results in a flat growth curve for species that need many years to approach their L_∞. The third parameter, t_0, sometimes called "the initial condition parameter," determines the hypothetical point in time when the fish has zero length. Of course, biologically this has no meaning, because at hatching (when the growth begins) the larva already has a certain length. This length at hatch may be called L_0 when we put $t = 0$ at the day of birth

$$L_0 = L_\infty(1 - \exp(-Kt_0))$$

However, L_0 may not be a realistic estimate of the length at birth because fish larvae would follow a different growth model as the egg development is affected by different factors during its development. Nevertheless, the larger (exploited) fish usually do follow the VBGF (Sparre and Venema 1992).

Ford–Walford graph

Using the single sample method, often called the "Petersen method," a growth curve can be estimated from the relative position of the modes in a single length-frequency sample. The crucial assumption is that the modes are equal time intervals apart, typically one year. One of the simplest methods of estimating the parameters of the VBGF for data representing equal time intervals is by means of a Ford–Walford plot (see Ford 1933, Walford 1946). The derivation of this plot is based on the growth curve with t_0 equal to zero:

$$\begin{aligned}
L_t &= L_\infty(1 - \exp(-Kt)) & (1)\\
L_t &= L_\infty - L_\infty \exp(-Kt) &\\
L_\infty - L_t &= L_\infty \exp(-Kt) & (2)
\end{aligned}$$

Subtracting L_{t+1} from equation (1) we have

$$\begin{aligned}
L_{t+1} - L_t &= L_\infty(1 - \exp(-K(t+1))) - L_\infty(1 - \exp(-Kt)) &\\
&= -L_\infty \exp(-K(t+1)) + L_\infty \exp(-Kt) &\\
&= -L_\infty \exp(-Kt)(1 - \exp(-K)) & (3)
\end{aligned}$$

Substituting (2) in (3) gives

$$\begin{aligned} L_{t+1} - L_t &= (L_\infty - L_t)(1 - \exp(-K)) \\ &= L_\infty(1 - \exp(-K)) - L_t + L_t \exp(-K) \\ L_{t+1} &= L_\infty(1 - \exp(-K)) + L_t \exp(-K) \end{aligned}$$

This is a linear equation, and suggests that length at age t, L_t, can be plotted against length at age one year later L_{t+1}. The straight line fitting these data will have a slope of $b = \exp(-K)$ and an intercept of $a = L_\infty(1 - \exp(-K))$. In this way it is possible to estimate K and L_∞ as (Gómez-Márquez 1994, King 1995):

$$K = -\log(b), \qquad L_\infty = a/(1 - b)$$

Gulland plot

This method is a variation of the Ford–Walford plot and was proposed by Chapman (1961) and later by Gulland (1969). It is based on the use of a constant time interval Δt. It can be shown that the VBGF implies that

$$L_{t+\Delta t} - L_t = L_\infty(1 - \exp(-K\Delta t)) - L_t(1 - \exp(-K\Delta t))$$

Thus, since K and L_∞ are constants, $(1 - \exp(-K\Delta t))$ will remain constant and the equation becomes a linear function $y = a + bx$, where

$$a = L_\infty(1 - \exp(-K\Delta t)), \qquad b = -(1 - \exp(-K\Delta t))$$

The growth parameters are derived from Sparre and Venema (1992):

$$K = -(1/\Delta t)\log(1 + b), \qquad L_\infty = -a/b = a/(1 - \exp(-K\Delta t))$$

Gulland–Holt plot

This procedure, proposed by Gulland and Holt (1959), takes into account that fish increase in length as they grow older, but their "growth rate" (increment in length per time unit), decreases with age approaching zero in very old fish. Taking the definition of the growth rate

$$\frac{\Delta L}{\Delta t} = \frac{L_{t+\Delta t} - L_t}{\Delta t}$$

and considering that the mathematical relationship between the length of a fish and the growth rate at a given time is a linear function, we have

$$\frac{\Delta L}{\Delta t} = a + bL_t$$

Using the VBGF we have

$$\frac{\Delta L}{\Delta t} = K(L_\infty - L_t) \tag{4}$$

Equation (4) can be rewritten as

$$\frac{\Delta L}{\Delta t} = KL_\infty - K\overline{L}_t \tag{5}$$

where $\overline{L}_t = (L_{t+\Delta t} - L_t)/2$. When Δt is small, $\overline{L}_t$ may be a reasonable approximation to the mean length. An advantage over other methods is that Δt does not need to be a constant. Using $\overline{L}_t$ as the independent variable and $\Delta L/\Delta t$ as the response variable, equation (5) becomes a linear relationship allowing the estimation of parameters be performed by regression.

$$\frac{\Delta L}{\Delta t} = a + b\overline{L}_t$$

where $K = -b$ and $L_\infty = -a/b$ (Sparre and Venema 1992, Gómez-Márquez 1994).

Beverton-Holt

This method, due to Beverton and Holt (1957), is used to estimate K and t_0 for a given L_∞ which may come from one of the above presented methods. From the VBGF we can obtain

$$L_\infty - L_t = L_\infty \exp(-K(t - t_0)), \qquad \log(L_\infty - L_t) = \log L_\infty - Kt + Kt_0$$

Rearranging, we finally have

$$\log(L_\infty - L_t) = \log L_\infty + Kt_0 - Kt$$

which is a linear function of t versus $\log(L_\infty - L_t)$ and the parameters are recovered by (see Gómez-Márquez 1994):

$$K = -b, \qquad a = \log L_\infty + Kt_0$$

Solving for t_0, we have

$$t_0 = \frac{a - \log L_\infty}{K}$$

Nonlinear regression

According to Sparre and Venema (1992), a nonlinear least squares regression is a superior method to the procedures described so far to estimate the parameters of the VBGF from a theoretical and statistical point of view. The computational work is considerable. The method estimates the growth parameters in such a way that the sum of squares of the deviations between the model and the observations is minimized, that is, it minimizes the sum with respect to the parameters L_∞, K, and t_0.

Syntax

`bhataplt` *freqvar midpoivar* [`if` *exp*] [`in` *range*] [, `gen(`*logdivar*`)` `nograph` *graph_options*]

`bhatgauc` *freqvar midpoivar* [`if` *exp*] [`in` *range*] [, `gen(`*gaucovar*`)` `nograph` *graph_options*]

`fordwal` *msizevar* [`if` *exp*] [`in` *range*] [, `nograph` *graph_options*]

`gullplot` *msizevar* [`if` *exp*] [`in` *range*] [, `nograph` *graph_options*]

`gullholt` *msizevar agevar* [`if` *exp*] [`in` *range*] [, `nograph` *graph_options*]

`bevholt` *msizevar agevar* [`if` *exp*] [`in` *range*] , `linf(`*#*`)` [`gen(`*estsizevar*`)` `nograph` *graph_options*]

`nl vbgf` *sizevar agevar* [*weight*] [`if` *exp*] [`in` *range*] [, *nl_options*]

`fweight`s and `aweight`s are allowed in `nl vbgf`.

Description

`bhataplt` calculates from the frequency and midpoint pair values in the *freqvar* and *midpoivar* variables, the logarithmic differences and draws the Bhattacharya plot using the observation numbers as plotting symbols in order to define negatively sloped lines, each one representing individual Gaussian components in mixed distributions. `bhataplt` is a new version integrating the two previous simple programs `diflogen` (logarithmic differences generator) and `bhatplot` (Bhattacharya's plot drawer) which were presented in *sg23* in STB-18.

`bhatgauc` calculates the parameters (mean and standard deviation) of the dominant Gaussian component in a specified range. *freqvar* is the frequency variable, and *midpoivar* is the midclass interval variable. `bhatgauc` displays also the graphical comparison of the observed frequencies and the estimated Gaussian component. This routine is a new version of the previous simple program `bhatmesd` in *sg23* in STB-18 but includes more versatile options.

`fordwal` estimates the L_∞ and K parameters of the VBGF using mean-at-age data in the variable *msizevar* according to the procedure proposed by Ford (1933) and Walford (1946). `fordwal` draws the l_t versus l_{t+1} graph and provides a table with the numerical results.

`gullplot` estimates the L_∞ and K parameters of the VBGF using mean-at-age data in *msizevar* according to the procedure proposed by Chapman (1961) and later by Gulland (1969). `gullplot` draws the l_t versus $l_{t+1} - l_t$ plot and provides a table with the numerical results.

`gullholt` estimates the L_∞ and K parameters of the VBGF using mean-size (length) at-age data in *msizevar* and an age variable in *agevar* according to the procedure proposed by Gulland and Holt (1959), draws the linear relationship used for the estimation, and provides a table with the estimated parameters of the VBGF.

`bevholt` estimates the K and t_0 (for a given L_∞) parameters of the VBGF using mean-size (length) at-age data in *msizevar* and an age variable in *agevar* according to the procedure proposed by Beverton and Holt (1957), draws the $\log(L_\infty - l_t)$ age graph, and provides a table with the estimated parameters in addition to the estimated VBGF.

`nl vgbf` uses Stata's `nl` program to fit a VBGF to the dependent variable *sizevar* which is size or mean size as a function of the age variable *agevar* by least squares. The VBGF is in the separate program `nlvbgf`. As an application of the `nl` procedure included in Stata, the `nl vbgf` program shares all the options for `nl`.

Options

`nograph` suppresses displaying the graph.

graph_options are any of the options allowed with `graph, twoway`; see [G] **graph options**.

`gen(`*logdivar*`)` creates the variable *logdivar* containing the logarithmic differences of frequencies.

`gen(`*gaucovar*`)` creates the variable *gaucovar* containing the frequencies of the estimated Gaussian component.

`gen(`*estsizevar*`)` creates the variable *estsizevar* containing the estimated sizes of the growth function determined.

`linf(`*#*`)` is not optional. The L_∞ value can be retrieved from `fordwal` or `gullplot`.

nl_options are the options in Stata's `nl` command.

Case 1. Length frequency analysis

To illustrate the analysis of length frequency, we use the data from catfish length of 641 fish collected in November, 1980 from a coastal lagoon of Mexico reported in Salgado-Ugarte (1985). A brief summary is provided by

```
. use catfilen
. tab sex, sum(bodlen)
            |         Summary of bodlen
        sex |        Mean   Std. Dev.       Freq.
------------+------------------------------------
          1 |    164.9633   34.039173         109
          2 |   184.38554   36.332419          83
          3 |    75.77951   9.6996377         449
------------+------------------------------------
      Total |    105.0078     49.5904         641
```

The length data were subjected to the Silverman test for multimodality using the programs included in Salgado-Ugarte, et al. (1997) obtaining the results in Table 1.

Table 1. Critical bandwidths and estimated significance levels for catfish length data, $n = 641$.

Number of modes	Critical bandwidth	p-values
1	23.36	0.0000
2	19.43	0.0000
3	9.64	0.1750
4	3.88	0.7330
5	3.23	0.7750
6	3.09	0.6000

Notice that the p-values were obtained from $B = 120$ bootstrap replications of size 641.

These data clearly have four modes. Because four modes can be obtained from 9.63 to 3.88 we use an intermediate bandwidth value for the density estimation $(9.63 + 3.88)/2 \approx 6.7$:

```
. warpdenm bodlen, b(6.7) m(10) k(6) gen(den6p7 mid6p7)
```

To transform the density (`den6p7`) to frequency values we have

```
. gen sumden=sum(den6p7)
. gen freq=den6p7*641/sumden[_N]
```

Then we apply the Bhattacharya method of Gaussian decomposition (Bhattacharya 1967). To achieve this, we employ the following updated versions of the programs included in Salgado-Ugarte et al. (1994). The program `bhataplt` calculates from frequency and midpoint pair values, the logarithmic differences and draws the Bhattacharya plot using the observations numbers as plotting symbols in order to define negatively sloped lines, each one representing individual gaussian components in mixed distributions.

It requires frequency and midpoint variables being possible to generate the logarithmic differences variable if desired. We can adjust a Gaussian distribution to the observed frequencies employing `bhatgauc`.

This program calculates the parameters (mean and standard deviation) of the dominant Gaussian component in a specified range. For the first component we used we have

```
. bhatgauc freq mid in 13/22
R-squared = 0.9945          Adj R-squared = 0.9939
Mean = 75.6083
s.d. = 11.9029
component size = 441
```

The results are summarized in Table 2 and displayed graphically in Figure 1.

Table 2. Estimated Gaussian components with parameters.

Component	Mean	SD	Size
1	75.6083	11.9029	441
2	137.5868	9.7123	75
3	174.8262	11.9622	42
4	215.9292	11.8713	66

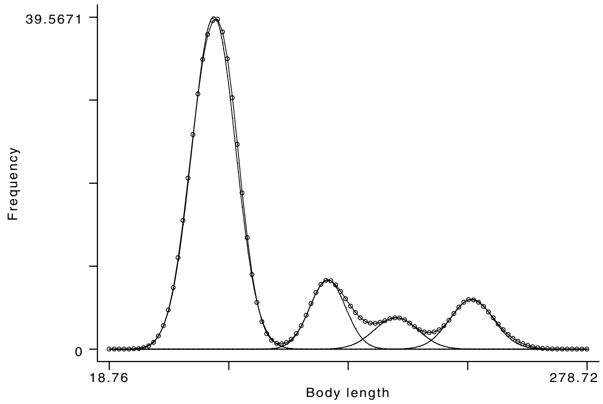


Figure 1. Observed smoothed frequency (from KDE) and estimated Gaussian components for catfish data.

The mean values are used to estimate the VBGF by preparing a file with the mean values only.

```
. input mblen
          mblen
  1. 75.6083
  2. 137.5868
  3. 174.8262
  4. 215.9292
  5. end
```

We can construct the Ford–Walford plot via `fordwal` which estimates the L_∞ and K parameters of the VBGF using mean-at-age data according to the procedure proposed by Ford (1933) and Walford (1946), draws the l_t versus l_{t+1} graph (Figure 2), and provides a table with the numerical results.

```
. fordwal m
Estimation of L_inf and K values by the Ford-Walford Method
------------------------------------------------------------
Intercept  =  76.4705          Slope        =   0.7704
R-squared  =   0.9709        Adj R-squared  =   0.9418
L_inf.     = 333.0560          K            =   0.2608
------------------------------------------------------------
```

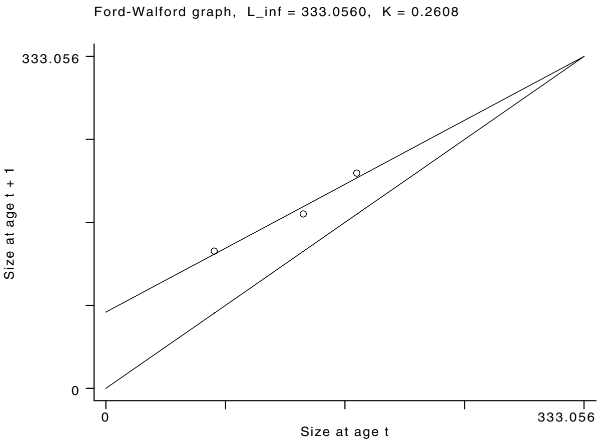


Figure 2. Ford–Walford plot for the mean values estimated by the Bhattacharya method for catfish data.

The same estimation for L_∞ can be obtained with the Gulland plot given by **gullpolt**. This program estimates the L_∞ and K parameters of the VBGF using mean-at-age data according to the procedure proposed by Chapman (1961) and later by Gulland (1969), draws the l_t versus $l_{t+1} - l_t$ graph (Figure 3) and provides a table with the numerical results.

```
. gullplot mblen

Estimation of L_inf and K values by the Gulland Method
------------------------------------------------------------
Intercept  =  76.4705         Slope         =  -0.2296

R-squared  =   0.7477         Adj R-squared =   0.4953

L_inf.     = 333.0560         K             =   0.2608
------------------------------------------------------------
```

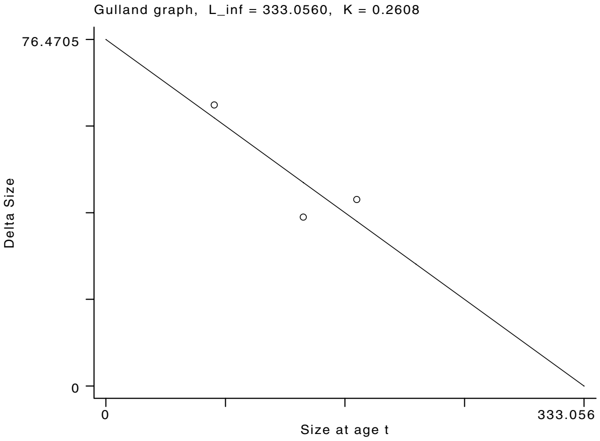


Figure 3. Gulland plot for the mean values estimated by the Bhattacharya method for catfish data.

Each mode represents a group of fish with similar age (cohorts). Assuming yearly cohorts and an initial age of one, we can generate an age variable by typing **gen age = _n**. With this "age" variable we can employ the Gulland–Holt method for VBGF parameter estimation via the **gullholt** program. This program estimates the L_∞ and K parameters of the VBGF using mean size (length) at age data according to the procedure proposed by Gulland and Holt (1959). It draws the linear relationship used for the estimation (Figure 4) and provides a table with the estimated L_∞ and K parameters of the von Bertalanffy growth function. Applied to the catfish data we have

```
. gullholt mblen age

Estimation of L_inf and K values by the Gulland-Holt Method
------------------------------------------------------------
Intercept  =  84.4403          Slope         =  -0.2466

R-squared  =   0.6798          Adj R-squared =   0.3596

L_inf.     = 342.3808          K             =   0.2466
------------------------------------------------------------
```

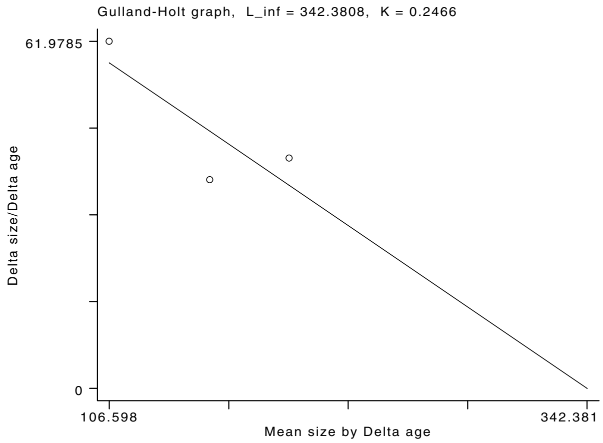


Figure 4. Gulland–Holt plot for VBGF parameter estimation for catfish data.

With the estimation of L_∞ and assuming an annual age we can get the estimation for t_0 with the Beverton and Holt method given in the bevholt program. bevholt estimates K and t_0 for a given L_∞ using mean size (length) at age data according to the procedure proposed by Beverton and Holt (1957). It draws the $\log(L_\infty - L_t)$ versus age graph (Figure 5) and provides a table with the estimated K and t_0 parameters in addition to the estimated VBGF. Figure 6 shows the adjusted curve with the observed points as well as an indication of the asymptotic length.

```
. bevholt mblen age, linf(333.06) gen(mblest)
Estimation of K and t_0 values by the Beverton-Holt Method
------------------------------------------------------------
Intercept  =   5.8069         Slope         =  -0.2574
R-squared  =   0.9960         Adj R-squared =   0.9940
K          =   0.2574         t_0           =  -0.0055
------------------------------------------------------------
Estimated von Bertalanffy Growth Function
------------------------------------------------------------
l_t = 333.0600 * (1 - exp( -0.2574 *(t +  0.0055 )))
------------------------------------------------------------
```

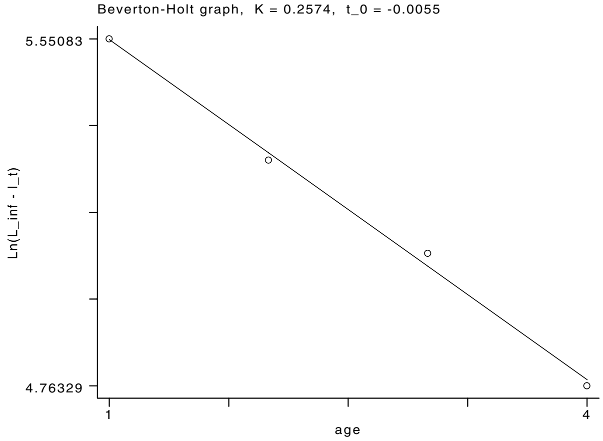


Figure 5. Beverton–Holt plot for estimating the K and t_0 parameters of the VBGF for the catfish data.

(Figure 6 on next page)

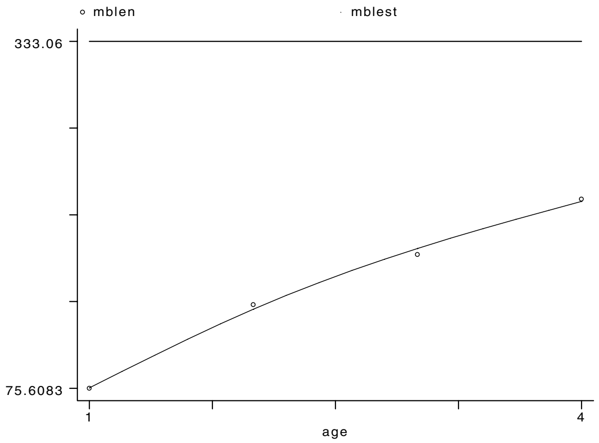


Figure 6. Observed mean-at-age points and estimated VBGF curve for the catfish data.

We can employ a nonlinear regression approach by using `nlvbgf.ado`, which uses Stata's `nl` program to fit a VBGF to the dependent variable which is size (individual or mean value) versus age by least squares. The VBGF is in the separate program `nlvbgf` which sets the objective function and the parameters. The initial values in `nlvbgf` are 500, 1, and 0.01 for L_∞, K, and t_0, respectively. As an application of the nl procedure included in Stata, the `nl vbgf` program shares all the options for `nl`.

```
. nl vbgf mblen age
  (output omitted )
  Source |       SS       df       MS                  Number of obs =        4
---------+------------------------------               F(  3,     1) =   998.30
   Model |  101802.373     3  33934.1242               Prob > F      =   0.0233
Residual |  33.9918952     1  33.9918952               R-squared     =   0.9997
---------+------------------------------               Adj R-squared =   0.9987
   Total |  101836.365     4  25459.0911               Root MSE      = 5.830257
                                                       Res. dev.     = 19.91082
von Bertalanffy growth function, mblen=b0*(1-exp(-b1*(age-b2)))
------------------------------------------------------------------------------
   mblen |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      b0 |   342.2139   100.7661      3.396   0.182      -938.1413    1622.569
      b1 |   .2435344   .1303093      1.869   0.313      -1.412203    1.899272
      b2 |  -.0406671    .272883     -0.149   0.906      -3.507975     3.42664
------------------------------------------------------------------------------
 (SE´s, P values, CI´s, and correlations are asymptotic approximations)
```

As we can see, the estimated parameters are almost identical, but the nonlinear approach gives additional information such as confidence intervals and t-values for the parameters and their probabilities.

Case 2. Length at age data from hard part reading

To illustrate the application of these programs to estimated age data we use the Japanese sea bass "suzuki" (Lateolabrax japonicus) data from Salgado-Ugarte (1995). From these we present the results of estimated age from scale reading during the spring months (beginning of the growth season).

Table 3. Number of individuals by sampling date and estimated age.

	Sample number					
Age	6	7	8	9	Total	(mean)
0	14	3	0	0	17	(168.18)
1	4	4	2	1	11	(257.10)
2	0	5	2	7	14	(361.29)
3	0	2	0	2	4	(462.00)
11	0	0	1	0	1	(760.00)
Total	18	14	5	10	47	

Notice that the mean body length by age is included in the totals column, and that samples numbers correspond to the following dates (1994): 6 = 19/02; 7 = 16/03; 8 = 14/04; 9 = 13/05.

This time, having age data we try the nonlinear approach first:

```
. drop _all

. input mlen age

          mlen        age
  1. 168.18 0
  2. 257.1 1
  3. 361.29 2
  4. 462 3
  5. 760 11
  6. end

. save suzmada

. nl vbgf mlen age
  (output omitted)
  Source |       SS       df       MS                 Number of obs =         5
---------+------------------------------              F(  3,     2) =   1183.25
   Model |   1015387.3     3  338462.435              Prob > F      =    0.0008
Residual |  572.088616     2  286.044308              R-squared     =    0.9994
---------+------------------------------              Adj R-squared =    0.9986
   Total |  1015959.39     5  203191.879              Root MSE      =  16.91284
                                                      Res. dev.     =  37.88867
von Bertalanffy growth function, mlen=b0*(1-exp(-b1*(age-b2)))
------------------------------------------------------------------------------
    mlen |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      b0 |    862.244   44.04351     19.577   0.003       672.7401    1051.748
      b1 |   .1766637   .0257078      6.872   0.021       .0660518    .2872756
      b2 |  -1.150208   .2169301     -5.302   0.034      -2.083583   -.2168328
------------------------------------------------------------------------------
 (SE´s, P values, CI´s, and correlations are asymptotic approximations)
```

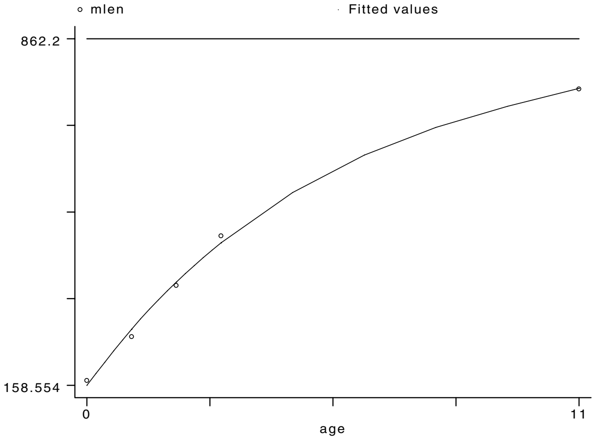


Figure 7. Nonlinear regression VBGF fit for “suzuki” mean length at age data.

The time interval from the estimated ages is not uniform. Thus we can use the Gulland–Holt method to try an alternative fit:

```
. gullholt mlen age

Estimation of L_inf and K values by the Gulland-Holt Method
------------------------------------------------------------
Intercept  = 138.0650         Slope         =  -0.1432

R-squared  =   0.6190         Adj R-squared =   0.4285

L_inf.     = 964.0516         K             =   0.1432
------------------------------------------------------------
```

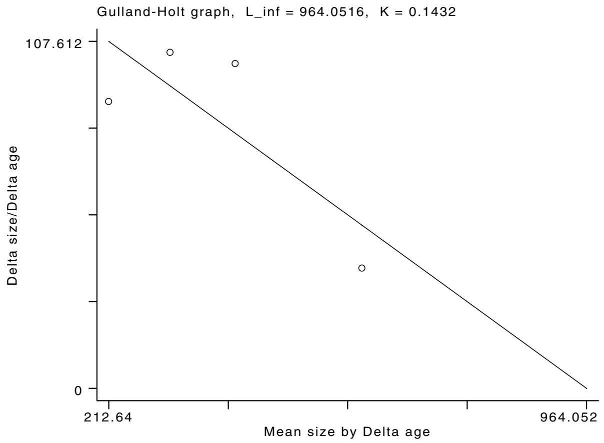


Figure 8. Gulland–Holt plot for “suzuki” mean length at age data.

Using this L_∞ value in the Beverton–Holt procedure we arrive at

```
. bevholt mlen age, l(964) gen(mleghbh)
Estimation of K and t_0 values by the Beverton-Holt Method
------------------------------------------------------------
Intercept  =   6.6535         Slope         =  -0.1229
R-squared  =   0.9946         Adj R-squared =   0.9929
K          =   0.1229         t_0           =  -1.7702
------------------------------------------------------------
Estimated von Bertalanffy Growth Function
------------------------------------------------------------
l_t = 964.0000 * (1 - exp( -0.1229 *(t +  1.7702 )))
------------------------------------------------------------
```

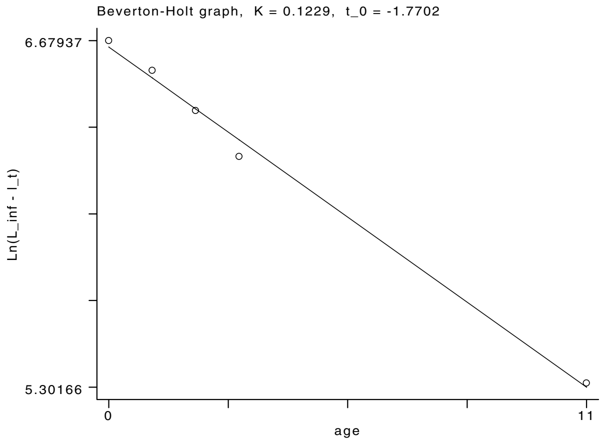


Figure 9. Beverton–Holt graph to estimate the K and t_0 parameters of the VBGF for the suzuki mean length at age data.

(*Figure 10 on next page*)

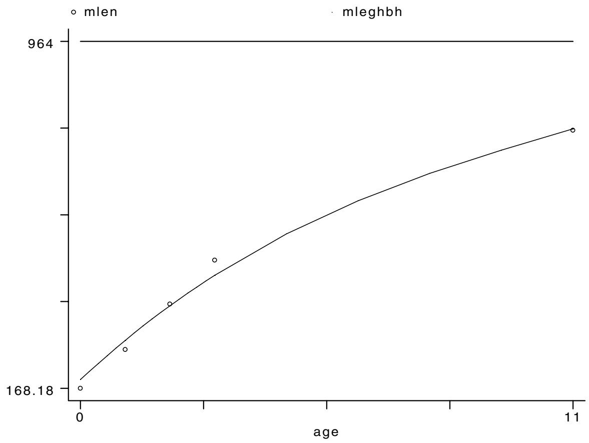


Figure 10. Alternative VBGF fitting to the "suzuki" mean length at age data.

Apparently, the nonlinear regression approach fits closer the observed mean length at age values.

We have found that these programs, though simple, provide a useful set of tools (saving a lot of time and effort) for fisheries data analysis including classical and modern approaches. The growth expressions determined with the assistance of these programs and additional biological information may be used as input for more complex simulation algorithms such as those from Hilborn and Walters (1992) or King (1995). We invite users to try our programs and let us know about any experience they may have.

Final notes

There are several other computerized procedures to estimate growth from age estimated by length-frequency analysis or hard part reading. In particular, we can mention those included in the FAO–ICLARM Stock Assessment Tools. Each of the included programs has the corresponding help file.

Acknowledgments

The authors are grateful to the Universidad Nacional Autónoma de México (FES Zaragoza and DGAPA) for their support through the DGAPA–PAPIIT project code IN217596. The second author received a scholarship from Fundación UNAM.

References

Beverton, R. J. H. and S. J. Holt. 1957. On the dynamics of the exploited fish populations. In *Fisheries Investigation of Ministry of Agriculture and Fisheries and Food Great Britain (2 Sea Fisheries)*, vol. 19.

Bhattacharya, C. G. 1967. A simple method of resolution of a distribution into Gaussian components. *Biometrics* 23: 115–135.

Chapman, D. G. 1961. Statistical problems in dynamics of exploited fisheries populations. *Proceedings 4th Berkeley Symposium on Mathematics, Statistics and Probability. Cont. Biol. and Prob. Med.* 4: 153–168.

Ford, E. 1933. An account of the herring investigations conducted at Plymouth during the years from 1924–1933. *Journal of Marine Biology Assessment* 19: 305–384.

Gómez-Márquez, J. L. 1994. Métodos para determinar la edad en los organismos acuáticos. FES Zaragoza UNAM Mexico. (In Spanish).

Gulland, J. A. 1969. *Manual of Methods for Fish Stock Assessment, Part I. Fish Population Analysis.* FAO Manual Fisheries Science.

——. 1983. *Fish Stock Assessment; a Manual of Basic Methods*. Chichester, UK: Wiley Interscience.

Gulland, J. A. and S. J. Holt. 1959. Estimation of growth parameters for data at unequal time intervals. *Journal Counseil CIEM* 25(1): 47–49.

Hilborn, R. and C. J. Walters. 1992. *Quantitative Fisheries Stock Assessment, Dynamics and Uncertainty.* London: Chapman & Hall.

King, M. 1995. *Fisheries Biology, Assessment and Management.* Fishing News Books.

Pauly, D. 1984. *Fish Population Dynamics in Tropical Waters: A Manual for use with Programmable Calculators.* ICLARM Studies and Reviews.

Pauly, D. and G. R. Morgan. 1987. *Length-Based Methods in Fisheries Research.* ICLARM Conference Proceedings.

Ricker, W. E. 1975. Computation and interpretation of biological statistics of fish populations. *Bulletin of the Fisheries Research Board of Canada* 191: 382.

Salgado-Ugarte, I. H. 1985. Algunos aspectos biológicos del bagre Arius melanopus Gűnther (Osteichthyes: Ariidae) en el sistema lagunar de Tampamachoco. Ver. Bachelor Thesis dissertation, ENEP Zaragoza, Universidad Nacional Autónoma de Mexico. (In Spanish).

——. 1995. Nonparametric methods for fisheries data analysis and their application in conjunction with other statistical techniques to study biological data of the Japanese sea bass Lateolabrax japonicus in Tokyo Bay. PhD Thesis dissertation, Graduate School of Agricultural and Life Sciences, The University of Tokyo.

Salgado-Ugarte, I. H., M. Shimizu, and T. Taniuchi. 1993. snp6: Exploring the shape of univariate data using kernel density estimators. *Stata Technical*

Bulletin 16: 8–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 155–173.

——. 1994. sg23: Semi-graphical determination of Gaussian components in mixed distributions. *Stata Technical Bulletin* 18: 15–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 127–146.

——. 1995a. snp6.1: ASH, WARPing, and kernel density estimation for univariate data. *Stata Technical Bulletin* 26: 23–31. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 161–172.

——. 1995b: snp6.2: Practical rules for bandwidth selection in univariate density estimation. Stata Technical Bulletin 27: 5–19. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 172–190.

——. 1997. snp13: Nonparametric assessment of multimodality for univariate data. *Stata Technical Bulletin* 38: 27–35. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 232–243.

Sparre, P. and S. C. Venema. 1992. Introduction to tropical fish stock assessment. FAO Fisheries Technical Paper No. 306.1.

Ursin, E. 1968. A mathematical model of some aspects of fish growth, respiration and mortality. *Journal of the Fisheries Research Board of Canada* 24: 2355–2453.

Von Bertalanffy, L. 1938. A quantitative theory of organic growth. *Human Biology* 10: 181–243.

Walford, L. A. 1946. A new graphic method of describing the growth of animals. *Biology Bulletin 90(2)*: 141–147.

sg129 Generalized linear latent and mixed models

Sophia Rabe-Hesketh, Institute of Psychiatry, King's College London, UK, spaksrh@iop.kcl.ac.uk
Andrew Pickles, University of Manchester, UK, andrew.pickles@man.ac.uk
Colin Taylor, Institute of Psychiatry, King's College London, UK, colin.taylor@iop.kcl.ac.uk

This insert describes the command `gllamm6` which fits generalized linear latent and mixed models (GLLAMM). These models include a large variety of latent variable models, e.g., multilevel, item-response, ordered latent class, and error-in-covariate models. We assume that the dataset is hierarchical with level-one units nested within level-two units that are nested within level-three units, and so on, a typical example being pupils in classes in schools. However, the level-one units may also be different variables observed on the same unit. Goldstein (1995) represents multivariate responses in this way and the approach is not unusual if the variables are repeated measures.

The GLLAMM models are defined by the conditional densities of the observed response variables given the latent and explanatory variables and the hierarchical structure and distribution of the latent variables.

The conditional densities of the responses given the latent and explanatory variables are generalized linear models with linear predictors given by

$$\eta = \sum_{l=1}^{L} \beta_l x_l + \sum_{n=2}^{N} \sum_{k=0}^{K^{(n)}-1} u_k^{(n)} \sum_{l=1}^{L_k^{(n)}} \lambda_{lk}^{(n)} x_{lk}^{(n)}$$

where x_l is the lth explanatory variable associated with the "fixed" effect β_l and $u_k^{(n)}$ is the kth random effect (or latent variable) at level n. Each random effect is multiplied by a linear combination of $L_k^{(n)}$ explanatory variables $x_{lk}^{(n)}$ with coefficients $\lambda_{lk}^{(n)}$. In order for the model to be identified, the coefficient $\lambda_{1k}^{(n)}$ of the first explanatory variable for each random effect is fixed at 1 and the (co)variances of the random effects are freely estimated. The latent variables have zero means.

Here we have omitted subscripts denoting membership of the observations in the clusters or levels of the hierarchy to simplify the notation required for the general n-level model. For a two-level model, we can use subscripts i and j to index the level-2 and level-1 units, respectively, giving

$$\eta_{ij} = \sum_{l=1}^{L} \beta_l x_{lij} + \sum_{k=0}^{K^{(2)}-1} u_{ki}^{(2)} \sum_{l=1}^{L_k^{(2)}} \lambda_{lk}^{(2)} x_{lkij}^{(2)} \qquad (1)$$

The conditional densities of the responses are specified by this linear predictor together with the link and family. Most of the links and families available for Stata's `glm` command are also available for `gllamm6`. If the level-1 units are different variables, different links and families may be specified for different "observations".

The structure of the latent variables is specified by the number of levels N (and the variables defining these levels, e.g., pupil, school, and so on) and the number of random effects at each level. Random effects at the same level are assumed to be correlated with each other whereas random effects at different levels are assumed to be independent.

The latent variables at a level-n may be assumed to have a multivariate normal distribution. Alternatively, the latent variables may be assumed to be discrete, having nonzero probability on a finite number of points (of dimensionality equal to the number of latent variables, $K^{(n)}$, at level n). If the number of points, or masses, is chosen to maximize the likelihood, the nonparametric maximum likelihood estimator (NPML) can be achieved (Lindsay et al. 1991).

The simplest example of a generalized linear latent and mixed model is a two-level random intercept model given by

$$\eta_{ij} = \sum_{l=1}^{L} \beta_l x_{lij} + u_i$$

where u_i is assumed to have a normal distribution with zero mean (we will drop subscripts and superscripts wherever possible). This model may be fitted using Stata's `xt` functions, for example `xtreg`, `xtlogit` and `xtpois`. The program `gllamm6` allows five extensions to this model, (1) multilevel models, (2) random coefficients, (3) discrete random effects, (4) factor loadings, and (5) mixed responses. These extensions are illustrated by the following models that correspond to the worked examples presented later. (The examples do not exhaust the list of possible model types.)

Three-level model

Subjects j in families i have measurements on several occasions k. The measurements y_{ijk} may be modeled by a generalized linear mixed model with linear predictor

$$\eta_{ijk} = \beta_0 + \beta_1 x_{1ijk} + \cdots + u_{ij}^{(2)} + u_i^{(3)} \tag{2}$$

where the random effects $u_{ij}^{(2)}$ for subject j in family i and $u_i^{(3)}$ for family i are assumed to be independently normally distributed.

Random coefficient model

Pupils j in schools i take an exam at two time points. The performance at time 2 may be modeled as a linear regression on the result at time 1, where both the slopes and intercepts differ between schools. The mean performance at time 2 is therefore modeled as

$$\eta_{ij} = \beta_0 + \beta_1 x_{ij} + u_{0i} + u_{1i} x_{ij} \tag{3}$$

Here x_{ij} is the time 1 result and u_{1i} is the corresponding random coefficient. The two random effects u_0 and u_1 are assumed to have a bivariate normal distribution.

Discrete random effects

Instead of assuming a bivariate normal distribution of the random effects in the above example, we may assume a bivariate discrete distribution, i.e., we assume that the random effects u_0, u_1 take on a number of discrete values z_{0r}, z_{1r}, with probabilities π_r, $r = 1, \ldots, R$. This corresponds to assuming that the population falls into a finite number of latent classes or types or can be approximated in this way. When the maximum number of classes is used, the distribution may be interpreted as a nonparametric distribution.

Item response model (factor loadings)

Item response models may be used to model the (binary) responses of subjects to a number of exam questions, or items. The log odds of subject i giving a correct answer to item j may be modeled using a Rasch model as

$$\eta_{ij} = \beta_j + u_i \tag{4}$$

where $-\beta_j$ represents the difficulty of question j and u_i represents the ability of subject i. This is a simple two-level model and may be fitted using `xtlogit`. If we introduce a further parameter λ_j, we obtain a two parameter Rasch model

$$\eta_{ij} = \beta_j + u_i \lambda_j$$

where λ_j represents the extent to which question j discriminates between subjects of different abilities. Here we are modeling a multivariate dataset by using the second index j for different variables. If the data are in long form with the responses to all the questions stacked into a single response vector, we can use dummy variables

$$x_{lij} = \begin{cases} 1, & \text{if } l = j \\ 0, & \text{otherwise} \end{cases}$$

to write the model in the form of Equation (1), giving

$$\eta_{ij} = \sum_l \beta_l x_{lij} + u_i \sum \lambda_l x_{lij} \tag{5}$$

Errors in covariates model (mixed responses)

Consider the problem of modeling how disease status depends on an explanatory variable which is subject to measurement error. This exposure may have been measured a number of times on a subset of subjects to estimate its reliability. We therefore have a number of responses per subject, disease status, y_{i1}, and one or more measures of exposure y_{i2}, $y_{i3}, \ldots$

We assume that the exposure measurements y_{i2} and y_{i3}, etc., are independently normally distributed with means given by

$$\eta_{ij} = \mu_{ij} = \gamma_0 + f_i, \quad j = 2, 3, \ldots \tag{6}$$

where f_i is a latent variable representing the difference between subject ith's exposure and the mean exposure γ_0 in the population. This difference in exposure may depend on a covariate, for example, age, as follows

$$f_i = \gamma_1 + \beta_1 x_i + u_i \tag{7}$$

where u_i is another latent variable. We now specify a disease model by assuming that y_{i1} is binomial with the logit of the probability π_{i1} given by

$$\eta_{i1} = \text{logit}(\pi_{i1}) = \gamma_2 + \gamma_3 x_i + \lambda f_i \tag{8}$$

This model is shown in the path diagram below (where boxes denote observed variables, circles denote latent variables and vectors point from explanatory variables to dependent variables):

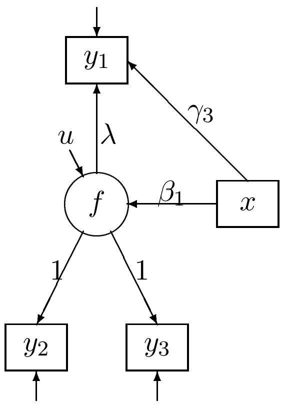


Figure 1. Path diagram for an errors in covariates model.

Substituting the model for f_i in (7) into the measurement and disease models (6) and (8) gives

$$\eta_{ij} = \beta_0 + \beta_1 x_i + u_i, \quad j = 2, 3, \ldots$$

where $\beta_0 = \gamma_0 + \gamma_1$, and

$$\eta_{i1} = \beta_2 + \beta_3 x_i + \lambda u_i$$

where $\beta_2 = \lambda\gamma_1 + \gamma_2$ and $\beta_3 = \lambda\beta_1 + \gamma_3$.

These two models can be written as a single mixed response model in the form of Equation (1) by using dummy variables E and D for "observations" corresponding to exposure and disease status, respectively:

$$\eta_{ij} = \beta_0 E_{ij} + \beta_1 E_{ij} x_i + \beta_2 D_{ij} + \beta_3 D_{ij} x_i + u_i(E_{ij} + \lambda D_{ij}) \quad j = 1, 2, 3 \tag{9}$$

Implementation

The program uses `ml` with method `d0` to maximize the likelihood. For a 2-level model, the likelihood is given by

$$\prod_i \int \{\prod_j f(y_{ij}|\mathbf{x}_i,\mathbf{u}_i)\} g(\mathbf{u}_i) d\mathbf{u}_i \tag{10}$$

where $f(y_{ij}|\mathbf{x}_i,\mathbf{u}_i)$ is the conditional density of the response variable given the latent and explanatory variables and $g(\mathbf{u}_i)$ is the prior density of the latent variables. When the latent variables are discrete, the integral becomes a sum of the form

$$\prod_i \sum_r \pi_r \prod_j f(y_{ij}|\mathbf{x}_i,\mathbf{u}_i=\mathbf{z}_r)$$

where the locations $\mathbf{z}_r$ and masses π_r are freely estimated. For a single normally distributed latent variable, the same expression is used to approximate the likelihood, where locations and masses are given by Gaussian quadrature.

If there are $K^{(n)} > 1$ correlated (multivariate normal) latent variables, at level n, we express them as a linear combination of uncorrelated random effects $\mathbf{v}$, $\mathbf{u} = \mathbf{L}\mathbf{v}$ where $\mathbf{L}$ is a lower triangular matrix. The multiple integral is then approximated by summing over $\mathbf{v} = \mathbf{z}_r$ using $K^{(n)}$ nested sums. The elements of $\mathbf{L}$ are estimated by `gllamm6` and the covariance matrix of the random effects is given by $\mathbf{L}'\mathbf{L}$ so that $\mathbf{L}$ is simply the Cholesky decomposition of the covariance matrix.

In (10), the contribution to the likelihood from the ith level-2 unit is found by integrating the product of the contributions from the level-1 units inside the level-2 unit over the level-2 random effects distribution. In a three-level model, the contribution from a 3-level unit is found by integrating the product of contributions from the level-2 units inside the level-3 unit over the level-3 random effects distribution. The likelihood for an n-level model is therefore computed using a recursive algorithm.

The parameters are transformed to ensure that they lie within their permitted ranges. For the normal (or gamma) density, the log of the standard deviation (or coefficient of variation) at level 1 is estimated to ensure a positive estimate on the natural scale. When quadrature is used, the Cholesky decomposition of the covariance matrix of the random effects at each level is estimated to ensure a positive definite covariance matrix. When there are no correlations, this corresponds to estimating the standard deviations directly. The sign of these estimates is arbitrary. When R discrete mass-points are specified for the random effects at a level, $R-1$ log odds are estimated to give the R probabilities and $R-1$ locations are estimated directly for each random effect. The last location is determined by constraining the mean of the discrete distribution to zero. The variance of the random effects distribution is not estimated directly but follows from the locations and masses. Approximate standard errors for the back-transformed estimates are obtained using the delta method (except for the variance of the discrete random effects distributions).

Syntax

`gllamm6` [*varlist*] [`if` *exp*] [`in` *range*] `,` `i(`*varlist*`)` [`nrf(`*#*`,...,`*#*`)` `eqs(`*eqnames*`)` `nocorrel`

`noconstant` `offset(`*varname*`)` `weight(`*varname*`)` `family(`*families*`)` `fv(`*varname*`)` `denom(`*varname*`)`

`link(`*links*`)` `lv(`*varname*`)` `s(`*eqname*`)` `ip(`*str*`)` `nip(`*#*`,...,`*#*`)` `from(`*matrix*`)` `lf0(`*##*`)` `gateaux(`*###*`)`

`search(`*#*`)` *maximize_options* `nodifficult` `level(`*#*`)` `eform` `allc` `trace` `nolog` `noest`]

The data must be in long form with all responses stacked into a single variable.

Options

`i(`*varlist*`)` gives the variables that define the hierarchical, nested clusters, from the lowest level (finest clusters) to the highest level, e.g., `i(pupil class school)`.

`nrf(`*#*`,...,`*#*`)` specifies the number of random effects for each level, i.e., for each variable in `i(`*varlist*`)`. The default is `nrf(1,...,1)`.

`eqs(`*eqnames*`)` specifies the equation names (defined before running `gllamm6`, see `help eq`) for the linear predictors multiplying the latent variables. If required, constants should be explicitly included in the equation definitions using variables equal to 1. If the option is not used, the latent variables are assumed to be random intercepts and only one random effect is allowed per level. The first lambda coefficient is set to one. The other coefficients are estimated together with the (co)variance(s) of the random effect(s).

`nocorrel` may be used to constrain all correlations to zero if there are several random effects at any of the levels and if these are modeled as multivariate normal.

`noconstant` omits the constant term from the fixed effects equation.

`offset`(*varname*) specifies a variable to be added to the linear predictor without estimating a corresponding regression coefficient (e.g., log exposure for Poisson regression).

`weight`(*wt*) specifies that variables *wt1*, *wt2*, and so on, contain frequency weights. The suffixes determine at what level each weight applies. For example, if the level-1 units are subjects, the level-2 units are families, and the result is binary, we can collapse dataset A into dataset B as follows:

```
A                                B
family subject result            family subject result  wt1  wt2
  1       1      0                 1       1      0      2    1
  1       2      0                 2       3      1      1    2
  2       3      1                 2       4      0      1    2
  2       4      0
  3       5      1
  3       6      0
```

The level-1 weight, `wt1`, indicates that subject 1 in dataset B represents two subjects within family 1 in dataset A, whereas subjects 3 and 4 in dataset B represent single subjects within family 2 in dataset A. The level 2 weight `wt2` indicates that family 1 in dataset B represents one family and family 2 represents two families, i.e., all the data for family 2 are replicated once. Collapsing the data in this way can make `gllamm6` run considerably faster.

`family`(*families*) specifies the families to be used for the conditional densities. The default is `family(gaussian)`. Also available are `binomial`, `poisson` and `gamma`. Several families may be given, in which case the variable allocating families to observations must be given using `fv`(*varname*).

`fv`(*varname*) is required if mixed responses requiring more than a single family of conditional distributions are analyzed. The variable indicates which family applies to which observation. A value of one refers to the first family, and so on.

`denom`(*varname*) gives the variable containing the binomial denominator for the responses whose family is specified as binomial. The default denominator is 1.

`link`(*links*) specifies the links to be used for the conditional densities (`identity, logit, probit, log, reciprocal`). If a single family is specified, the default link is the canonical link. Several links may be given in which case the variable allocating links to observations must be given using `lv`(*varname*). Numerically feasible choices of link depend upon the distributions of the covariates and the choice of conditional error and random effects distributions.

`lv`(*varname*) is the variable whose values indicate which link applies to which observation.

`s`(*eqname*) specifies that the log of the standard deviation (or of the coefficient of variation) at level 1 for normally (or gamma) distributed responses should be given by the linear predictor defined by `eqname`. This is necessary if the level-1 variance is heteroscedastic. For example, if dummy variables for groups are used, different variances are estimated for different groups.

`ip`(*string*) if *string* is '`g`', Gaussian quadrature points are used and if *string* is '`f`', the mass-points are freely estimated. The default is Gaussian quadrature.

`nip(`*#,. . .,#*`)` specifies the number of integration points or masses to be used for each integral or summation. When quadrature is used, a value may be given for each random effect. When freely estimated masses are used, a value may be given for each level of the model. If only one argument is given, the same number of integration points will be used for each summation.

`from`(*matrix*) specifies the matrix to be used for the initial values. Note that the column names and equation names have to be correct (see `help matrix`). The matrix may be obtained from a previous estimation command using `e(b)`. This is useful if another explanatory variable needs to be added or the number of masses needs to be increased.

`lf0`(*# #*) gives the number of parameters and the log-likelihood for a likelihood ratio test to compare the model to be estimated with a simpler model. A likelihood ratio chi-squared test is only performed if the `lf0()` option is used.

`gateaux`(*# # #*) may be used with method `ip(f)` to increase the number of mass points by one from a previous solution with parameter estimates specified using `from(matrix)` and number of parameters and log likelihood specified by `lf0(`*# #*`)`. The program searches for the location of the new mass-point by placing a very small mass at the location given by the first argument and moving it to the second argument in the number of steps specified by the third argument. (If there are several random effects, this search is done in each dimension resulting in a regular grid of search points.) If the maximum increase in likelihood is greater than 0, the location corresponding to this maximum is used as the initial value of the new location, otherwise the program stops. When this happens, it can be shown that for certain models the current solution represents the nonparametric maximum likelihood estimate.

`search(#)` causes the program to search for initial values for the random effects at level 2 (in range 0 to 3). The argument specifies the number of random searches. This option may only be used with `ip(g)` and when `from(matrix)` is not used.

maximize_options control the maximization process; see [R] **maximize**. One useful option is `iterate(#)` since by default the program does not limit the number of iterations. The `skip` option can be used if the matrix of initial parameter estimates specified in `from(matrix)` has extra parameters.

`nodifficult` causes `ml` not to use the difficult option, see [R] **maximize**.

`level(#)` specifies the confidence level in percent for confidence intervals of the coefficients.

`eform` causes the exponentiated coefficients and confidence intervals to be displayed.

`allc` causes all estimated parameters to be displayed in a regression table (including the raw random effects parameters) in addition to the usual output.

`trace` is one of the *maximize_options*, see [R] **maximize**. In addition to displaying the details of the maximum likelihood iterations, it displays details of the model being fitted.

`nolog` suppresses output for maximum likelihood iterations.

`noest` is used to prevent the program from carrying out the estimation. This may be used with the `trace` option to check that the model is correct and get the information needed to set up a matrix of initial values. Global macros are available that are normally deleted. Particularly useful may be `M_initf` and `M_initr`, matrices for the parameters (fixed part and random part, respectively).

Example 1: Three-level model

Three groups of subjects, some of whom were related to each other, completed a neuropsychological test with three levels of difficulty, resulting in 3 binary responses per subject (Rabe-Hesketh et al. 1999). These responses are stacked into a variable called `dtlm`. The variables `id`, `famnum` and `group` are the subject, family, and group identifiers respectively and `level` codes the levels of difficulty. A listing of these variables for observations 19 to 27 is given below.

```
. list id famnum group level dtlm in 19/27
             id     famnum      group      level       dtlm
 19.          7          6          1          0          0
 20.          7          6          1          1          0
 21.          7          6          1         -1          1
 22.          8         15          1          0          0
 23.          8         15          1          1          0
 24.          8         15          1         -1          1
 25.          9         10          1         -1          0
 26.          9         10          1          0          0
 27.          9         10          1          1          0
```

A two-level model with a random effect for subjects could be fitted using `xtlogit` with the following syntax:

```
xi: xtlogit dtlm i.group level, i(id) quad(20)
```

The syntax for the same model using `gllamm6` is

```
xi: gllamm6 dtlm i.group level, i(id) fam(binom) link(logit) nip(20)
```

Introducing a random effect for families gives a three-level logistic regression model as in Equation (2) that may be fitted as follows:

```
. xi: gllamm6 dtlm i.group level, i(id famnum) fam(binom) link(logit) nip(8)
gllamm model
log likelihood = -305.11873
------------------------------------------------------------------------------
    dtlm |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
Igroup_2 |      -.249   .3544871     -0.702   0.482       -.943782     .445782
Igroup_3 |  -1.052334   .3999883     -2.631   0.009      -1.836297   -.2683717
   level |  -1.648503   .1932191     -8.532   0.000      -2.027206   -1.269801
   _cons |  -1.485952   .2848916     -5.216   0.000      -2.044329   -.9275745
------------------------------------------------------------------------------
```

```
Variances and covariances of random effects
-----------------------------------------------------------------------------

***level 2 (id)

    var(1): 1.1345171 (.68948028)

***level 3 (famnum)

    var(1): .57347062 (.53124401)
-----------------------------------------------------------------------------
```

The variance at level 2 is estimated as 1.134 with a standard error of 0.689, and the variance at level 3 is estimated as 0.573 with a standard error of 0.531. The output only shows the final results of the estimation. Since `gllamm6` can take a while and we would like to make sure it's doing what we intend, we usually use the `trace` option which gives output on how the model is parameterized, as well as a full iteration log. To check if 8 points were sufficient, we can use `matrix a=e(b)` followed by the command above with the options `nip(20)` and `from(a)`. Some of the parameter estimates change in the third decimal place. Increasing the number of quadrature points by another 10 gives negligible changes. Issuing the command `gllamm6, eform` gives the same output as above but with exponentiated parameters and confidence intervals in the fixed-effects table.

Example 2: Random coefficient model

We will now analyze the Junior School Project data from the MLN manual (1995). Math results are available on pupils from different schools in the third and fifth years. We will fit a linear regression model of the year-5 results, `math5`, on the (mean centered) year-3 results, `math3`, with a random intercept and a random coefficient of `math3` for schools (see Equation (3)). A listing of the variables for observations 87 to 95 is shown below.

```
. list school pupil math5 math3 wt1 in 87/95
        school      pupil      math5      math3           wt1
 87.         5         21         28        3.6             1
 88.         5         22         30       -3.4             1
 89.         5         23         25       -3.4             1
 90.         5         24         37        6.6             2
 91.         5         25         36        1.6             1
 92.         6          1         28       -5.4             1
 93.         6          2         26        4.6             1
 94.         6          3         30       -6.4             1
 95.         6          4         37        5.6             1
```

The weight variable `wt1` is 1 for most pupils because there were only a few instances of two pupils in the same school having the same result for `math3` and `math5`.

Initially, we will assume zero correlation between the random slope and intercept by using the `nocor` option.

```
. gen cons = 1

. eq sch_c: cons

. eq sch_m3: math3

. gllamm6 math5 math3, i(school) nrf(2) eqs(sch_c sch_m3) nocor nip(8) w(wt)

gllamm model

log likelihood = -2763.3492
------------------------------------------------------------------------------
   math5 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   math3 |   .6137155   .0443395     13.841   0.000       .5268117    .7006192
   _cons |   30.68167   .2952554    103.916   0.000       30.10298    31.26036
------------------------------------------------------------------------------

Variance at level 1
-----------------------------------------------------------------------------

  26.935556 (1.357001)

Variances and covariances of random effects
-----------------------------------------------------------------------------

***level 2 (school)

    var(1): 4.0987663 (.99349649)
  cov(1,2): fixed at 0
    var(2): .03747035 (.01981001)
-----------------------------------------------------------------------------
```

The equations `sch_c` and `sch_m3` specify that, in the linear predictor, the first random effect is multiplied by the constant 1 and the second random effect is multiplied by `math3`, so that the random effects represent a random intercept and a random slope, respectively. The within-school (residual) variance is estimated as 26.94 with a standard error of 1.36, the random intercept variance is estimated as 4.10 with a standard error of 0.99, and the random slope variance is estimated as 0.037 with a standard error of 0.020.

We can now use these estimates as initial estimates for a model that allows the random intercept and slope to be correlated:

```
. matrix a=e(b)
. gllamm6 math5 math3, i(school) nrf(2) eqs(sch_c sch_m3) nip(8) w(wt) from(a)
gllamm model
log likelihood = -2757.0046
------------------------------------------------------------------------------
   math5 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   math3 |   .6073703   .0424769     14.299   0.000       .5241172    .6906234
   _cons |   30.65957   .3311764     92.578   0.000       30.01047    31.30866
------------------------------------------------------------------------------
Variance at level 1
-----------------------------------------------------------------------------
  26.934402 (1.3527458)
Variances and covariances of random effects
-----------------------------------------------------------------------------
***level 2 (school)
    var(1): 4.146341 (.95033595)
  cov(1,2): -.31583363 (.09461675) cor(1,2): -.86904153
    var(2): .03185449 (.01655591)
-----------------------------------------------------------------------------
```

The intercept and slope are highly negatively correlated (correlation is −0.869). Here the Cholesky decomposition of the covariance matrix, **L**, was estimated and the covariance matrix and corresponding standard errors were computed from **L** and its standard errors. To view the "raw" parameters, use the command `gllamm6, allc`.

Example 3: Discrete random effects

We will use the data from Example 2 and fit a random coefficient model with discrete random effects in two dimensions (intercept, slope) using the `ip(f)` option.

```
. eq sch_c: cons
. eq sch_m3: math3
. gllamm6 math5 math3, i(school) nrf(2) eqs(sch_c sch_m3) nip(3) w(wt) ip(f)
gllamm model
log likelihood = -2753.4705
------------------------------------------------------------------------------
   math5 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   math3 |   .6071688    .040136     15.128   0.000       .5285037    .6858338
   _cons |   30.62458   .3660507     83.662   0.000       29.90714    31.34203
------------------------------------------------------------------------------
Variance at level 1
-----------------------------------------------------------------------------
  27.002647 (1.3044172)
Probabilities and locations of random effects
-----------------------------------------------------------------------------
***level 2 (school)
    prob: 0.539, 0.1851, 0.2759
    loc1: -.32601, -3.4409, 2.9461
  var(1): 4.643575
cov(1,2): -.3318127
    loc2: .07632, .16678, -.26104
  var(2): .02708799
-----------------------------------------------------------------------------
```

The coordinates of the three points have intercepts and slopes of (−0.326, 0.076),(−3.441, 0.167), and (2.946, −0.261), with probabilities of 0.539, 0.185, and 0.276, respectively.

We can use the Gateaux-derivative method to check if introduction of a further mass point yields a larger maximized likelihood. We need to move a small mass through a fine 2D grid of values of the random effects and check whether this increases the likelihood anywhere. We can do this using the `gateaux()` option to specify the limits and number of steps for the search in each dimension. In addition, we have to pass the current likelihood to `gllamm6` using the `lf0()` option. After finding the maximum Gateaux derivative point, the estimation of the extended model automatically starts.

```
. matrix a=e(b)

. local ll=e(ll)

. local k=e(k)

. gllamm6 math5 math3, i(school) nrf(2) eqs(sch_c sch_m3) nip(4) w(wt) ip(f)
>   from(a) gateaux(-10 10 30) lf0(`k´ `ll´)

Gateaux derivative

maximum gateaux derivative is .54030611

gllamm model                                         Number of obs   =        887
                                                     LR chi2(3)      =       4.62
Log likelihood = -2751.1611                          Prob > chi2     =     0.2019

------------------------------------------------------------------------------
   math5 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   math3 |   .6169166   .0457258     13.492   0.000       .5272957    .7065375
   _cons |   30.65183   .3611123     84.882   0.000       29.94406     31.3596
------------------------------------------------------------------------------

Variance at level 1
-----------------------------------------------------------------------------
  26.644576 (1.2922189)

Probabilities and locations of random effects
-----------------------------------------------------------------------------

***level 2 (school)
    prob: 0.5334, 0.1597, 0.0316, 0.2753

    loc1: -.34238, -3.3806, -2.6291, 2.9257
  var(1): 4.4623696
cov(1,2): -.3474582

    loc2: .06313, .08057, .88985, -.27123
  var(2): .04845184
-----------------------------------------------------------------------------
```

A very small mass of 0.032 has been placed at (−2.629, 0.889) without affecting the other masses substantially.

Example 4: Item-response models

Data from five multiple choice questions of Section 6 of the Law School Admission Test (LSAT, Bock and Lieberman 1970) with the binary responses being correct/incorrect, will be used to illustrate how item-response models may be fitted using `gllamm6`. The responses are stacked into the variable `resp` and the variables `i1` to `i5` are indicators for the five items. Here we list some of these variables for observations 1 to 10.

```
. list id resp wt2 i1 i2 i3 in 1/10

            id      resp       wt2        i1        i2        i3
  1.         1         0         3         1         0         0
  2.         1         0         3         0         1         0
  3.         1         0         3         0         0         1
  4.         1         0         3         0         0         0
  5.         1         0         3         0         0         0
  6.         2         0         6         1         0         0
  7.         2         0         6         0         1         0
  8.         2         0         6         0         0         1
  9.         2         0         6         0         0         0
 10.         2         1         6         0         0         0
```

The variable wt2 is a level-2 weight and gives the number of subjects with the same response pattern across the five items. A simple Rasch (or one parameter logistic) model (see Equation (4)) may be fitted using

```
. gllamm6 resp i1 i2 i3 i4 i5, nocons link(logit) fam(bin) i(id) nip(10) w(wt)
gllamm model
log likelihood = -2466.9376
------------------------------------------------------------------------------
    resp |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      i1 |   2.730013    .130441     20.929   0.000       2.474353    2.985673
      i2 |   .9986051   .0791772     12.612   0.000       .8434207     1.15379
      i3 |   .2398536   .0717746      3.342   0.001       .0991779    .3805292
      i4 |    1.30645    .084638     15.436   0.000       1.140563    1.472338
      i5 |   2.099404   .1054449     19.910   0.000       1.892736    2.306072
------------------------------------------------------------------------------
Variances and covariances of random effects
-----------------------------------------------------------------------------
***level 2 (id)
    var(1): .57022544 (.10486119)
-----------------------------------------------------------------------------
```

(Note that the same model may be fitted using `xtlogit resp i1-i5, nocons i(id) quad(10)` if the data are not in “collapsed” form.)

A two-parameter item-response model (see Equation (5)) may be fitted as follows:

```
. eq id: i1 i2 i3 i4 i5
. gllamm6 resp i1 i2 i3 i4 i5, nocons link(logit) fam(bin) i(id)/*
>  */ eqs(id) nip(10) w(wt) lf0(6 -2466.9376252)
gllamm model                                       Number of obs   =       5000
                                                   LR chi2(4)      =       0.57
Log likelihood = -2466.6534                        Prob > chi2     =     0.9665
------------------------------------------------------------------------------
    resp |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      i1 |   2.773177   .2056814     13.483   0.000       2.370049    3.176305
      i2 |   .9902013   .0900178     11.000   0.000       .8137696    1.166633
      i3 |   .2491494   .0762736      3.267   0.001       .0996559    .3986429
      i4 |   1.284759   .0990366     12.973   0.000       1.090651    1.478868
      i5 |   2.053274   .1353578     15.169   0.000       1.787978     2.31857
------------------------------------------------------------------------------
Variances and covariances of random effects
-----------------------------------------------------------------------------
***level 2 (id)
    var(1): .68158076 (.42601869)
    loadings for random effect 1
    i2: .87541533 (.36267279)
    i3: 1.0790901 (.43509454)
    i4: .83378413 (.36723665)
    i5: .79561351 (.3805886)
-----------------------------------------------------------------------------
```

Using the `lf0()` option caused the likelihood ratio test to be shown. The two-parameter model does not fit the data significantly better than the one-parameter model. The loading of item 1 was constrained to 1 and the variance of the random effect was estimated freely. To obtain the parameters of the model where the standard deviation is constrained to 1 instead, we can interpret the standard deviation $\sqrt{.68158076}$ as the first loading and multiply all other loadings by this value.

Example 5: Errors in covariate model

An epidemiological dataset with variables on diet and coronary heart disease (CHD) (Morris et al. 1977) will be used to illustrate how the program may be used for logistic regression with errors in covariates. The aim is to estimate the relationship between fiber intake and risk of CHD where fiber is subject to measurement error and has been measured twice on a subset of subjects. The variable `resp` contains the responses for CHD and fiber and the variables `diet` and `chd` indicate whether the observation in `resp` is fiber or whether it is CHD status, respectively. The variable `var` is 1 for diet measurements and 2 for CHD measurements. A variable `age`, centered around 50, was multiplied by `chd` and `diet` to obtain `agec` and `aged`, respectively. Some of the variables are listed below for observations 425 to 437.

```
. list id resp diet var agec aged in 425/437, nolab
              id       resp      diet        var       agec       aged
425.         217   21.32756         1          1          0      -3.43
426.         217          0         0          2      -3.43          0
427.         218   23.10387         1          1          0      -7.21
428.         218          0         0          2      -7.21          0
429.         219   15.64263         1          1          0      -8.13
430.         219          0         0          2      -8.13          0
431.         219   14.87973         1          1          0      -8.13
432.         220   13.46374         1          1          0      -1.13
433.         220          0         0          2      -1.13          0
434.         220   14.73168         1          1          0      -1.13
435.         221   15.95863         1          1          0       3.67
436.         221          0         0          2       3.67          0
437.         221   17.28778         1          1          0       3.67
```

The model is given in Equation (9) where the dummy variables E and D are represented by `diet` (exposure) and `chd` (disease) and Ex and Dx by `aged` and `agec`, respectively. The model is similar to the two-parameter Rasch model in that we have a factor loading λ. The coefficient of `diet` (E_{ij}) in the linear term $(E_{ij} + \lambda D_{ij})$ multiplying the latent variable is 1 and we therefore specify `diet` first in the equation for the latent variable (the first lambda is always constrained to 1).

```
. eq id: diet chd

. gllamm6 resp diet t2 chd agec aged, nocons i(id) eqs(id) link(ident logit) /*
>          */ fam(gauss binom) lv(var) fv(var) nip(20)

gllamm model

log likelihood = -1379.1556
------------------------------------------------------------------------------
    resp |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    diet |   17.21291   .3073353     56.007   0.000       16.61054    17.81528
     chd |   -1.97982   .1894402    -10.451   0.000      -2.351116   -1.608524
    agec |   .0250153   .0236923      1.056   0.291      -.0214207    .0714512
    aged |   -.102406   .0444538     -2.304   0.021      -.1895339   -.0152781
------------------------------------------------------------------------------

Variance at level 1
-----------------------------------------------------------------------------
  7.1729268 (1.1464247)

Variances and covariances of random effects
-----------------------------------------------------------------------------

***level 2 (id)

    var(1): 24.504317 (2.6165604)

    loadings for random effect 1
    chd: -.12655812 (.04962148)

-----------------------------------------------------------------------------
```

After adjusting for age, the measurement error variance is 7.17 and the variance of latent exposure is 24.50, giving a reliability of 0.77. The odds ratio of CHD for unit increase in true fiber intake is given by

```
. disp exp(-.12655812)
.88112294
```

A semiparametric mixture model may also be fitted to this dataset using `gllamm6` (see Rabe-Hesketh and Pickles, 1999).

References

Bock, R. D. and M. Lieberman. 1970. Fitting a response model for n dichotomously scored items. *Psychometrika* 33: 179–197.

Goldstein, H. 1995. *Multilevel Statistical Models.* 2d ed. Maryland: Arnold.

Lindsay, B. G., C. C. Clogg, and J. Grego. 1991. Semiparametric estimation in the Rasch model and related exponential response models, including a simple latent class model for item analysis. *Journal of American Statistical Association* 86: 96–107.

Morris, J. N., J. W. Marr, and D. G. Clayton. 1977. Diet and heart: postscript. *British Medical Journal* 2: 1307–1314.

Rabe-Hesketh, S. and A. Pickles. 1999. Correcting for measurement error using a nonparametric exposure distribution and non-normal errors. Submitted for publication.

Rabe-Hesketh, S., T. Touloupulou, and R. M. Murray. 1999. Multilevel modeling of cognitive function in schizophrenics and their first degree relatives. *Multivariate Behavioral Research.* In press.

STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:			
an	announcements	*ip*	instruction on programming
cc	communications & letters	*os*	operating system, hardware, & interprogram communication
dm	data management		
dt	datasets	*qs*	questions and suggestions
gr	graphics	*tt*	teaching
in	instruction	*zz*	not elsewhere classified
Statistical Categories:			
sbe	biostatistics & epidemiology	*ssa*	survival analysis
sed	exploratory data analysis	*ssi*	simulation & random numbers
sg	general statistics	*sss*	social science & psychometrics
smv	multivariate analysis	*sts*	time-series, econometrics
snp	nonparametric methods	*svy*	survey sampling
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified
srd	robust methods & statistical diagnostics		

In addition, we have granted one other prefix, *stata*, to the manufacturers of Stata for their exclusive use.

Guidelines for authors

The Stata Technical Bulletin (STB) is a journal that is intended to provide a forum for Stata users of all disciplines and levels of sophistication. The STB contains articles written by StataCorp, Stata users, and others.

Articles include new Stata commands (ado-files), programming tutorials, illustrations of data analysis techniques, discussions on teaching statistics, debates on appropriate statistical techniques, reports on other programs, and interesting datasets, announcements, questions, and suggestions.

A submission to the STB consists of

1. An insert (article) describing the purpose of the submission. The STB is produced using plain TeX so submissions using TeX (or LaTeX) are the easiest for the editor to handle, but any word processor is appropriate. If you are not using TeX and your insert contains a significant amount of mathematics, please FAX (409–845–3144) a copy of the insert so we can see the intended appearance of the text.

2. Any ado-files, `.exe` files, or other software that accompanies the submission.

3. A help file for each ado-file included in the submission. See any recent STB diskette for the structure a help file. If you have questions, fill in as much of the information as possible and we will take care of the details.

4. A do-file that replicates the examples in your text. Also include the datasets used in the example. This allows us to verify that the software works as described and allows users to replicate the examples as a way of learning how to use the software.

5. Files containing the graphs to be included in the insert. If you have used STAGE to edit the graphs in your submission, be sure to include the `.gph` files. Do not add titles (e.g., “Figure 1: ...”) to your graphs as we will have to strip them off.

The easiest way to submit an insert to the STB is to first create a single “archive file” (either a `.zip` file or a compressed `.tar` file) containing all of the files associated with the submission, and then email it to the editor at `stb@stata.com` either by first using `uuencode` if you are working on a Unix platform or by attaching it to an email message if your mailer allows the sending of attachments. In Unix, for example, to email the current directory and all of its subdirectories:

```
tar -cf - . | compress | uuencode xyzz.tar.Z > whatever
mail stb@stata.com < whatever
```

International Stata Distributors

International Stata users may also order subscriptions to the *Stata Technical Bulletin* from our International Stata Distributors.

Company: Applied Statistics &
Systems Consultants
Address: P.O. Box 1169
17100 NAZERATH-ELLIT
Israel
Phone: +972 (0)6 6100101
Fax: +972 (0)6 6554254
Email: assc@netvision.net.il
Countries served: Israel

Company: Axon Technology Company Ltd
Address: 9F, No. 259, Sec. 2
Ho-Ping East Road
TAIPEI 106
Taiwan
Phone: +886-(0)2-27045535
Fax: +886-(0)2-27541785
Email: hank@axon.axon.com.tw
Countries served: Taiwan

Company: Chips Electronics
Address: Lokasari Plaza 1st Floor Room 82
Jalan Mangga Besar Raya No. 82
JAKARTA
Indonesia
Phone: 62 - 21 - 600 66 47
Fax: 62 - 21 - 600 66 47
Email: puyuh23@indo.net.id
Countries served: Indonesia

Company: Dittrich & Partner Consulting
Address: Kieler Strasse 17
5. floor
D-42697 Solingen
Germany
Phone: +49 2 12 / 26 066 - 0
Fax: +49 2 12 / 26 066 - 66
Email: sales@dpc.de
URL: http://www.dpc.de
Countries served: Germany, Austria, Italy

Company: IEM
Address: P.O. Box 2222
PRIMROSE 1416
South Africa
Phone: +27-11-8286169
Fax: +27-11-8221377
Email: iem@hot.co.za
Countries served: South Africa, Botswana,
Lesotho, Namibia, Mozambique,
Swaziland, Zimbabwe

Company: MercoStat Consultores
Address: 9 de junio 1389
CP 11400 MONTEVIDEO
Uruguay
Phone: 598-2-613-7905
Fax: Same
Email: mercost@adinet.com.uy
Countries served: Uruguay, Argentina, Brazil,
Paraguay

Company: Metrika Consulting
Address: Mosstorpsvagen 48
183 30 Taby STOCKHOLM
Sweden
Phone: +46-708-163128
Fax: +46-8-7924747
Email: sales@metrika.se
URL: http://www.metrika.se
Countries served: Sweden, Baltic States,
Denmark, Finland, Iceland,
Norway

Company: Ritme Informatique
Address: 34, boulevard Haussmann
75009 Paris
France
Phone: +33 (0)1 42 46 00 42
+33 (0)1 42 46 00 33
Email: info@ritme.com
URL: http://www.ritme.com
Countries served: France, Belgium,
Luxembourg

(*List continued on next page*)

International Stata Distributors

(*Continued from previous page*)

Company: Scientific Solutions S.A.
Address: Avenue du Général Guisan, 5
CH-1009 Pully/Lausanne
Switzerland
Phone: 41 (0)21 711 15 20
Fax: 41 (0)21 711 15 21
Email: info@scientific-solutions.ch
Countries served: Switzerland

Company: Smit Consult
Address: Doormanstraat 19
5151 GM Drunen
Netherlands
Phone: +31 416-378 125
Fax: +31 416-378 385
Email: info@smitconsult.nl
URL: http://www.smitconsult.nl
Countries served: Netherlands

Company: Survey Design & Analysis
Services P/L
Address: 249 Eramosa Road West
Moorooduc VIC 3933
Australia
Phone: +61 (0)3 5978 8329
Fax: +61 (0)3 5978 8623
Email: sales@survey-design.com.au
URL: http://survey-design.com.au
Countries served: Australia, New Zealand

Company: Timberlake Consultants
Address: Unit B3 Broomsleigh Bus. Park
Worsley Bridge Road
LONDON SE26 5BN
United Kingdom
Phone: +44 (0)208 697 3377
Fax: +44 (0)208 697 3388
Email: info@timberlake.co.uk
URL: http://www.timberlake.co.uk
Countries served: United Kingdom, Eire

Company: Timberlake Consulting S.L.
Address: Calle Mendez Nunez, 1, 3
41011 Sevilla
Spain
Phone: +34 (9) 5 422 0648
Fax: +34 (9) 5 422 0648
Email: timberlake@zoom.es
Countries served: Spain

Company: Timberlake Consultores, Lda.
Address: Praceta Raúl Brandao, n°1, 1°E
2720 ALFRAGIDE
Portugal
Phone: +351 (0)1 471 73 47
Fax: +351 (0)1 471 73 47
Email: timberlake.co@mail.telepac.pt
Countries served: Portugal

Company: Unidost A.S.
Rihtim Cad. Polat Han D:38
Kadikoy
81320 ISTANBUL
Turkey
Phone: +90 (216) 414 19 58
Fax: +30 (216) 336 89 23
Email: info@unidost.com
URL: http://abone.turk.net/unidost
Countries served: Turkey

Company: Vishvas Marketing-Mix Services
Address: C\O S. D. Wamorkar
"Prashant" Vishnu Nagar, Naupada
THANE - 400602
India
Phone: +91-251-440087
Fax: +91-22-5378552
Email: vishvas@vsnl.com
Countries served: India