STATA TECHNICAL BULLETIN

March 2000
STB-54

A publication to promote communication among Stata users

Contents of this issue

stata54 | Multiple curves plotted with stcurv command

Mario Cleves, Stata Corporation, mcleves@stata.com

Abstract: Stata's `stcurv` command which is used after `streg` to plot the fitted cumulative hazard, survival, and hazard functions, has been modified so that multiple curves can be plotted on the same graph.

Keywords: parametric survival, survival models, regression.

`stcurv` has been modified so that multiple curves can be plotted on the same graph. This is done by specifying multiple options, `at1()`, `at2()`, ..., one for each curve to be plotted.

Syntax

`stcurv` [, cumhaz survival hazard range(*##*)

[at1(*varname*=*#* [*varname*=*#*...]) [at2(*varname*=*#* [*varname*=*#*...]) [...]]]

graph_options]

The multiple `at1()`, `at2()`, ..., options are new. See [R] **streg** for a description of the other options.

`stcurv` is used after `streg` to plot the cumulative hazard, survival, and hazard functions at the mean value of the covariates or at values specified by the `at()` options.

New options

`at1(`*varname*=*#* ...`)`, `at2(`*varname*=*#* ...`)`, ..., `at10(`*varname*=*#* ...`)` specify that multiple curves (up to ten) are to be plotted on the same graph. `at1()`, `at2()`, ..., `at10()` work like the `at()` option: the option causes the function to be evaluated at the value of the covariates specified and at the mean of all unlisted covariates. `at1()` specifies the values of the covariates for the first curve, `at2()` specifies the values of the covariates for the second curve, and so on.

Up to ten `at()` options can be specified at one time. Each `at()` option produces a separate curve on the same graph.

Example

We demonstrate the use of the multiple `at()` options by fitting a log-logistic regression model to the cancer data distributed with Stata, and plotting several predicted survival curves at various covariate values. For this example, we combine `drug==2` and `drug==3` into one group.

```
. use cancer, clear
(Patient Survival in Drug Trial)
. replace drug=2 if drug==3
(14 real changes made)
. stset studytim, failure(died)
  (output omitted)
. streg age drug, dist(llog) nolog

         failure _d:  died
   analysis time _t:  studytim

Log-logistic regression -- accelerated failure-time form
No. of subjects =          48                     Number of obs   =        48
No. of failures =          31
Time at risk    =         744
                                                  LR chi2(2)      =     35.14
Log likelihood  =    -43.21698                    Prob > chi2     =    0.0000
------------------------------------------------------------------------------
      _t |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |  -.0803289   .0221598     -3.625   0.000      -.1237614   -.0368964
    drug |   1.420237   .2502148      5.676   0.000       .9298251    1.910649
   _cons |   5.026474   1.225037      4.103   0.000       2.625446    7.427502
------------------------------------------------------------------------------
 /ln_gam |  -.8456552   .1479337     -5.716   0.000        -1.1356   -.5557104
------------------------------------------------------------------------------
   gamma |    .429276   .0635044                          .3212293    .5736646
------------------------------------------------------------------------------
```

We first obtain a graph with two predicted survival curves, one for each drug treatment group, at the overall average age.

```
. stcurv, survival at1(drug=1) at2(drug=2) c(ll) xlab ylab
```

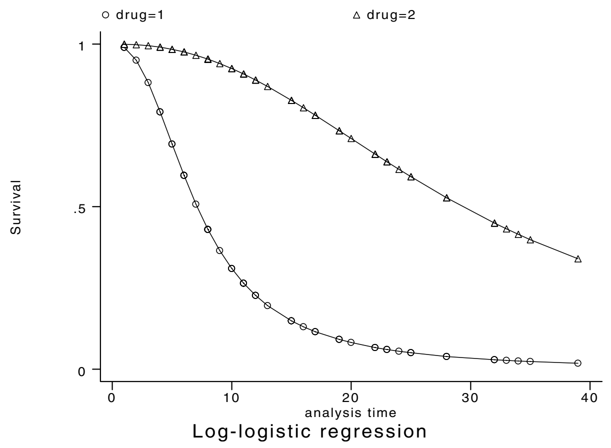


Figure 1. Predicted survival curves for drug treatment groups at overall average age.

We specified two `at()` options, one for each drug group. Now let's plot the two treatment groups, not at the average patient age, but, for example, at `age` 40.

```
. stcurv, survival at1(drug=1 age=40) at2(drug=2 age=40) c(ll) xlab ylab
```

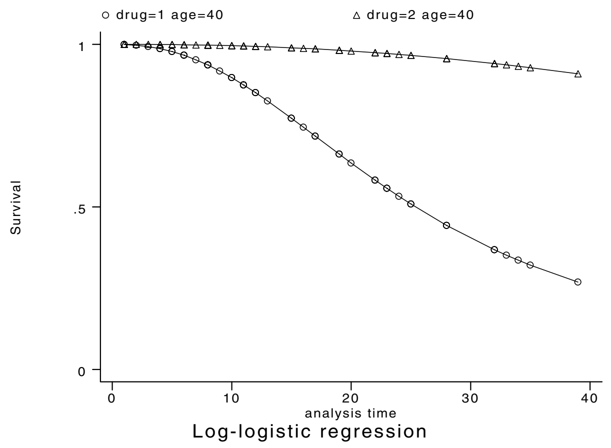


Figure 2. Predicted survival curves for drug treatment groups at age 40.

Again we specified `at()` twice, but now we included `age`=40 in each option's argument. We could include additional curves in the graph; for example, to the previous graph we now add two more curves each at `age` 65.

```
. stcurv, survival at1(drug=1 age=40) at2(drug=2 age=40) at3(drug=1 age=65)
> at4(drug=2 age=65) c(llll) xlab ylab
```

(*Graph on next page*)

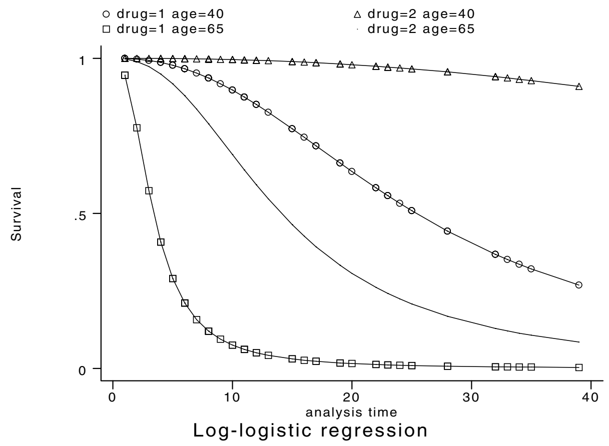


Figure 3. Predicted survival curves for drug treatment groups at ages 40 and 65.

stata55	Search web for installable packages

William Gould, Stata Corporation, wgould@stata.com
Alan Riley, Stata Corporation, ariley@stata.com

Abstract: `webseek` searches the web for user-written additions to Stata, which is to say, new commands. The search includes but is not limited to additions published in the STB. The commands found are available for immediate installation using the `net` command or, under Windows and Macintosh, by clicking on the link shown in `webseek`'s output. `webseek` can find additions based on topic, author name, or the command name.

Keywords: search, net, web, user-written additions, programs, commands.

Syntax

`webseek` *keywords* [`, or nostb tocpkg toc pkg` <u>`e`</u>`verywhere` <u>`f`</u>`ilenames` <u>`h`</u>`elp` <u>`r`</u>`esult` <u>`t`</u>`ype errnone`]

Description

`webseek` searches the web for user-written additions to Stata, which is to say, new commands. The search includes, but is not limited to, additions published in the STB.

The commands found are available for immediate installation using the `net` command or, under Windows and Macintosh, by clicking on the link shown in `webseek`'s output. `webseek` can find additions based on topic, author name, or the command name.

Options

`or` is relevant only when multiple keywords are specified. By default, only packages that include all the keywords are listed. `or` changes this to list packages that contain any of the keywords.

`nostb` restricts the search to non-STB sources or, said differently, causes `webseek` not to list matches that were published in the STB.

`tocpkg`, `toc`, and `pkg` determine what is searched. `tocpkg` is the default, meaning that both table of contents (tocs) and packages (pkgs) are searched. `toc` restricts the search to table of contents only. `pkg` restricts the search to packages only.

`everywhere` and `filenames` determine where in packages `webseek` looks for keywords. The default is `everywhere`. `filenames` restricts `webseek` to search for matches only in the filenames associated with a package. Specifying `everywhere` implies `pkg`.

`help`, `result`, and `type` determine how and where results are displayed.

`help` specifies that results are to be displayed in the help window, where you can point and click to visit the links. `help` is the default with Stata for Windows and Stata for Macintosh. `help` may not be specified with Stata for Unix (because there is no help window).

`result` specifies that results are to be displayed in the standard Stata results window. `result` is the default with Unix but the option may be specified with Windows or Macintosh.

`type` is the default on no platform but may be specified on all. It presents output much like `result`, but without highlighting. Its advantage is that the results of a search can be logged.

In addition, you may set the global macro `$webseek` to contain `help`, `result`, or `type` and so specify your own default.

`errnone` is an option for programmers using `webseek` as a subroutine. It causes the return code to be 111 rather than 0 when no matches are found.

Remarks

Not just we at Stata, but others can write new commands for Stata, so if Stata cannot do something it may be that someone has written an addition to do it. The problem is finding that addition.

`webseek` searches the web for `net`-installable additions to Stata. `net` (see **[R] net**) is the Stata command that can install new additions to Stata. If you knew, for instance, that a user A. Smith wrote an addition you wanted and that it was available as package `veryneat` at http://www.university.edu/~asmith, you could type

```
. net from http://www.university.edu/~asmith
. net install veryneat
```

and then you would have the `veryneat` command. Probably A. Smith provided a help file to go with the new command, so typing `help veryneat` should now tell you something about how to use this new command. Eventually, you would discover that command `veryneat` was very useful or it was not worth the disk space it occupied. If the latter, you could type

```
. ado uninstall veryneat
```

and so remove it from your computer.

The problem is in finding the `veryneat` command in the first place. `webseek` helps with that.

Example 1: Find what is available about "random effects"

```
. webseek random effect
```

Comments:

1. It is best to search for the singular. '`webseek random effect`' will find both "random effect" and "random effects".
2. '`webseek random effect`' will also find "random-effect" (note the hyphen) because `webseek` performs a string search, not a word search.
3. '`webseek random effect`' lists all packages containing the words "random" and "effect", not necessarily used together.
4. If you wanted all packages containing the word "random" or the word "effect", you would type '`webseek random effect, or`'.

Example 2: Find what is available by author Jeroen Weesie

```
.  webseek weesie
```

Comments:

1. You could type '`webseek jeroen weesie`' but that might list less because perhaps the last name is used without the first.
2. You could type '`webseek Weesie`' and that would produce the same results. Capitalization, both in what you type and what is at the site, is ignored in the search.

Example 3: Same as example 2, but do not list STB materials

```
.  webseek weesie, nostb
```

Comments:

1. The STB tends to dominate search results because so much has been published in the STB. If you know what you are looking for is not in the STB, specifying the `nostb` option will narrow the search.
2. '`webseek weesie`' lists everything '`webseek weesie, nostb`' lists, and more. If you just type '`webseek weesie`', look down the list. STB materials are listed first and non-STB materials are listed after that.

Example 4: Find the user-written command kursus

```
        . webseek kursus, file
```

Comments:

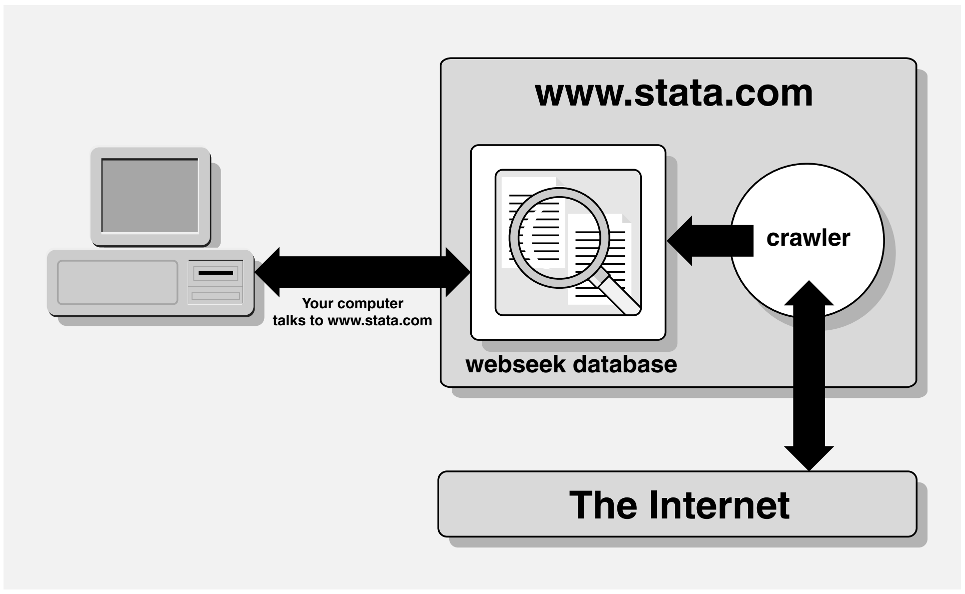
1. You could just type '`webseek kursus`' and that will list everything '`webseek kursus, file`' lists, and more. Since you know `kursus` is a command, however, there must be a kursus.ado file associated with the package. Typing '`webseek kursus, file`' narrows the search.
2. You could also type '`webseek kursus.ado, file`' to narrow the search even more.

Where does webseek look?

`webseek` looks everywhere, not just at www.stata.com. `webseek` begins by looking at www.stata.com, but then follows every link, which takes it to other places, and it then follows every link, which takes it to yet more places, and so on.

Authors: please let us know if you have a site we should include in our search by sending email to webseek@stata.com. We will then link to your site from ours and so ensure that `webseek` finds your materials. That is not strictly necessary, however, as long as your site is linked from some site that is linked to ours, even if that link is indirect.

How does webseek really work?



www.stata.com maintains a database of Stata resources. When you use `webseek`, `webseek` contacts stata.com with your request, stata.com searches its database, and returns the result to you.

Another part of the system is called the crawler: it searches the web for new Stata resources to add to the `webseek` database and it verifies that the resources already found are still available. Given how the crawler works, when a new resource becomes available, the crawler takes about two days to notice it and, similarly, if a resource disappears, the crawler takes roughly two days before it is removed from the database.

Note

When you use `webseek`, it creates file wseekres.hlp in the current directory. If the file bothers you, you may erase it.

dm73.1 Contrasts for categorical variables: update

John Hendrickx, University of Nijmegen, Netherlands, J.Hendrickx@mailbox.kun.nl

Abstract: Bug fixes and enhancements to `desmat` and associated programs for models with categorical independent variables are described.

Keywords: Contrasts, interactions, categorical variables.

Changes to desmat

The program `desmat` can be used to create dummy variables for categorical variables using a variety of contrasts (Hendrickx 1999). This update corrects bugs in the original version and adds a minor enhancement. These bugs can occur if categorical variables have values other than their rank number, in which case dummies using the deviation, difference, or Helmert contrasts will be incorrect. It also turns out that `orthpoly` can produce errors if large values such as years are used. This problem has been reported and circumvented in `desmat` by subtracting the lowest value of the variable before calling `orthpoly`.

An enhancement to `desmat` is the option to assign a contrast to a variable by using a `pzat` characteristic. For example, to specify that the variable `educ` should be treated as continuous by `desmat`, use

```
. char educ[pzat] dir
```

The `pzat` characteristic overrides the default parameterization specified as an option to the `desmat` statement. For example:

```
. desmat educ focc, dif
```

`desmat` will treat `educ` as a continuous variable but will use the difference contrast for `focc`. This can also be achieved by appending `=par[(ref)]` to specific model terms; for example:

```
. desmat educ=dir focc, dif
```

Using the `pzat` characteristic can be more practical in large models where a specification per variable would become overly long. A specification per variable can be used to override the `pzat` characteristic. For example, specifying `educ=sim(1)` in the above statement will cause the simple contrast to be used for `educ`.

Changes to desrep

`desrep` can be used after estimating a model to produce an overview of the results using informative labels. It will now work properly with `mlogit` (the previous version stripped equation names from `_b()` and `_se()` when formatting the results). `desrep` will also print model results such as the procedure name, dependent variable, sample size, log likelihood, F-statistic, chi-square, etc. If certain `e()` macros have been defined by a procedure, they will be printed by `desrep` with a suitable label.

Replacement of tstall by destest

In Hendrickx (1999), `tstall` was provided to perform a Wald test on all model terms after estimating a model generated by `desmat`. An enhanced version renamed `destest` can now do tests on specific terms only. The syntax is

`destest` [*termlist*] [`, equal joint`]

The *termlist* consists of one or more terms as specified in `desmat`. A term can consist of a single variable, or two or more variables separated by either asterisks or periods. If asterisks are used, they will be changed into periods by `destest`, that is, only the highest order interaction will be tested. This syntax makes it easier to copy the model syntax and test the highest order terms, which is what people will usually want to do. If `destest` is specified without any arguments, all terms from the last `desmat` model will be tested.

The default is to test whether the effects of each separate term are equal to zero. If the option `joint` is specified, `destest` will test instead whether all the effects in *termlist* are jointly equal to zero. If the option `equal` is specified, `destest` will test whether the effects of each separate term are equal. The `joint` and `equal` options may be combined to test whether all effects are jointly equal, although this would be a somewhat peculiar hypothesis.

Reference

Hendrickx, J. 1999. dm73: Using categorical variables in Stata. *Stata Technical Bulletin* 52: 2–8.

dm76	ICD-9 diagnostic and procedure codes

William Gould, Stata Corporation, wgould@stata.com

Abstract: Two commands are provided for dealing with ICD-9 codes; `icd9` for use with diagnostic codes and `icd9p` for use with procedure codes.

Keywords: ICD-9-CM diagnostic codes, ICD-9-CM procedure codes.

Completing the installation

The installation process for the `icd9` and `icd9p` commands are a little different than the standard. In addition to `net install`, you must `net get` and then you must type `icd9 install` and `icd9p install`:

```
. net install dm76
. net get dm76
. icd9 install
. icd9p install
```

The `net get` copies two datasets that `icd9` and `icd9p` need that contain the mapping from codes to text. The `icd9 install` and `icd9p install` then moves each of the datasets from the current directory to the directory in which the commands are installed.

Syntax

Note: `icd9` is for use with ICD-9 *diagnostic* codes and `icd9p` is for use with *procedure codes*. These are two commands whose syntax exactly parallels each other. Below we write `icd9`[`p`] to mean both commands:

```
icd9[p] check varname [, any list generate(newvar) ]

icd9[p] clean varname [, dots pad ]

icd9[p] generate newvar = varname, main

icd9[p] generate newvar = varname, description [long end]

icd9[p] generate newvar = varname, range(icd9rangelist)

icd9[p] lookup icd9rangelist

icd9[p] search ["]text["] [["]text["] [...]] [, or]

icd9[p] install [, replace]

icd9[p] query
```

icd9rangelist is

icd9code	meaning the particular code
icd9code`*`	meaning all codes starting with *icd9code*
icd9code`/`*icd9code*	meaning the code range including endpoints

or any combination of the above, such as "`001* 018/019 E* 018.02`". Note that *icd9codes* must be typed with leading zeros: `1` is an error; type `001` (diagnostic code) or `01` (procedure code).

Description

`icd9` and `icd9p` assist with working with ICD-9-CM codes. ICD-9-CM refers to the fifth edition of the International Classification of Diseases, 9th revision, Clinical Modification.

ICD-9 codes come in two forms: diagnostic codes and procedure codes. 001 (cholera), 572.0 (abscess of liver), 941.45 (deep 3rd deg burn nose), and E873 (watercraft explosion) are examples of diagnostic codes, although some people write (and datasets record) 94145 rather than 941.45. `icd9` understands both ways of recording the codes. 01 (incise-excis brain/skill), 01.5 (skill biopsy), 55 (operations on kidney), and 55.01 (nephrotomy) are examples of procedure codes, although some people write 5501 rather than 55.01. `icd9p` understands both ways of recording codes.

`icd9` and `icd9p` exactly parallel each other, it is just that `icd9` is for use with diagnostic codes and `icd9p` for use with procedure codes. Below we will write `icd9`[`p`] to mean both commands.

`icd9`[p] `check` verifies that already existing variable *varname* contains valid ICD-9 codes. If not, `icd9`[p] `check` provides a full report on the problems. Use of `icd9`[p] `check` is optional. `icd9`[p] `check` is useful for tracking down problems when any of the other `icd9`[p] commands tell you "variable does not contain ICD-9 codes". `icd9`[p] `check` is a little more thorough, too, in that it verifies that each of the recorded codes actually exists in the official list.

`icd9`[p] `clean` also verifies that already existing variable *varname* contains valid ICD-9 codes and, if it does, `icd9`[p] `clean` modifies the variable to contain the codes in either of two standard formats—with or without the periods separating the main code from the detail. Use of `icd9`[p] `clean` is optional; all `icd9`[p] commands work equally well with cleaned or uncleaned codes. There are numerous ways of writing the same ICD-9 code and `icd9`[p] `clean` is designed (1) to ensure consistency and (2) to make subsequent output look better.

`icd9`[p] `generate` produces new variables based on already existing variables containing (cleaned or uncleaned) ICD-9 codes. `icd9`[p] `generate, main` produces *newvar* containing the main code. `icd9`[p] `generate, description` produces *newvar* containing a textual description of the ICD-9 code. `icd9`[p] `generate, range()` produces numeric *newvar* containing 1 if *varname* records an ICD-9 code in the range listed and 0 otherwise.

`icd9`[p] `lookup` and `icd9`[p] `search` are utility routines useful interactively. `icd9`[p] `lookup` simply displays descriptions of codes specified on the command line, so if you have a yearning to know what diagnostic E913.1 means, you can type "`icd9 lookup e913.1`". Whatever data you have in memory is irrelevant—and remains unchanged—when using `icd9`[p] `lookup`. `icd9`[p] `search` is like `icd9`[p] `lookup` except that it turns the problem around; `icd9`[p] `search` looks for relevant ICD-9 codes from the description given on the command line. For instance, you could type "`icd9 search liver`" or "`icd9p search liver`" to obtain a list of codes containing the word liver.

`icd9`[p] `install` has to do with installation of the `icd9`[p] command. See the section *Completing the installation* above.

`icd9`[p] `query` displays the identity of the source from which were obtained the ICD-9 codes and textual descriptions that `icd9`[p] uses.

Note that ICD-9 codes are commonly written two ways, with and without periods. For instance, with diagnostic codes, one can write 001, 86221, E8008, and V822, or one can write 001., 862.21, E800.8, and V82.2. With procedure codes, one can write 01, 50, 502, 5021, or one can write 01., 50., 50.2, 50.21. The `icd9`[p] command does not care which syntax you use or even whether you are consistent. Case also is irrelevant: v822, v82.2, V822, and V82.2 are all equivalent. Codes may be recorded with or without leading and trailing blanks.

Options for use with icd9[p] check

`any` tells `icd9`[p] `check` to verify the codes fit the format of ICD-9 codes but to skip checking whether the codes are actually valid. This makes `icd9`[p] `check` run faster. For instance, diagnostic code 230.52 (or 23052 if you prefer) looks to be valid, but in fact there is no such ICD-9 code, at least currently. Without the `any` option, 230.52 (23052) would be flagged as an error. With `any`, 230.52 (23052) is not considered an error.

`list` tells `icd9`[p] `check` that invalid codes found in the data—1, 1.1.1, and perhaps 230.52 assuming `any` is not also specified—are to be individually listed.

`generate(`*newvar*`)` specifies that `icd9`[p] `check` is to create new variable *newvar* containing, for each observation, 0 if the code is valid and a number from 1 to 10 if not. The positive numbers indicate the kind of problem and correspond to the listing produced by `icd9`[p] `check`. For instance, 10 means the code could be valid, it just turns out not to be on the official list.

Options for use with icd9[p] clean

`dots` specifies whether periods are to be included in the final format. Do you wish diagnostic codes recorded, for instance, 86221 or 862.21? Without the `dots` option, the former format is used. With the `dots` option, the latter format is used.

`pad` specifies that the codes are to be padded with spaces, front and back, to make the codes line up vertically in listings. Specifying `pad` makes the resulting codes look better when used with most other Stata commands.

Technical Note: If you specify `pad`, the following character positions are used with *diagnostic* codes:

position	`nodot`	position	`dot`
1	`E` or `" "`	1	`E` or `" "`
2–4	rest of main code	2-4	rest of main code
5–6	detail code or spaces	5	`"."` or `" "`
		6–7	detail code or spaces

If `pad` is not specified, the ICD-9 diagnostic code is written without leading or trailing blanks, meaning

position	`nodot`
1–3 or 1–4	optional `E` + rest of main code
4–5 or 5–6	detail code or nothing

and

position	`dot`
1–3 or 1–4	optional `E` + rest of main code
4 or 5	`"."` or nothing
5–6 or 6–7	detail code or nothing

With procedure codes (which never have leading letters), the column positions when `pad` is specified are

position	`nodot`	position	`dot`
1–2	main code	1-2	main code
3–4	detail code or spaces	3	`"."` or `" "`
5	`" "`	4–5	detail code or spaces

If `pad` is not specified, the ICD-9 procedure code is written without trailing blanks.

Options for use with icd9[p] generate

`main`, `description`, and `range()` specify what `icd9`[p] `generate` is to calculate. In all cases, *varname* specifies a variable containing ICD-9 codes.

`main` specifies that the main code is to be extracted from the ICD-9 code. For procedure codes, the main code is the first two characters. For diagnostic codes, the main code is usually the first three or four characters (the characters before the dot if the code has dots). In any case, `icd9`[p] `generate` does not care whether the code is padded with blanks in front or how strangely it might be written; `icd9`[p] `generate` will find the main code and extract it. The resulting variable is itself an ICD-9 code and may be used with the other `icd9`[p] subcommands. This includes `icd9`[p] `generate, main` because main codes of main codes are main codes.

`description` creates *newvar* containing descriptions of the ICD-9 codes.

`long` is for use with `description`. It specifies that the new variable, in addition to containing the text describing the code, is to contain the code, too. Without `long`, *newvar* in an observation might contain "bronchus injury-closed". With `long`, it would contain " 862.21 bronchus injury-closed".

`end` modifies `long` and places the code at the end of the string: "bronchus injury-closed 862.21". Specifying `end` implies `long`.

`range()` allows you to create indicator variables equal to 1 when the ICD-9 code is in the inclusive range specified.

Options for use with icd9[p] search

`or` specifies that ICD-9 codes are to be searched for any entry that contains any of the words specified after `icd9`[p] `search`. The default is to list only entries that contain all the words specified.

Options for use with icd9[p] install

`replace` specifies that the completion of the installation is to be done again. Specify `replace` if you type `icd9`[p] `install`, are told that you have already done that, and really do want to reinstall.

Remarks

Let us begin with diagnostic codes—the codes `icd9` processes. The format of an ICD-9 diagnostic code is

$$\left[\text{blanks}\right]\left\{\texttt{0–9,V,v}\right\}\left\{\texttt{0–9}\right\}\left\{\texttt{0–9}\right\}\left[\texttt{.}\right]\left[\texttt{0–9}\left[\texttt{0–9}\right]\right]\left[\text{blanks}\right]$$

or

$$\left[\text{blanks}\right]\left\{\texttt{E,e}\right\}\left\{\texttt{0–9}\right\}\left\{\texttt{0–9}\right\}\left\{\texttt{0–9}\right\}\left[\texttt{.}\right]\left[\texttt{0–9}\left[\texttt{0–9}\right]\right]\left[\text{blanks}\right]$$

`icd9` can deal with ICD-9 diagnostic codes written any of the ways the above allows. Items in square brackets are optional. The code might start with some number of blanks. Braces { } indicate required items. The code either then has a digit from `0` to `9` or the letter V (uppercase or lowercase) (first line) or it has the letter E (uppercase or lowercase, second line). After that, it has two or more digits, perhaps followed by a period, and after that it may have up to two more digits (perhaps followed by more blanks).

All of the following meet the above definition:

```
001
001.
   001
001.9
         0019
86222
862.22
E800.2
e8002
V82
v82.2
V822
```

Meeting the above definition does not make the code valid. There are 233,100 possible codes meeting the above definition, of which 15,186 are currently defined.

Examples of currently defined diagnostic codes include

code	description
001	cholera*
001.0	cholera d/t vib cholerae
001.1	cholera d/t vib el tor
001.9	cholera nos
...	
999	complic medical care nec*
...	
V01	communicable dis contact*
V01.0	cholera contact
V01.1	tuberculosis contact
V01.2	poliomyelitis contact
V01.3	smallpox contact
V01.4	rubella contact
V01.5	rabies contact
V01.6	venereal dis contact
V01.7	viral dis contact nec
V01.8	communic dis contact nec
V01.9	communic dis contact nos
...	
E800	rr collision nos*
E800.0	rr collision nos-employ
E800.1	rr coll nos-passenger
E800.2	rr coll nos-pedestrian
E800.3	rr coll nos-ped cyclist
E800.8	rr coll nos-person nec
E800.9	rr coll nos-person nos
...	

“Main codes” refer to the part of the code to the left of the period. 001, 002, ..., 999, V01, ..., V82, E800, ..., E999 are main codes. There are 1,182 diagnostic main codes.

The main code corresponding to a detailed code can be obtained by taking the part of the code to the left of the period, except for codes beginning with 176, 764, 765, V29, and V69. Those main codes are not defined and yet, there are more detailed codes under them:

(*Continued on next page*)

code	description
176	CODE DOES NOT EXIST, but 8 codes starting with 176 do exist:
176.0	skin - kaposi's sarcoma
176.1	sft tisue - kpsi's srcma
...	
764	CODE DOES NOT EXIST, but 44 codes starting with 764 do exist:
764.0	lt-for-dates w/o fet mal*
764.00	light-for-dates wtnos
...	
765	CODE DOES NOT EXIST, but 22 codes starting with 765 do exist:
765.0	extreme immaturity*
765.00	extreme immatur wtnos
...	
V29	CODES DOES NOT EXIST, but 6 codes stating with V29 do exist:
V29.0	nb obsrv suspct infect
V29.1	nb obsrv suspct neurlgcl
...	
V69	CODE DOES NOT EXIST, but 6 codes starting with V69 do exist:
V69.0	lack of physical exercse
V69.1	inapprt diet eat habits
...	

Our solution is to define four new codes:

code	description
176	kaposi's sarcoma (Stata)*
764	light-for-dates (Stata)*
765	immat & preterm (Stata)*
V29	nb suspct cnd (Stata)*
V69	lifestyle (Stata)*

Thus, there are 15,186 + 5 = 15,191 diagnostic codes of which 1,181 + 5 = 1,186 are main codes.

Things are less confusing with respect to procedure codes—the codes processed by `icd9p`. The format of ICD-9 procedure codes is

$$\big[\text{blanks}\big]\big\{\texttt{0–9}\big\}\big\{\texttt{0–9}\big\}\big[\texttt{.}\big]\big[\texttt{0–9}\big[\texttt{0–9}\big]\big]\big[\text{blanks}\big]$$

Thus, there are 10,000 possible procedure codes of which 4,275 are currently valid. The first two digits represent the main code, of which there are 100 feasible and 98 are currently used (00 and 17 are not used).

Descriptions

The descriptions given for each of the codes is as found in the original source with, in the case of procedure codes, the addition of five new codes by us. An asterisk on the end of a description indicates that the corresponding ICD-9 diagnostic code has subcategories.

`icd9`[`p`] `query` reports the original source of the information on the codes:

```
. icd9 query
_dta:
  1.  Dataset obtained 24aug1999
  2.  from http://www.hcfa.gov/stats/pufiles.htm
  3.  file http://www.hcfa.gov/stats/icd9v16.exe
  4.  Codes 176, 764, 765, V29, and V69 defined
  5.  -- 176 kaposi´s sarcoma (Stata)*
  6.  -- 765 immat & preterm (Stata)*
  7.  -- 764 light-for-dates (Stata)*
  8.  -- V29 nb suspct cnd (Stata)*
  9.  -- V69 lifestyle (Stata)*

. icd9p query
_dta:
  1.  Dataset obtained 24aug1999
  2.  from http://www.hcfa.gov/stats/pufiles.htm
  3.  file http://www.hcfa.gov/stats/icd9v16.exe
```

Example

You have a dataset containing up to three diagnostic codes and up to two procedures on a sample of 1,000 patients:

```
. use patients, clear
. list in 1/10
        patid  diag1      diag2      diag3      proc1      proc2
  1.        1  65450                            9383
  2.        2  23v.6      37456                 8383       17
  3.        3  V10.02
  4.        4  102.6                            629
  5.        5   861.01
  6.        6  38601      2969                  9337
  7.        7  705                              7309       8385
  8.        8  v53.32                           7878       951
  9.        9   20200     7548       E8247      0479
 10.       10  464.11     20197                 4641
```

Do not try to make sense of this data because, in constructing this example, the diagnostic and procedure codes were chosen at random.

Begin by noting that variable `diag1` is recorded sloppily—sometimes the dot notation is used, sometimes not, and sometimes there are leading blanks. That does not matter. We decide to begin by using `icd9 clean` to clean up this variable:

```
. icd9 clean diag1
diag1 contains invalid ICD-9 codes
r(459);
```

`icd9 clean` refused because there are invalid codes among the 1,000 observations. We can use `icd9 check` to find out about the problems:

```
. icd9 check diag1
diag1 contains invalid codes:
    1.  Invalid placement of period                       0
    2.  Too many periods                                  0
    3.  Code too short                                    0
    4.  Code too long                                     0
    5.  Invalid 1st char (not 0-9, E, or V)               0
    6.  Invalid 2nd char (not 0-9)                        0
    7.  Invalid 3rd char (not 0-9)                        1
    8.  Invalid 4th char (not 0-9)                        0
    9.  Invalid 5th char (not 0-9)                        0
   10.  Code not defined                                  0
                                                -----------
        Total                                             1
```

There is only one observation with a problem. We can find that observation by asking `icd9 check` to flag the problem observations (or observation, as it is in this case):

```
. icd9 check diag1, gen(prob)
diag1 contains invalid codes:
    1.  Invalid placement of period                       0
    2.  Too many periods                                  0
    3.  Code too short                                    0
    4.  Code too long                                     0
    5.  Invalid 1st char (not 0-9, E, or V)               0
    6.  Invalid 2nd char (not 0-9)                        0
    7.  Invalid 3rd char (not 0-9)                        1
    8.  Invalid 4th char (not 0-9)                        0
    9.  Invalid 5th char (not 0-9)                        0
   10.  Code not defined                                  0
                                                -----------
        Total                                             1
. list patid diag1 prob if prob
         patid  diag1          prob
   2.        2  23v.6             7
```

Let's assume we go back to the patient records and determine that this should have been coded 230.6:

```
. replace diag1 = "230.6" if patid==2
(1 real change made)
. drop prob
```

We now try again to clean up the formatting of the variable:

```
. icd9 clean diag1
(643 changes made)
. list in 1/10
```

```
       patid  diag1     diag2      diag3      proc1      proc2
  1.       1  65450                           9383
  2.       2  2306      37456                 8383       17
  3.       3  V1002
  4.       4  1026                            629
  5.       5  86101
  6.       6  38601     2969                  9337
  7.       7  705                              7309       8385
  8.       8  V5332                           7878       951
  9.       9  20200     7548       E8247      0479
 10.      10  46411     20197                 4641
```

Perhaps we prefer the dot notation. `icd9 clean` can be used again on `diag1`, and now we will continue to clean up `diag2` and `diag3`:

```
. icd9 clean diag1, dots
(936 changes made)

. icd9 clean diag2, dots
(551 changes made)

. icd9 clean diag3, dots
(100 changes made)

. list in 1/10
       patid  diag1     diag2     diag3     proc1      proc2
  1.       1  654.50                        9383
  2.       2  230.6     374.56              8383       17
  3.       3  V10.02
  4.       4  102.6                         629
  5.       5  861.01
  6.       6  386.01    296.9               9337
  7.       7  705                            7309       8385
  8.       8  V53.32                        7878       951
  9.       9  202.00    754.8     E824.7    0479
 10.      10  464.11    201.97              4641
```

We now turn to cleaning the procedure codes. We use `icd9p` (emphasis on the *p*) to clean these codes:

```
. icd9p clean proc1, dots
(816 changes made)

. icd9p clean proc2, dots
(140 changes made)

. list in 1/10
       patid  diag1     diag2     diag3     proc1     proc2
  1.       1  654.50                        93.83
  2.       2  230.6     374.56              83.83     17
  3.       3  V10.02
  4.       4  102.6                         62.9
  5.       5  861.01
  6.       6  386.01    296.9               93.37
  7.       7  705                            73.09     83.85
  8.       8  V53.32                        78.78     95.1
  9.       9  202.00    754.8     E824.7    04.79
 10.      10  464.11    201.97              46.41
```

It is important to understand that both `icd9 clean` and `icd9p clean` only verify that the variable being cleaned follows the construction rules for the code; it does not check that the code is itself valid. `icd9`[`p`] `check` does that:

```
. icd9p check proc1
(proc1 contains valid ICD-9 procedure codes; 168 missing values)

. icd9p check proc2

proc2 contains invalid codes:
    1.  Invalid placement of period                     0
    2.  Too many periods                                0
    3.  Code too short                                  0
    4.  Code too long                                   0
    5.  Invalid 1st char (not 0-9)                      0
    6.  Invalid 2nd char (not 0-9)                      0
    7.  Invalid 3rd char (not 0-9)                      0
    8.  Invalid 4th char (not 0-9)                      0
   10.  Code not defined                                1
                                              -----------
        Total                                           1
```

Note that diag2 has an invalid code. We could find it using icd9p check, generate() just as we previously found the bad diagnostic code using icd9 check, generate().

icd9[p] can create new variables containing textual descriptions of our diagnostic and procedure codes. For instance,

```
. icd9 gen td1 = diag1, desc
. sort patid
. list patid diag1 td1 in 1/10

       patid  diag1     td1
  1.       1  654.50    cerv incompet preg-unsp
  2.       2  230.6     ca in situ anus nos
  3.       3  V10.02    hx-oral/pharynx malg nec
  4.       4  102.6     yaws of bone & joint
  5.       5  861.01    heart contusion-closed
  6.       6  386.01    meniere dis cochlvestib
  7.       7  705       disorders of sweat gland*
  8.       8  V53.32    ftng autmtc dfibrillator
  9.       9  202.00    ndlr lym unsp xtrndl org
 10.      10  464.11    ac tracheitis w obstruct
```

Note that icd9[p] generate, description does not preserve the sort order of the data (and neither does icd9[p] check unless you specify the any option).

Recall that procedure-code proc2 had an invalid code. Even so, icd9p generate, description is willing to create a textual description variable:

```
. icd9p gen tp2 = proc2, desc
(1 non-missing values invalid and so could not be labeled)
. sort patid
. list patid proc2 tp2 in 1/10

       patid  proc2     tp2
  1.       1
  2.       2  17
  3.       3
  4.       4
  5.       5
  6.       6
  7.       7  83.85     musc/tend lng change nec
  8.       8  95.1      form & structur eye exam*
  9.       9
 10.      10
```

tp2 contains nothing when proc2 is 17 because 17 is not a valid procedure code.

icd9[p] generate can also create variables containing main codes:

```
. icd9 gen main1 = diag1, main
. list patid diag1 main1 in 1/10

       patid  diag1         main1
  1.       1  654.50          654
  2.       2  230.6           230
  3.       3  V10.02          V10
  4.       4  102.6           102
  5.       5  861.01          861
  6.       6  386.01          386
  7.       7  705             705
  8.       8  V53.32          V53
  9.       9  202.00          202
 10.      10  464.11          464
```

icd9p generate, main can similarly generate main procedure codes.

Sometimes one is merely examining an observation:

```
. list diag* if patid==563

     diag1     diag2     diag3
563. 526.4
```

If we wondered what 526.4 was, we could type

```
. icd9 lookup 526.4

1 match found:
    526.4    inflammation of jaw
```

`icd9`[`p`] `lookup` has the ability to list ranges of codes:

```
. icd9 lookup 526/527

12 matches found:
    526      jaw diseases*
    526.0    devel odontogenic cysts
    526.1    fissural cysts of jaw
    526.2    cysts of jaws nec
    526.3    cent giant cell granulom
    526.4    inflammation of jaw
    526.5    alveolitis of jaw
    526.8    other jaw diseases*
    526.81   exostosis of jaw
    526.89   jaw disease nec
    526.9    jaw disease nos
    527      salivary gland diseases*
```

`icd9`[`p`] `search` has the ability to go from description to code:

```
. icd9 search jaw disease

4 matches found:
    526      jaw diseases*
    526.8    other jaw diseases*
    526.89   jaw disease nec
    526.9    jaw disease nos
```

Saved results

`icd9`[`p`] `check` saves scalars `r(e1)`, `r(e2)`, ..., `r(e10)` reporting the number of errors of type 1, 2, ..., 10, and `r(esum)` reporting the total number of errors.

dm77 Removing duplicate observations in a dataset

Duolao Wang, London School of Hygiene and Tropical Medicine, London, UK, duolao.wang@lshtm.ac.uk

Abstract: A command is given that removes duplicated observations in a dataset and retains the unique observations without repetition.

Keywords: Duplicated observations.

Syntax

`unique1 using` *filename*

Description

`unique1` removes the duplicated observations in the current dataset and retains the unique observations without any repetition. The observations are in the same order as the original dataset except that repeated observations are deleted. If *filename* is specified without an extension, `.dta` is assumed.

Remarks

The disk dataset must be a Stata-format dataset; that is, it must have been created using the `save` command.

Examples

You have a dataset stored on disk that you wish to remove the duplicated observations.

```
. use testdata
. list
             id          x          y
  1.          2   01/08/76          A
  2.          2   01/08/76          A
  3.          3   14/04/98          A
  4.          3   14/04/98          B
  5.          3   14/04/98          B
  6.          1   22/01/64          C
  7.          1   22/01/64          C
  8.          1   14/10/87          C

. clear
. unique1 using testdata
. list
           id          x          y
  1.        2   01/08/76          A
  2.        3   14/04/98          A
  3.        3   14/04/98          B
  4.        1   22/01/64          C
  5.        1   14/10/87          C
```

gr34.3 An update to drawing Venn diagrams

Jens M. Lauritsen, County of Fyn, Denmark, jm.lauritsen@dadlnet.dk

Abstract: When John Venn (1834-1923) published his work on logic and developed the “Venn Diagram”, he used circles to indicate the combination of two and three variables and ellipses to show the combination of four variables. The previous version of the `venndiag` routine used squares to represent the combinations. The current update extends the design of the Venn Diagram to use circles or ellipses. Venn diagrams are useful when one wishes to either show overlapping combinations of simultaneous outcomes e.g., displaying which of the allergens birch tree, cat, molds, and so on, make you wheeze on a graph, or when the user wishes to calculate a new variable which reflects those combinations.

Keywords: Venn Diagram, ellipse, multiple-choice answers.

Introduction

When John Venn (1834–1923) published his work on logic and developed the “Venn Diagram”, he used circles to indicate the combination of two and three variables and ellipses to show the combination of four variables. The previous versions of the `venndiag` routine introduced in Lauritsen (1999a, 1999b, 1999c) used squares to represent the combinations. The current update extends the design of the Venn Diagram to use circles or ellipses. The user can specify the desired design as an option.

The syntax has been slightly changed with addition of the design types with options `square`, `ellipse`, and `circle` and two placement options `xoff` and `yoff` which set distances of titles from the top of the diagram and the left margin, respectively. A few adaptations as a consequence of the changed design have been made to other options, as described in the help file for `venndiag`.

New syntax

`venndiag` *varlist* [`if` *exp*] [`in` *range*] [`,` `square` `ellipse` `circle` `label(`*str*`)` `show(`*str*`)` `missing`

`gen(`*varnames*`)` `list(`*variables*`)` `print` `saving(`*filename*`)` `c1(`*#*`)` `c2(`*#*`)` `c3(`*#*`)` `c4(`*#*`)` `noframe`

`nograph` `nolabel` `t1title(`*str*`)` `t2title(`*str*`)` `t3title(`*str*`)` `r1title(`*str*`)` `r2title(`*str*`)` `r3title(`*str*`)`

`r4title(`*str*`)` `r5title(`*str*`)` `r6title(`*str*`)` `pen(`*#*`)` `thick(`*#*`)` `xoff(`*#*`)` `yoff(`*#*`)` `ca(`*#*`)`]

The *varlist* must contain from two to four numerical variables and if generating a variable, that variable must be nonexisting.

New options

`square` shows rectangles as in previous versions.

`ellipse` shows ellipses with two to four variables (this is the default for four variables).

`circle` shows circles (this is the default for two or three variables).

`xoff(`*#*`)` defines the top margin, that is, the distance from the top to `r1title` with a default value of 6000 in Stata’s graphics coordinates.

`yoff(`*#*`)` defines the left margin, that is, the distance from the left to `r1title` with a default value of 22000 in Stata’s graphics coordinates.

`ca(`*#*`)` tells `venndiag` to count on specified value for all variables, e.g., `ca(2)` means to use 2 as the outcome.

Examples

Using examples similar to those in Lauritsen (1999a), we show that the default design for two and three variables is circles as shown in Figures 1 and 2.

```
. venndiag astma season
```

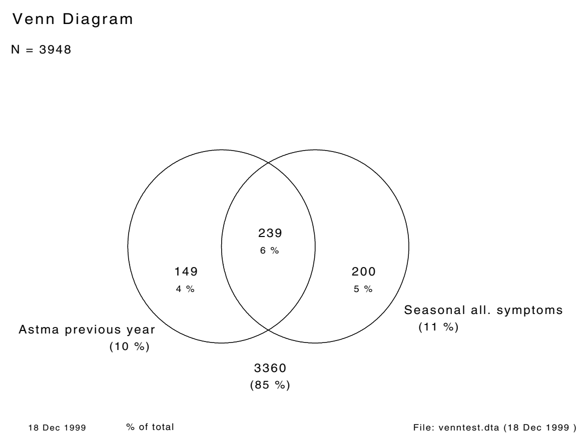


Figure 1. A simple example of two variables.

```
. venndiag astma season eczema, saving(figure2)
```

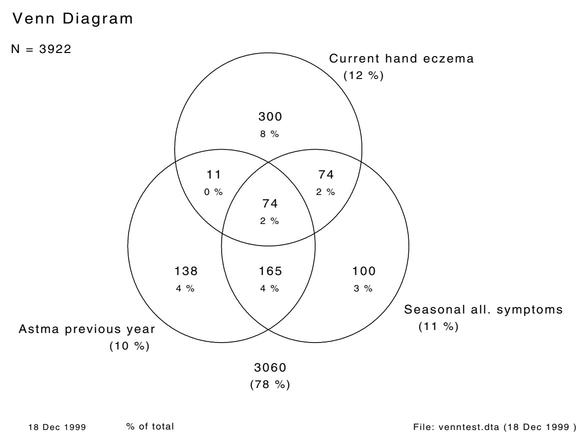


Figure 2. A simple example of three variables.

For four variables, the default is ellipses as shown in Figure 3. Variable labels and percentages are placed in relation to the circle or ellipse which represents each variable. Some experimentation might be needed if you have long labels.

```
. venndiag eczema astma season atopia, ellipse
```

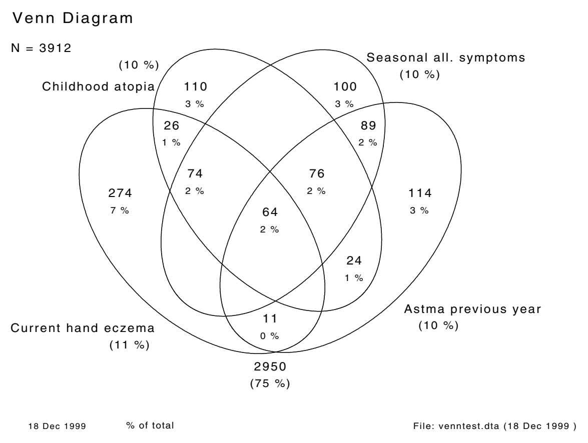


Figure 3. Ellipses used for displaying four variables.

Drawing ellipses

When drawing the ellipses, a procedure similar to the following is used. The program lines for drawing ellipses are actually quite simple. The idea is to first save your own data as a temporary file (before), clear, and generate 1000 (x, y) points based on the formula for an ellipse, draw a graph of this and then finally restore your own data. Try experimenting with the last parameters, which define the shape of the ellipses.

```
program define ellipse        /* draw ellipse on screen  */
version 6
  /* parameters 1: Rotation of ellipse in degrees  2:offset X  3:offset Y  4+5: defines shape of ellipse* /
  tempfile before
  save `before´
  local V = (`1´/360)* 2*_pi
  local lam = `4´        /*size of ellipse ~ length */
  local eps = `5´       /*shape of ellipse ~ if = 0 the result will be a circle*/
  local offx = `2´
  local offy = `3´
  clear
  set obs 1001
  tempvar i x y
  gen `i´ = -_pi+(2*_pi/1000)*(_n-1)
  gen `x´ =  ((1+`eps´)*(`lam´)*cos(`i´))/(1+(`eps´)*cos(`V´-`i´))*100 + `offx´
  gen `y´  = ((1+`eps´)*(`lam´)*sin(`i´))/(1+(`eps´)*cos(`V´-`i´))*100 + `offy´
  gph open
  gph vline `y´ `x´
  gph text 2000 18000 0 -1  Angle in this graph is `1´
  gph text 3500 18000 0 -1  Offset X: `2´   Offset Y: `3´
  gph text 4500 18000 0 -1  Parameter: Size=`4´ Shape=`5´
  gph close
  use `before´, clear
end
ellipse 90 15000 20000 15 0.854
more
ellipse 180 5000 6000 8 0.9
more
ellipse 180 5000 8000 25 0.65
```

Acknowledgments

Martin Villumsen sorted out the mathematics of drawing ellipses in different angles. Thanks to N. Cox who provided the idea for adding circles to a graph and to Alan Riley at Stata Corporation for help on macros and passing values to programs.

References

Lauritsen, J. M. 1999a. gr34: Drawing Venn diagrams. *Stata Technical Bulletin* 47: 3–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 65–71.

——. 1999b. gr34.1: Drawing Venn diagrams. *Stata Technical Bulletin* 48: 2. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 71–72.

——. 1999c. gr34.2: Drawing Venn diagrams. *Stata Technical Bulletin* 49: 8.

gr43 Overlaying graphs

Adrian Mander, MRC Biostatistics Unit, Cambridge, UK, adrian.mander@mrc-bsu.cam.ac.uk

Abstract: This function allows multiple graphs to be displayed on common axes. As any graphical function is allowed, this command can produce graphs for longitudinal data or looking at overlayed histograms.

Keywords: Graphs, stratified graphs.

Syntax

`overlay` *varlist* [`if` *exp*] , `by(`*varlist*`)` [`saving(`*filename*`)` `function(`*str*`)` `ylab(`*numlist*`)` `xlab(`*numlist*`)`

`graph_options`]

Options

`by(`*varlist*`)` specifies the strata for the multiple graphs.

`saving(`*filename*`)` saves the graph as *filename*`.gph`.

`function(`*str*`)` specifies the command that draws the graph. If this is not specified, then the `graph` function is used.

`ylab(`*numlist*`)` specifies axes labels.

`xlab(`*numlist*`)` specifies axes labels.

Description

This function draws several graphs in one area of the graphics window. As a result this function is very versatile and will work well with any graph function that allows the user to specify the axes. The function will, by default, try to calculate axes that remain unchanged for each graph, this may fail and the user then has to specify the axes using `xlab` and `ylab`.

Any options for the graphing function can be added to the end of the command line. These can be options such as the plotting symbol and connecting points.

Examples

Data is taken from a clinical trial that measures peak flow for asthma sufferers over time. To plot individual lines per person through time is achieved by

```
. overlay pef day0, by(patient) c(l) s(.) sort saving(graph1)
```

which produces the graph in Figure 1.

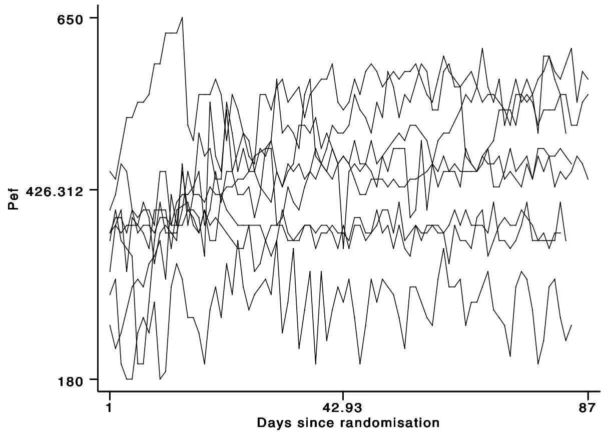


Figure 1. Plotting lines for several people in a clinical trial.

The *varlist* is passed directly to `graph` so `pef` is on the y-axis.

To illustrate the use of `kdensity` instead of `graph`, consider

```
. overlay pef if patient<5028, by(patient) function(kdensity) xlab(150,350,680) ylab(0,0.02, 0.045) s(.)
```

For `kdensity` it was necessary to specify the axes since otherwise the graph would be incorrect. Figure 2 shows the kernel density estimates for 4 patients.

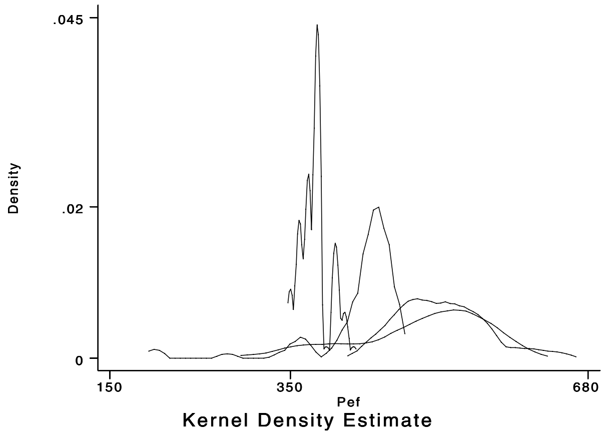


Figure 2. Kernel density estimates for four people.

Note that `overlay` can even overlay histograms although this may seem a little confusing in black and white. Consider Figure 3 which results from

```
. overlay pef if patient<5028, by(patient) xlab(150,350,680) bin(8) ylab(0,1.1)
```

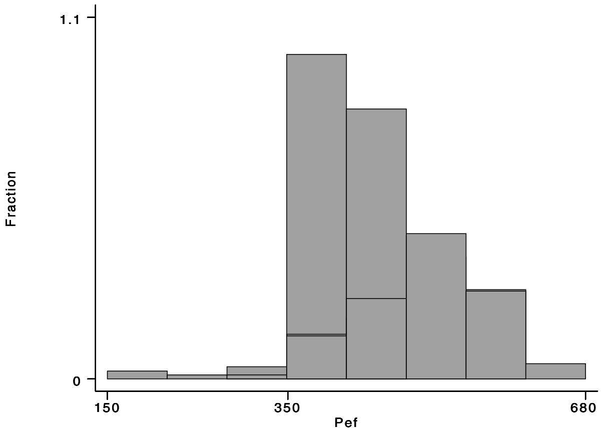


Figure 3. Overlaying histograms.

ip29.1 Metadata for user-written contributions to the Stata programming language: extensions

Nicholas J. Cox, University of Durham, UK, n.j.cox@durham.ac.uk
Christopher F. Baum, Boston College, baum@bc.edu

Abstract: The `archutil` package published in STB-52 for working with files or packages in the Statistical Software Components archive has been extensively revised. `archlist` has been superseded by `archdesc`, which offers additional features and incorporates a correction regarding behavior when logging. A new component, `archinst`, allows the user to install a package from the archive in one command.

Keywords: SSC-IDEAS, Statalist, internet, files, packages, `archutil`.

The `archutil` package published by Baum and Cox (1999) has been extensively revised. The original version contained utilities `archlist`, `archtype` and `archcopy`. `archlist` has been superseded by `archdesc`, which offers additional features and incorporates a correction regarding behavior when logging. A new component, `archinst`, allows the user to install a package from the archive in one command.

These commands work with files or packages from the Statistical Software Components (SSC) archive (often called the Boston College archive). They require a net-aware variant of Stata 6.0.

Syntax

`archdesc` [{ *package* | *letter* }] [`using` *filename*] [`, replace` <u>`no`</u>`log`]

`archinst` *package* [`,` *net_install_options*]

`archcopy` *filename.ext* [`,` *copy_options*]

`archtype` *filename.ext*

Description

`archdesc` describes the contents of the archive.

`archdesc`, with neither a letter nor a package specified, lists all packages in the archive. By default, it also puts a log of the listing in `ssc-ideas.lst`.

`archdesc` *letter* (where *letter* is one of `a-z` or `_`) lists all packages in the archive whose names begin with that letter.

`archdesc` *package* (where *package* is a name two or more letters long beginning with `a-z` or `_`) describes that package if it exists; or all packages beginning with the same letter if it does not. Thus a faulty guess still produces information on nearby names.

If `archdesc` is accompanied by logging results to a file, any existing logging is temporarily suspended.

`archinst` *package* installs that package from the archive.

`archcopy` *filename.ext* copies *filename.ext* from the SSC archive to the appropriate directory or folder within `STBPLUS`, determined automatically. (If curious, type `sysdir` to see where this is.) This is appropriate for individual `.ado` or `.hlp` files. `archcopy` is rarely needed, given `archinst`.

`archtype` *filename.ext* types *filename.ext* from the SSC archive. This is appropriate for individual `.ado` or `.hlp` files.

Options

`replace` specifies that *filename* is to be overwritten.

`nolog` overrides the default behavior of `archdesc`, with no specification of either a letter or a package, which is to log to `ssc-ideas.lst`.

net_install_options are options of `net install`. See help on `net` or [R] **net**.

copy_options are options of `copy`. See help on `copy` or [R] **copy**.

archdesc and logging

Depending on how it is called, `archdesc` varies in whether it echoes results to a log file by default.

`archdesc` by itself will produce quite lengthy output (as of January 2000, more than 600 lines). Such output may be too much to scan visually with ease, and it has some value as a reference source. The default is therefore that output will be echoed to a log file. This default can be overridden with the `nolog` option.

In contrast, `archdesc` with a letter or package name produces much less output, which will not be logged to a file unless explicitly requested.

Logging here refers to opening a log file for `archdesc` results and closing it afterwards, which are all handled automatically by `archdesc`. Any existing logging is temporarily suspended.

However, if you are already logging to a file, and wish the results of `archdesc` to be included in the log with other results of your session, then that is achieved by issuing either `archdesc, nolog` or `archdesc` *whatever* within your session. The earlier opening and (if desired) later closing of the log are the user's responsibility, as usual.

archdesc and archlist

`archdesc` supersedes `archlist`, documented by Baum and Cox (1999).

`archlist` as published by Baum and Cox (1999) would not resume logging to a log file previously being used if there was a problem with the `using` subcommand. Suppose, for example, that a user had typed

```
. log using log1
...
. archlist using log2
```

and `log2.log` already existed. The correct syntax would have been

```
. archlist using log2, replace
```

The syntax error would have halted the program, but logging to `log1.log` would not have been resumed.

`archdesc` handles this problem more gracefully. In addition, a corrected version of `archlist` is included on the electronic media (floppy disk or website copies) accompanying this insert, even though users are recommended to switch to `archdesc`.

Examples

In the examples below the somewhat lengthy output of these commands is suppressed here to save space.

```
. archdesc using ssc.txt, replace
. archdesc w
. archdesc whitetst using whitetst.txt
. archinst whitetst
. archcopy whitetst.ado
. archcopy whitetst.hlp
. archtype whitetst.hlp
```

Acknowledgments

Helpful advice was received from Bill Gould, Jens Lauritsen, Vince Wiggins, and Desmond Williams.

Reference

Baum, C. F. and N. J. Cox. 1999. ip29: Metadata for user-written contributions to the Stata programming language. *Stata Technical Bulletin* 52: 10–12.

sbe32 Automated outbreak detection from public health surveillance data

López Vizcaíno, M. E.; Santiago Pérez, M. I.; Abraira García, L.; Dirección Xeral de Saude Publica, Spain, dxsp3@jet.es

Abstract: The early detection of outbreaks in epidemiological surveillance is an important challenge in order to introduce effective control measures. In this insert, we adapt and program an algorithm developed by Farrington et al. (1996) to process weekly reports of infectious diseases, which is based on a loglinear regression model. The output is a threshold value for the current week above which the observed count is declared to be unusual.

Keywords: Outbreak, regression, threshold, public health surveillance.

Introduction

Epidemiological surveillance is the systematic collection, analysis, and interpretation of data for public health purposes. One of its aims is the early detection of outbreaks in order to introduce effective control measures. Many available methods for this purpose are based on parametric procedures, which compare actual numbers of cases with a warning threshold calculated from historical data. The statistical methodology to do the detection of unusual disease clusters must cope with several difficulties as fluctuations in the historical data series may be due to seasonal cycles and secular trends, and by past outbreaks. In addition, the method must be sufficiently robust to accommodate a wide range of microorganisms. The available methodology is reviewed in Farrington et al. (1996). In this paper, the authors developed an automated procedure to process weekly reports of infectious diseases, which is based on a loglinear regression model, adjusted for overdispersion, seasonality, secular trends, and past outbreaks. The model is used to calculate an expected value for the current week based on historical data, together with a threshold value above which an observed count is declared to be unusual. The baseline data to fit the regression model are specified by the following mechanism: if the current week is t_0, only data from weeks $t_0 - 3$ to $t_0 + 3$ from the previous five years are included. In this insert, we present a program to calculate threshold values using a modified version of Farrington's algorithm. The data are weekly reports of infectious disease from a passive surveillance system based on laboratory reporting.

Methodology

The baseline count y_i is assumed to be generated by a Poisson-like process, except that the variation is greater than that of a true Poisson for some organisms. In this case, negative binomial regression is used to estimate the model for the weekly counts from historical data. We assume a serial correlation between baseline counts within the same year and independence otherwise. The model fitted is

$$y_i \sim \text{Poisson}(g_i)$$

$$g_i = \exp(\alpha + \beta t_i + \delta n_i + u_i) = \exp(\alpha + \beta t_i + \delta n_i)\exp(u_i) = m_i e_i$$

where e_i is the random effect of the model, and μ_i is the systematic component. The random effect e_i is assumed to follow a gamma distribution with mean one and variance $(\phi - 1)/\mu_i$, ϕ being the overdispersion parameter:

$$e_i \sim \text{Gamma}\left(\frac{\mu_i}{\phi - 1}, \frac{\mu_i}{\phi - 1}\right)$$

resulting in the negative binomial distribution with mean μ_i and variance $\phi\mu_i$ for the baseline count y_i. The Poisson model corresponds to $\phi = 0$, while μ_i, the systematic component, can be modeled as

$$\log \mu_i = \alpha + \beta t_i + \delta n_i$$

where βt_i is a linear time trend that is omitted if not significant, and δn_i adjusts the geographic effect in reporting. This is an additional component to the model used in Farrington's algorithm. Moreover, we have introduced several modifications related to the estimation procedure. The variables included in the model are y_i, the number of cases reported at week i, t_i, the time measured in weeks, and n_i, the number of hospitals reporting cases at week i.

The model yields a 99% prediction interval for the current week, and the threshold value is calculated as the upper limit of that interval. When no cases are reported in a week, we assume that no outbreak occurred and thus no model is fitted. As a consequence, no threshold is calculated.

The output of the program is a table displaying the list of microorganisms with the observed number of cases and the threshold value for the current week. In addition, a warning message is displayed when the actual report exceeds the threshold.

Syntax

`obvset` [*var1 var2 var3 var4 var5*]

`outbrk` #$_{\text{week}}$ #$_{\text{year}}$

where *var1* is the numerical variable of reports, *var2* is the numerical variable identifying the week, *var3* is the numerical variable identifying the year, *var4* is the numerical variable with the number of hospitals reporting the cases, and *var5* is the string variable containing the name of the microorganisms.

The arguments $\#_{\text{week}}$ and $\#_{\text{year}}$ are, respectively, the number of the weeks and years in which we want to detect if an outbreak has occurred. `outbrk` works after setting the variables with `obvset`.

Description

`outbrk` calculates threshold values for outbreak detection of infectious diseases based on historical data. It was developed for data consisting of weekly reports of positive microbiological diagnostics from a passive surveillance system based on laboratory reporting.

`outbrk` can be used for outbreak detection within other surveillance systems of communicable diseases weekly reporting.

`obvset` doesn't allow the user to save these settings with the dataset. When exiting Stata, the current settings are cleared. `obvset` will be helpful if you need to run `outbrk` for different weeks. Without arguments, `obvset` displays current settings, if any.

Note that `outbrk` uses `poisml` introduced in Hilbe (1998).

Example

We illustrate the use of `outbrk` with salmonella data from the National Microbiological Reporting System (SIM). The data consist of weekly reports of serotyping salmonella species, one of the most common reported cause of gastrointestinal infection, from the above surveillance system within the period 1992–1998. In this example, we apply `outbrk` for the detection of the possibility of existence of outbreaks due to different salmonella serotypes in the third week of the year 1998. First, we describe the dataset:

```
. describe

Contains data from salmo.dta
  Microbiological weekly reports of salmonella
  obs:         3,360
  vars:            5  size:        164,7
----------------------------------------------------------------------
   1. organism  str25  %25s                   microorganism name
   2. year      float  %6.0g                  year identify number
   3. week      float  %6.0g                  week identify number
   4. counts    float  %6.0g                  number of cases reported
   5. nhosp     float  %6.0g                  number of hospitals
                                              reporting
----------------------------------------------------------------------
```

Typing `obvset` without arguments, we verify that no variables have been set. Therefore, we have to set the variables by typing

```
. obvset counts week year nhosp organism
```

Now, if we type `obvset` without arguments:

```
. obvset

Reports count is:COUNTS
Week identifier is:WEEK
Year identifier is:YEAR
Hospitals count is:NHOSP
Organism identifier is:ORGANISM
```

After setting the variables, we can use `outbrk`:

```
. outbrk 3 1998
YEAR 1998; WEEK 3
-----------------+----------------------------------
        Organism |    Reports   Threshold     Warning
-----------------+----------------------------------
   S.enteritidis |         17   34.76               -
      S.infantis |          0                       -
   S.typhimurium |         19   18.29         Warning
      S.virchow |          0                       -
 Salmonella gr.B |          6   17.20               -
 Salmonella gr.C |          0                       -
Salmonella gr.C1 |          0                       -
Salmonella gr.C2 |          1   3.01                -
 Salmonella gr.D |          2   6.91                -
  Salmonella sp. |         15   27.59               -
-----------------+----------------------------------
```

This table shows the different salmonella serotypes list, the reports in the third week of 1998, the calculated threshold value, and a warning message if the reported counts exceed that value. In this week, the number of cases reported for Salmonella typhimurium exceeds the threshold value, so a further epidemiological investigation is needed to check if this warning is an outbreak. There are no counts reported for S. Infantis, S. Virchow, Salmonella gr. C and Salmonella gr. C1; therefore no threshold value was calculated. This detection system provides epidemiologists with a tool for use in conjunction with other surveillance methods. Its main function is to focus attention on a potential outbreak, which is especially valuable when large numbers of different microorganisms are reported each week.

Acknowledgments

This work was presented at the First Iberian Stata User's Group meeting, which was held the 20th and 21st of May in Cordoba, Spain. Thanks to Aurelio Tobias for helpful comments. The data in the example are from the National Microbiological Reporting System.

References

Farrington, C. P., N. J. Andrews, A. D. Beale, and M. A. Catchpole. 1996. A statistical algorithm for the early detection of outbreaks of infectious disease. *Journal of the Royal Statistical Society*, Series A 159: 547–563.

Hilbe, J. 1998. sg91: Robust variance estimators for MLE Poisson and negative binomial regression. *Stata Technical Bulletin* 45: 26–28. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 177–180.

sg84.2 | Concordance correlation coefficient: update for Stata 6

Thomas J. Steichen, RJRT steicht@rjrt.com
Nicholas J. Cox, University of Durham, UK, n.j.cox@durham.ac.uk

Abstract: The program for concordance correlation previously published in STB-43 and STB-45 has been updated to the syntax of Stata 6.0 and corrected for some deficiencies, principally to do with graphics and speed of calculation. A new option now permits the saving of the standard normal plot.

Keywords: Concordance correlation, graphics, measurement comparison.

Description

`concord` computes Lin's (1989) concordance correlation coefficient, ρ_c, for agreement on a continuous measure obtained by two persons or methods and provides an optional graphical display of the observed concordance of the measures. `concord` also provides statistics and optional graphics for Bland and Altman's (1986, 1995) limits-of-agreement, *loa*, procedure. The *loa*, a data-scale assessment of the degree of agreement, is a complementary approach to the relationship-scale approach of ρ_c.

This insert documents enhancements and changes to `concord` and provides the syntax needed to use a new feature. A full description of the method and of the operation of the original command and options was given by Steichen and Cox (1998a). A few revisions were documented later by Steichen and Cox (1998b). This updated program does not change the implementation of the underlying statistical methodology, or modify the original operating characteristics of the program; rather, it follows the syntax changes of Stata version 6.0.

Syntax

`concord` *var1 var2* [*weight*] [`if` *exp*] [`in` *range*] [`,` `by(`*byvar*`)` `summary` `level(`*#*`)` `graph({ccc | loa})`

`noref reg npsaving(`*filename* [`,` *replace*]`)` `nosnd(`*snd_var* [`,` *replace*]`)` *graph_options*]

New option

`npsaving(`*filename* [`, replace`]`)` saves the standard normal plot generated by `graph(loa)`. The *filename* is assumed to have extension `gph`. If *filename* does not exist, it is created. If *filename* exists, an error will occur unless `replace` is also specified. This option is ignored if `graph(loa)` is not requested. Note that the usual `saving()` option saves the loa plot itself when `graph(loa)` is specified (and the concordance plot when `graph(ccc)` is specified).

Explanation

The primary purpose of this version is to revise `concord` to meet and to exploit syntax changes in Stata 6. In addition, some deficiencies in the previous implementation have been corrected.

First, `concord` previously failed when attempting a `saving()` of the loa plot generated by the `graph(loa)` option. This has been fixed. Second, the program did not allow the standard normal plot, which is also generated by the `graph(loa)` option, to be saved. The new `npsaving()` option now allows that. Third, it did not allow variable labels to appear on the axes of the loa graph in place of variable names. They will now appear if they are defined. Fourth, a few minor changes have been made to speed up calculation.

A consequence of updating to Stata 6 is that the workarounds `t1title(".")` and `t2title(".")` to blank out default titles are no longer required. Blanking out can now be obtained directly by, for example, `t1title(" ")` and the previous workarounds now work literally, placing a period in the requested title.

Saved Results

The system `S_#` macros are unchanged. In addition, the saved results are returned in `r()`. Specifically, if the `by()` option is not used, `concord` saves:

`S_1`	`r(N)`	number of observations compared	`S_7`	`r(z_tr_ul)`	upper CI limit (z-transform)
`S_2`	`r(rho_c)`	concordance correlation coefficient, $\hat{\rho}_c$	`S_8`	`r(C_b)`	bias-correction factor, C_b
`S_3`	`r(se_rho_c)`	standard error of $\hat{\rho}_c$, $\sigma_{\hat{\rho}_c}$	`S_9`	`r(diff)`	mean difference
`S_4`	`r(asym_ll)`	lower CI limit (asymptotic)	`S_10`	`r(sd_diff)`	standard deviation of mean difference
`S_5`	`r(asym_ul)`	upper CI limit (asymptotic)	`S_11`	`r(LOA_ll)`	lower limit-of-agreement value
`S_6`	`r(z_tr_ll)`	lower CI limit (z-transform)	`S_12`	`r(LOA_ul)`	upper limit-of-agreement value

References

Bland, J. M. and D. G. Altman. 1986. Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet* I: 307–310.

——. 1995. Comparing methods of measurement: why plotting difference against standard is misleading. *Lancet* 346: 1085–1087.

Lin, L. I-K. 1989. A concordance correlation coefficient to evaluate reproducibility. *Biometrics* 45: 255–68.

Steichen, T. J. and N. J. Cox. 1998a. sg84: Concordance correlation coefficient. *Stata Technical Bulletin* 43: 35–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 137–143.

——. 1998b. sg84.1: Concordance correlation coefficient, revisited. *Stata Technical Bulletin* 45: 21–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 143–145.

sg116.1 Update to hotdeck imputation

Adrian Mander, MRC Biostatistics Unit, Cambridge, adrian.mander@mrc-bsu.cam.ac.uk
David Clayton, MRC Biostatistics Unit, Cambridge, david.clayton@mrc-bsu.cam.ac.uk

Abstract: Two additional options have been added to the `hotdeck` command.

Keywords: Hotdeck imputation method.

Two additional options have been added to the `hotdeck` command introduced in Mander and Clayton (1999).

New options

`seed(`*#*`)` specifies the random number generator seed.

`infiles(`*filename filename ...*`)` specifies a list of files that have missing values replaced by imputed values. The `infiles` option allows the user to analyze several imputed datasets that have been created by other programs.

Reference

Mander, A. and D. Clayton. 1999. sg116: Hotdeck imputation. *Stata Technical Bulletin* 51: 32–34.

sg120.2 Correction to roccomp command

Mario Cleves, Stata Corporation, mcleves@stata.com

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

A bug was discovered in the `roccomp` program when more than two modalities were being compared and the modalities were not specified in alphabetical order. The output table reordered the modality variable names placing them in alphabetical order. This could result in the wrong modalities being compared and incorrect significant probabilities reported. This has been corrected. The output table will now present results for each modality in the same order as specified on the command line.

Reference

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

sg130 Box–Cox regression models

David M. Drukker, Stata Corporation, ddrukker@stata.com

Abstract: This article describes the `boxcox2` command which obtains maximum likelihood estimates for the parameters from any of four distinct Box–Cox regression models. The article also includes a brief introduction to the four Box–Cox regression models. Several examples are used to illustrate how this command can be run and how to interpret the output.

Keywords: Box–Cox regression, nonlinear regression, flexible functional form, specification test.

Syntax

`boxcox2` *depvar* [*indepvars*] [*weight*] [`if` *exp*] [`in` *range*] [`,` `model(`<u>`lhs`</u>`only`|<u>`rhs`</u>`only`|<u>`lam`</u>`bda`|`theta)`

<u>`notr`</u>`ans(`*varlist*`)` `lrtest` `from(`*init_specs*`)` <u>`noc`</u>`onstant` `nolog` `nologlr` <u>`iter`</u>`ate(`*#*`)` <u>`l`</u>`evel(`*#*`)`]

`fweight`s, and `iweight`s are allowed; see [U] **14.1.6 weight**.

`boxcox2` shares the features of all estimation commands; see [U] **23 Estimation and post-estimation commands**.

Syntax for predict

`predict` [*type*] *newvarname* [`if` *exp*] [`in` *range*] [`,` { `yhat` | `xbt` | <u>`res`</u>`iduals` }]

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Description

`boxcox2` finds the maximum likelihood estimates of the parameter(s) of the Box–Cox transform, the coefficients on the independent variables, and the standard deviation of the normally distributed errors for a model in which *depvar* is regressed on *indepvars*. The user has the option of estimating

Option	Estimates
`lhsonly`	$y_j^{(\theta)} = \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + \epsilon_j$
`rhsonly`	$y_j = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \epsilon_j$
`rhsonly notrans()`	$y_j = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$
`lambda`	$y_j^{(\lambda)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \epsilon_j$
`lambda notrans()`	$y_j^{(\lambda)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$
`theta`	$y_j^{(\theta)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \epsilon_j$
`theta notrans()`	$y_j^{(\theta)} = \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$

Any transformed variable must be strictly positive.

Note: this command estimates a superset of the models accommodated by the `boxcox` command of official Stata. See [R] **boxcox** for information on the official command.

Options

`model(lhsonly|rhsonly|lambda|theta)` specifies which of the four models to fit.

`model(lhsonly)` applies the Box–Cox transform to *depvar* only. `model(lhsonly)` is the default value.

`model(rhsonly)` causes the transform to be applied to the *indepvars* only.

`model(lambda)` causes the transform to be applied to both *depvar* and *indepvars*, and they are transformed by the same parameter.

`model(theta)` causes the transform to be applied to both *depvar* and *indepvars*, but this time each side is transformed by a separate parameter.

`notrans(`*varlist*`)` specifies that the variables in *varlist* are to be included as nontransformed independent variables.

`lrtest` specifies that a likelihood-ratio test of significance is to be performed and reported for each independent variable.

`from()` allows the user to specify the initial values for Box–Cox transformation parameter(s); see [R] **maximize**.

Model	Initial value specification
`lhsonly`	from(θ_0, copy)
`rhsonly`	from(λ_0, copy)
`lambda`	from(λ_0, copy)
`theta`	from(λ_0 θ_0, copy)

`noconstant` suppresses the constant term (intercept) in the model.

`nolog` suppresses the iteration log when estimating the full model.

`nologlr` suppresses the iteration log when estimating the restricted models required by the `lrtest` option. If `nologlr` is specified when `lrtest` is not, then it is ignored.

`iterate(#)` specifies the maximum number of iterations that the maximum likelihood optimizer will undertake in search of a solution.

`level(#)` specifies the confidence level, in percent, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] **23.5 Specifying the width of confidence intervals**.

Options for predict

`yhat` calculates the predicted value of y.

`xbt`, the default, calculates the "linear" prediction. For all the models except `model(lhsonly)`, all the *indepvars* are transformed.

`residuals` calculates the residuals after the predicted value of y has been subtracted from the actual value.

Remarks

The Box–Cox transform

$$y^{(\lambda)} = \frac{y^\lambda - 1}{\lambda}$$

has been widely used in applied data analysis. Box and Cox (1964) developed the transformation and argued that the transformation could make the residuals more closely normal and less heteroscedastic. Cook and Weisberg (1982) discuss the transform in this light. Since the transform embeds several popular functional forms, it has received some attention as a method for testing functional forms. In particular,

$$y^{(\lambda)} = \begin{cases} y - 1 & \text{if } \lambda = 1 \\ \ln(y) & \text{if } \lambda = 0 \\ 1 - 1/y & \text{if } \lambda = -1 \end{cases}$$

Davidson and MacKinnon (1993) discuss this use of the transform. Atkinson (1985) also gives a good general treatment.

Theta model

`boxcox2` obtains the maximum likelihood estimates of the parameters for four different models. The most general of the models, the `theta` model, is

$$y_j^{(\theta)} = \beta_0 + \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \ldots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$$

where $\epsilon \sim N(0, \sigma^2)$. Here the dependent variable y is subject to a Box–Cox transform with parameter θ. Each of the *indepvars* $x_1, x_2, \ldots, x_k$ is transformed by a Box–Cox transform with parameter λ. The $z_1, z_2, \ldots, z_l$, specified in the `notrans()` option, are independent variables that are not transformed.

Box and Cox (1964) argued that this transformation would leave behind residuals that more closely resemble a normal distribution than those produced by a simple linear regression model. Users should bear in mind that the normality of ϵ is assumed and that `boxcox2` obtains maximum likelihood estimates of the $k + l + 4$ parameters under this assumption. `boxcox2` does not choose λ and θ so that the residuals are approximately normally distributed. Users interested in this type of transformation to normality should see the official Stata commands `lnskew0` and `bcskew0` in [R] **lnskew0**. However, these commands work on a more restrictive model in which none of the independent variables are transformed.

Example

Consider an example using the auto data.

```
. boxcox2 mpg weight price, notrans(foreign) model(theta) lrtest
Estimating comparison model

Iteration 0:   log likelihood = -234.39434
Iteration 1:   log likelihood = -228.26891
Iteration 2:   log likelihood = -228.26777
Iteration 3:   log likelihood = -228.26777

Estimating full model

Iteration 0:   log likelihood = -194.13727
Iteration 1:   log likelihood = -184.34212
Iteration 2:   log likelihood = -180.18783
Iteration 3:   log likelihood =  -177.5195
Iteration 4:   log likelihood = -176.08846
Iteration 5:   log likelihood = -175.67353
Iteration 6:   log likelihood = -175.67343
Iteration 7:   log likelihood = -175.67343

Estimating comparison models for LR tests

Iteration 0:   log likelihood = -179.58214
Iteration 1:   log likelihood = -177.59036
Iteration 2:   log likelihood = -177.58739
Iteration 3:   log likelihood = -177.58739

Iteration 0:   log likelihood = -203.92855
Iteration 1:   log likelihood = -201.30202
Iteration 2:   log likelihood = -201.18246
Iteration 3:   log likelihood = -201.18233
Iteration 4:   log likelihood = -201.18233

Iteration 0:   log likelihood = -178.83799
Iteration 1:   log likelihood = -175.98405
Iteration 2:   log likelihood = -175.97931
Iteration 3:   log likelihood = -175.97931


                                                  Number of obs   =         74
                                                  LR chi2(4)      =     105.19
Log likelihood = -175.67343                       Prob > chi2     =      0.000
------------------------------------------------------------------------------
         |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 /lambda |   .7601691   .6289991      1.209   0.227      -.4726464    1.992985
  /theta |  -.7189315   .3244439     -2.216   0.027       -1.35483   -.0830332
------------------------------------------------------------------------------

Estimates of scale-variant parameters
----------------------------------------------------------
         |      Coef.   chi2(df)   P>chi2(df)  df of chi2
---------+------------------------------------------------
Notrans  |
 foreign |  -.0114338     3.828     0.050         1
   _cons |   1.377399
---------+------------------------------------------------
Trans    |
  weight |   -.000239    51.018     0.000         1
   price |  -6.18e-06     0.612     0.434         1
---------+------------------------------------------------
  /sigma |   .0138489
----------------------------------------------------------

---------------------------------------------------------------
   Test              Restricted
    H0:            log likelihood      X~chi2        Pr > chi2
---------------------------------------------------------------
theta=lambda = -1     -181.64479       11.94           0.001
theta=lambda =  0      -178.2406        5.13           0.023
theta=lambda =  1     -194.13727       36.93           0.000
---------------------------------------------------------------
```

The output is composed of the iteration logs and three distinct tables. The first table contains a standard header for a maximum likelihood estimator and a standard output table for the Box–Cox transform parameters. The second table contains the estimates of the scale-variant parameters. The third table contains the output from likelihood-ratio tests on three standard functional form specifications.

If we were to interpret this output, the right-hand-side transformation does not significantly add to the regression while the left-hand-side transformation makes the 5% but not the 1% cutoff. `price` is certainly not significant and foreign lies right on the 5% cutoff. `weight` is clearly significant. The output also says that both the linear and multiplicative inverse specifications are strongly rejected. A natural log specification can be rejected at the 5% but not the 1% level.

Technical Note

Spitzer (1984) showed that the Wald statistics of whether the coefficients of the right-hand-side variables, transformed or untransformed, are significantly different than zero are not invariant to changes in the scale of the transformed dependent variable. Davidson and MacKinnon (1993) also discuss this point. It is worth noting that this problem is an example of the manipulability of Wald statistics in nonlinear models. Lafontaine and White (1986) analyze this problem numerically and Phillips and Park (1988) analyze it analytically using Edgeworth expansions. See Drukker (2000) for a more detailed discussion of this issue. Since the parameter estimates and their Wald tests are not scale invariant, no Wald tests or confidence intervals are reported for these parameters. However, when the `lrtest` option is specified, likelihood-ratio tests are performed and reported. Schlesselman (1971) showed that, if a constant is included in the model, then the parameter estimates of the Box–Cox transforms are scale invariant. For this reason, it is highly recommended that the `noconstant` option not be used.

The `lrtest` option does not perform a likelihood-ratio test on the constant. Hence, no value for this statistic is reported. Unless the data are properly scaled, the restricted model frequently does not converge. For this reason, no likelihood-ratio test on the constant is performed by the `lrtest` option. However, if a user has a special interest in performing this test, then it can be done by estimating the constrained model separately. If problems with convergence are encountered, rescaling the data by their means may help.

Lambda model

A less general model than the one above is called the `lambda` model. It specifies that the same parameter be used in both the left-hand side and right-hand side transformations. Specifically,

$$y_j^{(\lambda)} = \beta_0 + \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \cdots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$$

where $\epsilon \sim N(0,\sigma^2)$. Here the *depvar* variable y and each of the *indepvars* $x_1, x_2, \ldots, x_k$ are transformed by a Box–Cox transform with the common parameter λ. Again, the $z_1, z_2, \ldots, z_l$ are independent variables that are not transformed.

Example

Again using the auto data we have

```
. boxcox2 mpg weight price, notrans(foreign) model(lambda) lrtest nolog nologlr
Estimating comparison model

Estimating full model

Estimating comparison models for LR tests

                                                  Number of obs   =         74
                                                  LR chi2(3)      =     102.21
Log likelihood = -177.16463                       Prob > chi2     =      0.000

------------------------------------------------------------------------------
      mpg|      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 /lambda |  -.3188704   .2185908     -1.459   0.145      -.7473006    .1095597
------------------------------------------------------------------------------

Estimates of scale-variant parameters
----------------------------------------------------------
         |      Coef.   chi2(df)   P>chi2(df)  df of chi2
---------+------------------------------------------------
Notrans  |
 foreign |  -.0271361     1.924     0.165         1
   _cons |   15.11529
---------+------------------------------------------------
Trans    |
  weight |  -3.964759    48.047     0.000         1
   price |  -.5849437     2.478     0.115         1
---------+------------------------------------------------
  /sigma |   .0748609
----------------------------------------------------------
```

```
---------------------------------------------------------
   Test         Restricted     LR statistic      P-Value
    H0:       log likelihood      X~chi2        Pr > chi2
---------------------------------------------------------
lambda = -1     -181.64479         8.96           0.003
lambda =  0      -178.2406         2.15           0.142
lambda =  1     -194.13727        33.95           0.000
---------------------------------------------------------
```

The options `nolog` and `nologlr` were specified to suppress the iteration logs. Aside from this change, the output of this example has the same outline as that of the previous one. The most important change is in the first table. Since the requested model has only one Box–Cox transform parameter, only one is reported. The interpretation is similar to the previous case.

Left-hand-side only model

More restrictive still than a common transformation parameter is transforming the dependent variable only. Since the dependent variable is on the left hand side of the equation, this model is known as the `lhsonly` model. In this case, one is estimating the parameters of the model

$$y_j^{(\theta)} = \beta_0 + \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + \epsilon_j$$

where $\epsilon \sim N(0, \sigma^2)$. In this case only the *depvar*, y, is transformed by a Box–Cox transform with the parameter θ. Note that the $z_1, z_2, \ldots, z_L$ have been dropped from the specification. Since the *indepvars* $x_1, x_2, \ldots, x_K$ are not transformed, the $z_1, z_2, \ldots, z_L$ are superfluous.

This is the model that is estimated by Stata's `boxcox` command. Even here, `boxcox2` offers some advantages over `boxcox`. In particular, one can easily obtain likelihood-ratio tests of the significance of the independent variables. In contrast, `boxcox` offers Wald statistics that use variance estimates of the coefficients which are conditional on θ. This difference is important. Spitzer (1984) shows that the variance estimates conditional on θ will underestimate the true variance.

Example

In this example, `mpg` is again hypothesized to be a function of `weight`, `price` and `foreign` in a Box–Cox model in which only `mpg` is subject to the transform.

```
. boxcox2 mpg weight price foreign, model(lhs) lrtest  nolog nologlr
Estimating comparison model
Estimating full model
Estimating comparison models for LR tests

                                                  Number of obs   =         74
                                                  LR chi2(3)      =     105.04
Log likelihood = -175.74705                       Prob > chi2     =      0.000

------------------------------------------------------------------------------
      mpg|      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  /theta |  -.7826999    .281954     -2.776   0.006       -1.33532   -.2300802
------------------------------------------------------------------------------

Estimates of scale-variant parameters
---------------------------------------------------------
         |      Coef.   chi2(df)   P>chi2(df)  df of chi2
---------+-----------------------------------------------
Notrans  |
  weight |  -.0000294    58.056     0.000         1
   price |  -4.66e-07     0.469     0.493         1
 foreign |  -.0097564     4.644     0.031         1
   _cons |   1.249845
---------+-----------------------------------------------
  /sigma |    .013249
---------------------------------------------------------

---------------------------------------------------------
   Test         Restricted     LR statistic      P-Value
    H0:       log likelihood      X~chi2        Pr > chi2
---------------------------------------------------------
theta = -1      -176.04312         0.59           0.442
theta =  0      -179.54104         7.59           0.006
theta =  1      -194.13727        36.78           0.000
---------------------------------------------------------
```

It is worth noting that this model rejects both the linear and log specification of `mpg` but fails to reject that $1/$`mpg` is linear in the independent variables. These findings are in line with the what an engineer would have expected *ex ante*. In engineering terms, gallons per mile represent actual energy consumption and energy consumption should be approximately linear in weight.

Right-hand-side only model

The fourth model leaves the *depvar* alone and transforms a subset of the *indepvars* using the parameter λ. This is the `rhsonly` model. In this model the *depvar*, y, is given by

$$y_j = \beta_0 + \beta_1 x_{1j}^{(\lambda)} + \beta_2 x_{2j}^{(\lambda)} + \ldots + \beta_k x_{kj}^{(\lambda)} + \gamma_1 z_{1j} + \gamma_2 z_{2j} + \cdots + \gamma_l z_{lj} + \epsilon_j$$

where $\epsilon \sim N(0,\sigma^2)$. Here each of the *indepvars* $x_1, x_2, \ldots, x_k$ are transformed by a Box–Cox transform with the parameter λ. Again, the $z_1, z_2, \ldots, z_l$ are independent variables that are not transformed.

Example

Here is an example with the `rhsonly` model. In this example, `price` and `foreign` are not included in the list of covariates. (You are invited to use the auto data and check that they fare no better here than above.)

```
. boxcox2 mpg weight, model(rhs) lrtest  nolog nologlr
Estimating Full Model
Estimating comparison models for LR tests
Comparison model for LR test on weight is a linear regression
Lambda is not identified in the restricted  model

                                                  Number of obs   =         74
                                                  LR chi2(2)      =      82.90
Log likelihood = -192.94368                       Prob > chi2     =      0.000

------------------------------------------------------------------------------
      mpg|      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 /lambda |  -.4460916   .6551107     -0.681   0.496      -1.730085    .8379017
------------------------------------------------------------------------------

Estimates of Scale Variant Parameters
---------------------------------------------------------
         |      Coef.   chi2(df)   P>chi2(df)  df of chi2
---------+-----------------------------------------------
Notrans  |
   _cons |   1359.092
---------+-----------------------------------------------
Trans    |
  weight |  -614.3874    82.901     0.000         2
---------+-----------------------------------------------
  /sigma |   3.281854
---------------------------------------------------------


---------------------------------------------------------
   Test         Restricted     LR statistic      P-Value
   H0:        log likelihood      X~chi2       Pr > chi2
---------------------------------------------------------
lambda = -1      -193.2893         0.69          0.406
lambda =  0     -193.17892         0.47          0.493
lambda =  1     -195.38869         4.89          0.027
---------------------------------------------------------
```

The interpretation of the output is similar to all the cases above, except for one caveat. As requested, a likelihood-ratio test was performed on the lone independent variable. However, when it is dropped to form the constrained model, the comparison model is not a right-hand-side only Box–Cox model, but rather a simple linear regression on a constant model. When weight is dropped, there are no longer any transformed variables. Hence, λ is not identified and it must also be dropped. This process leaves a linear regression on a constant as the “comparison model”. It also implies that the test statistic has 2 degrees of freedom instead of 1. At the top of the output, there is a more concise warning which informs the user of this point.

Technical Note

A similar identification issue can also arise in the `lambda` and `theta` models when only one independent variable is specified. In these cases, warnings also appear on the output to remind the user.

Saved Results

`boxcox2` saves in `e()`:

Scalars			
`e(N)`	number of observations	`e(chi2)`	LR statistic of full vs comparison
`e(ll)`	log likelihood	`e(ll0)`	log likelihood of the restricted model
`e(df_m)`	full model degrees of freedom	`e(df_r)`	restricted model degrees of freedom
`e(ll_t1)`	log likelihood of model when λ=θ=1	`e(chi2_t1)`	LR statistic of λ=θ=1 vs full model
`e(p_t1)`	p-value of λ=θ=1 vs full model	`e(ll_tm1)`	log likelihood of model when λ=θ=-1
`e(chi2_tm1)`	LR statistic of λ=θ=-1 vs full model	`e(p_tm1)`	p-value of λ=θ=-1 vs full model
`e(ll_t0)`	log likelihood of model when λ=θ=0	`e(chi2_t0)`	LR statistic of λ=θ=0 vs full model
`e(p_t0)`	p-value of λ=θ=0 vs full model	`e(ic)`	number of iterations
`e(rc)`	return code		
Macros			
`e(cmd)`	`boxcox2`	`e(ntrans)`	`yes` if there were nontransformed *indepvars*
`e(model)`	model estimated	`e(predict)`	program used to implement `predict`
`e(chi2type)`	`LR`	`e(lrtest)`	`lrtest` if requested
`e(depvar)`	name of dependent variable	`e(wtype)`	weight type
`e(wexp)`	weight expression		
Matrices			
`e(b)`	coefficient vector	`e(V)`	variance–covariance matrix of the estimators (see note below)
`e(df)`	degrees of freedom of LR tests on *indepvars*, if requested	`e(pm)`	p-values for LR tests on *indepvars*, if requested
`e(chi2m)`	LR statistics for tests on *indepvars*, if requested		
Functions			
`e(sample)`	marks estimation sample		

Note that `e(V)` contains all zeros except for the element(s) that correspond to the parameter(s) of the Box–Cox transform.

Methods and Formulas

`boxcox2` is implemented as an ado-file.

In the internal computations

$$y^{(\lambda)} = \begin{cases} \dfrac{y^\lambda - 1}{\lambda} & \text{if } |\lambda| > 10^{-10} \\ \ln(y) & \text{otherwise} \end{cases}$$

The unconcentrated log likelihood for the `theta` model is

$$\ln L = \left(\frac{-N}{2}\right)\big(\ln(2\pi) + \ln(\sigma^2)\big) + (\theta - 1)\sum_i^N \ln(y_i) - \left(\frac{1}{2\sigma^2}\right)\text{SSR}$$

where

$$\text{SSR} = \sum_i^N (y_i^{(\theta)} - \beta_0 + \beta_1 x_{i1}^{(\lambda)} + \beta_2 x_{i2}^{(\lambda)} + \ldots + \beta_k x_{ik}^{(\lambda)} + \gamma_1 z_{i1} + \gamma_2 z_{i2} + \ldots + \gamma_l z_{il})^2$$

Writing the SSR in matrix form,

$$\text{SSR} = (\mathbf{Y}^{(\theta)} - \mathbf{X}^{(\lambda)}\mathbf{b}' - \mathbf{Z}\mathbf{g}')'(\mathbf{Y}^{(\theta)} - \mathbf{X}^{(\lambda)}\mathbf{b}' - \mathbf{Z}\mathbf{g}')$$

where $\mathbf{Y}^{(\theta)}$ is an $N \times 1$ vector of elementwise transformed data, $\mathbf{X}^{(\lambda)}$ is an $N \times k$ matrix of elementwise transformed data, $\mathbf{Z}$ is an $N \times l$ matrix of untransformed data, $\mathbf{b}$ is a $1 \times k$ vector of coefficients, and $\mathbf{g}$ is a $1 \times l$ vector of coefficients. Letting

$$\mathbf{W}_\lambda = \left(\mathbf{X}^{(\lambda)}\ \mathbf{Z}\right)$$

the horizontal concatenation of $\mathbf{X}^{(\lambda)}$ and $\mathbf{Z}$, and

$$\mathbf{d}' = \begin{pmatrix} \mathbf{b}' \\ \mathbf{g}' \end{pmatrix}$$

the vertical concatenation of the coefficients, yields

$$\text{SSR} = (\mathbf{Y}^{(\theta)} - \mathbf{W}_\lambda \mathbf{d}')'(\mathbf{Y}^{(\theta)} - \mathbf{W}_\lambda \mathbf{d}')$$

For given values of λ and θ, the solutions for $\mathbf{d}'$ and σ^2 are

$$\widehat{\mathbf{d}}' = (\mathbf{W}_\lambda' \mathbf{W}_\lambda)^{-1} \mathbf{W}_\lambda' \mathbf{Y}^{(\theta)}$$

and

$$\widehat{\sigma}^2 = \frac{1}{N}\left(\mathbf{Y}^\theta - \mathbf{W}_\lambda \widehat{\mathbf{d}}'\right)' \left(\mathbf{Y}^\theta - \mathbf{W}_\lambda \widehat{\mathbf{d}}'\right)$$

Substituting these solutions into the log-likelihood function yields the concentrated log-likelihood function,

$$\ln L_c = \left(-\frac{N}{2}\right)\left(\ln(2\pi) + 1 + \ln(\widehat{\sigma}^2)\right) + (\theta - 1)\sum_i^N \ln(y_i)$$

The unconcentrated log likelihood for the `lambda` model is

$$\ln L = \left(\frac{-N}{2}\right)\left(\ln(2\pi) + \ln(\sigma^2)\right) + (\lambda - 1)\sum_i^N \ln(y_i) - \left(\frac{1}{2\sigma^2}\right)\text{SSR}$$

where

$$\text{SSR} = \sum_i^N (y_i^{(\lambda)} - \beta_0 + \beta_1 x_{i1}^{(\lambda)} + \beta_2 x_{i2}^{(\lambda)} + \ldots + \beta_k x_{ik}^{(\lambda)} + \gamma_1 z_{i1} + \gamma_2 z_{i2} + \ldots + \gamma_l z_{il})^2$$

Writing the SSR in matrix form,

$$\text{SSR} = (\mathbf{Y}^{(\lambda)} - \mathbf{X}^{(\lambda)}\mathbf{b}' - \mathbf{Z}\mathbf{g}')'(\mathbf{Y}^{(\lambda)} - \mathbf{X}^{(\lambda)}\mathbf{b}' - \mathbf{Z}\mathbf{g}')$$

where $\mathbf{Y}^{(\lambda)}$ is an $N \times 1$ vector of transformed data and everything else has already been defined. Using $\mathbf{W}_\lambda$ and $\mathbf{d}$ as defined above, the sum of squared residuals is now

$$\text{SSR} = (\mathbf{Y}^{(\lambda)} - \mathbf{W}_\lambda \mathbf{d}')'(\mathbf{Y}^{(\lambda)} - \mathbf{W}_\lambda \mathbf{d}')$$

For a given value of λ, the solutions for $\mathbf{d}'$ and σ^2 are

$$\widehat{\mathbf{d}}' = (\mathbf{W}_\lambda' \mathbf{W}_\lambda)^{-1} \mathbf{W}_\lambda' \mathbf{Y}^{(\lambda)}$$

and

$$\widehat{\sigma}^2 = \frac{1}{N}\left(\mathbf{Y}^\lambda - \mathbf{W}_\lambda \widehat{\mathbf{d}}'\right)' \left(\mathbf{Y}^\lambda - \mathbf{W}_\lambda \widehat{\mathbf{d}}'\right)$$

Substituting these solutions into the log-likelihood function yields the concentrated log-likelihood function:

$$\ln L_c = \left(-\frac{N}{2}\right)\left(\ln(2\pi) + 1 + \ln(\widehat{\sigma}^2)\right) + (\lambda - 1)\sum_i^N \ln(y_i)$$

The unconcentrated log likelihood for the `lhsonly` model is

$$\ln L = \left(\frac{-N}{2}\right)\left(\ln(2\pi) + \ln(\sigma^2)\right) + (\theta - 1)\sum_i^N \ln(y_i) - \left(\frac{1}{2\sigma^2}\right)\text{SSR}$$

where

$$\text{SSR} = \sum_i^N (y_i^{(\theta)} - \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik})^2$$

Writing the SSR in matrix form,

$$\text{SSR} = (\mathbf{Y}^{(\theta)} - \mathbf{X}\mathbf{b}')'(\mathbf{Y}^{(\theta)} - \mathbf{X}\mathbf{b}')$$

where $\mathbf{Y}^{(\theta)}$ is an $N \times 1$ vector of transformed data, $\mathbf{X}$ is an $N \times k$ matrix of untransformed data, $\mathbf{b}$ is a $1 \times k$ vector of coefficients. For a given value of θ, the solutions for $\mathbf{b}'$ and σ^2 are

$$\widehat{\mathbf{b}}' = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}^{(\theta)}$$

and

$$\widehat{\sigma}^2 = \frac{1}{N}\Big(\mathbf{Y}^{(\theta)} - \mathbf{X}\widehat{\mathbf{b}}'\Big)'\Big(\mathbf{Y}^{(\theta)} - \mathbf{X}\widehat{\mathbf{b}}'\Big)$$

Substituting these solutions into the log-likelihood function yields the concentrated log-likelihood function:

$$\ln L_c = \left(-\frac{N}{2}\right)\big(\ln(2\pi) + 1 + \ln(\widehat{\sigma}^2)\big) + (\theta - 1)\sum_i^N \ln(y_i)$$

The unconcentrated log likelihood for the **rhsonly** model is

$$\ln L = \left(\frac{-N}{2}\right)\big(\ln(2\pi) + \ln(\sigma^2)\big) - \left(\frac{1}{2\sigma^2}\right)\text{SSR}$$

where

$$\text{SSR} = \sum_i^N (y_i - \beta_0 + \beta_1 x_{i1}^{(\lambda)} + \beta_2 x_{i2}^{(\lambda)} + \ldots + \beta_k x_{ik}^{(\lambda)} + \gamma_1 z_{i1} + \gamma_2 z_{i2} + \ldots + \gamma_l z_{il})^2$$

Writing the SSR in matrix form,

$$\text{SSR} = (\mathbf{Y} - \mathbf{X}^{(\lambda)}\mathbf{b}' - \mathbf{Z}\mathbf{g}')'(\mathbf{Y} - \mathbf{X}^{(\lambda)}\mathbf{b}' - \mathbf{Z}\mathbf{g}')$$

where $\mathbf{Y}$ is an $N \times 1$ vector of untransformed data, $\mathbf{X}^{(\lambda)}$ is an $N \times k$ matrix of transformed data, $\mathbf{Z}$ is an $N \times l$ matrix of untransformed data, $\mathbf{b}$ is a $1 \times k$ vector of coefficients, and $\mathbf{g}$ is a $1 \times l$ vector of coefficients. Letting

$$\mathbf{W}_\lambda = \Big(\mathbf{X}^{(\lambda)}\ \mathbf{Z}\Big)$$

the horizontal concatenation of $\mathbf{X}^{(\lambda)}$ and $\mathbf{Z}$, and

$$\mathbf{d}' = \begin{pmatrix} \mathbf{b}' \\ \mathbf{g}' \end{pmatrix}$$

yields

$$\text{SSR} = (\mathbf{Y} - \mathbf{W}_\lambda\mathbf{d}')'(\mathbf{Y} - \mathbf{W}_\lambda\mathbf{d}')$$

For a given value of λ, the solutions for $\mathbf{d}'$ and σ^2 are

$$\widehat{\mathbf{d}}' = (\mathbf{W}_\lambda'\mathbf{W}_\lambda)^{-1}\mathbf{W}_\lambda'\mathbf{Y}$$

and

$$\widehat{\sigma}^2 = \frac{1}{N}\Big(\mathbf{Y} - \mathbf{W}_\lambda\widehat{\mathbf{d}}'\Big)'\Big(\mathbf{Y} - \mathbf{W}_\lambda\widehat{\mathbf{d}}'\Big)$$

Substituting these solutions into the log-likelihood function yields the concentrated log-likelihood function:

$$\ln L_c = \left(-\frac{N}{2}\right)\big(\ln(2\pi) + 1 + \ln(\widehat{\sigma}^2)\big)$$

References

Atkinson, A. C. 1985. *Plots, Transformations, and Regression.* Oxford: Clarendon Press.

Box, G. E. P. and D. R. Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society*, Series B 26: 211–243.

Cook, R. D. and S. Weisberg. 1982. *Residuals and Influence in Regression.* New York: Chapman & Hall.

Davidson, R. and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics.* Oxford: Oxford University Press.

Drukker, D. M. 2000. sg131: On the manipulability of Wald statistics in Box–Cox regression models. *Stata Technical Bulletin* 54: 36–42.

Lafontaine, F. and K. J. White. 1986. Obtaining any Wald statistic you want. *Economics Letters* 21: 35–40.

Phillips, P. C. B. and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083.

Schlesselman, J. 1971. Power families: A note on the Box and Cox transformation. *Journal of the Royal Statistical Society*, Series B 33: 307–311.

Spitzer, J. J. 1984. Variance estimates in models with the Box–Cox transformation: Implications for estimation and hypothesis testing. *The Review of Economics and Statistics* 66: 645–652.

sg131 | On the manipulability of Wald tests in Box–Cox regression models

David M. Drukker, Stata Corporation, ddrukker@stata.com

Abstract: This article illustrates the fact that the value of the Wald Test of the significance of a coefficient on an independent variable in a Box–Cox regression model is not invariant to changes in the scale of any of the transformed variables. The article shows that this result is a special case of the manipulability of the Wald statistic in nonlinear models, a topic that has been treated in the literature by Lafontaine and White (1986) and Phillips and Park (1988). The article considers several candidate methods for dealing with the problem and concludes that using likelihood-ratio tests is the best alternative.

Keywords: Box–Cox regression, nonlinear regression, Wald tests, nonlinear models, scale invariance, scale invariant test statistics, scale variant test statistics.

Introduction

This article illustrates the fact that the value of the Wald test of the significance of a coefficient on an independent variable in a Box–Cox regression model is not invariant to changes in the scale of any of the transformed variables. Spitzer (1984) first discovered this fact in a study of Box–Cox regression models. Later, in independent work, Gregory and Veal (1985) and Lafontaine and White (1986) showed that certain classes of nonlinear transformations of a Wald test produce significantly different values and conclusions in a finite sample. Phillips and Park (1988) used Edgeworth expansions to generalize this conclusion to a very general class of nonlinear transformations.

The manipulability of the Wald test of the significance of an independent variable in a Box–Cox regression is a special case of the more general phenomenon that Wald tests are not invariant to nonlinear transforms. This article uses two different Stata commands that perform Box–Cox regression and examples from the auto dataset to illustrate this fact. The first command `box` has a syntax similar to the new `boxcox2` which is documented in the online help and in Drukker (2000). The command `box` is neither an official Stata command nor a command released in another format, such as the STB. `box` was written by the author for the purpose of writing this article. Researchers interested in estimating Box–Cox models are encouraged to use `boxcox2`.

In Box–Cox regressions, some or all of the variables are transformed by the Box–Cox transform which is

$$w^{(\theta)} = \frac{w^\theta - 1}{\theta}$$

Box–Cox regressions can take one of four different forms, depending on which variables are transformed. The examples in this article all use the Box–Cox model in which only the dependent variable is transformed. Since the dependent variable appears on the left-hand-side, this model is called the “left-hand-side only model”. Specifically, the model estimated in these examples is

$$y_j^{(\theta)} = \beta_0 + \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + \epsilon_j$$

where $\epsilon \sim N(0, \sigma^2)$. Although the discussion is focused on this one model, the issues generalize to all four of the models estimated by `boxcox2`. For an introduction to Box–Cox models, see Davidson and MacKinnon (1993). For a discussion of their implementation in Stata, see Drukker (2000).

Obtaining any Wald statistic

The variance of the Wald test to nonlinear transforms is the root of the problem. Hence, a good place to begin our investigation is with a simple example of this phenomenon. Consider the following example, which is similar to one given in Lafontaine and White (1986). Running a linear regression of `mpg` on `weight` and `price`, using the auto dataset produces

```
. regress mpg price weight

  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  2,    71) =   66.85
   Model |  1595.93249     2  797.966246               Prob > F      =  0.0000
Residual |  847.526967    71  11.9369995               R-squared     =  0.6531
---------+------------------------------               Adj R-squared =  0.6434
   Total |  2443.45946    73  33.4720474               Root MSE      =   3.455

------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   price |  -.0000935   .0001627     -0.575   0.567       -.000418    .0002309
  weight |  -.0058175   .0006175     -9.421   0.000      -.0070489   -.0045862
   _cons |   39.43966   1.621563     24.322   0.000       36.20635    42.67296
------------------------------------------------------------------------------
```

Now, let's perform several equivalent tests of whether or not `weight` is significant in the regression.

```
. test weight

 ( 1)  weight = 0.0

       F(  1,    71) =   88.75
            Prob > F =    0.0000

. testnl _b[weight]=0

  (1)  _b[weight]=0

              F(1, 71) =       88.75
              Prob > F =        0.0000

. testnl _b[weight]^2=0

  (1)  _b[weight]^2=0

              F(1, 71) =       22.19
              Prob > F =        0.0000
```

Note that when the same Wald test is performed with `test` and `testnl`, the same value of the Wald statistic is obtained. However, when the logically equivalent, but algebraically distinct test of whether or not $\beta^2_{\text{weight}} = 0$ is performed, the value of the test is approximately a fourth of its original value. As shown in Lafontaine and White (1986, 35), "because the nonlinear form of the Wald statistic stems from a Taylor series approximation, different values and possibly different diagnostics are obtained from the above seeming equivalent tests".

Phillips and Park (1988) used Edgeworth expansions to generalize the previous research. They were able to demonstrate that

> Under general conditions Wald statistics which are based upon different but algebraically equivalent forms all have the same asymptotic distribution under the null hypothesis that the restriction holds. However, numerical outcomes of the tests and their finite sample distributions can be substantially different for different forms of the same restrictions.

Scale-variant Wald statistics in Box–Cox regressions

Spitzer (1984) first discovered that Wald tests of the significance of an independent variable were not invariant to changes in the scale of any transformed variable in a Box–Cox regression. He derives the variance estimator for the coefficients on the right-hand-side independent variables in a Box–Cox regression model in which only the dependent variable is transformed. Spitzer shows analytically that this variance estimator depends on the scale of the dependent variable. He goes on to give numerical examples that show how the value of the Wald test of the significance of the independent variables can be manipulated by changing the scale of the dependent variable.

Consider a similar example using the auto dataset. This example will illustrate the lack of scale-invariance and it will also show how this is a special case of the manipulability of Wald tests via nonlinear restrictions. Begin with a Box–Cox regression in which only the dependent variable is transformed.

```
. box mpg weight price , nolog
 Estimating comparison model

 Maximizing concentrated likelihood

 Maximizing the unconcentrated likelihood

                                                 Number of obs   =        74
                                                 LR chi2(3)      =    100.40
Log likelihood = -178.06886                      Prob > chi2     =    0.0000
```

```
------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
ntrans   |
  weight |  -.0000263   .0000232     -1.132   0.258      -.0000718    .0000192
   price |  -1.34e-06   1.18e-06     -1.135   0.256      -3.66e-06    9.75e-07
   _cons |   1.272474   .4123654      3.086   0.002       .4642524    2.080695
---------+--------------------------------------------------------------------
Ancillary|
   theta |  -.7568509   .2901099     -2.609   0.009      -1.325456   -.1882459
   sigma |   .0132169   .0116479      1.135   0.256      -.0096125    .0360462
---------+--------------------------------------------------------------------

. test _b[ntrans: weight]=0
 ( 1)  [ntrans]weight = 0.0
           chi2(  1) =    1.28
         Prob > chi2 =    0.2575
```

A Wald test of the significance of `weight` is also performed. Note that this test produces a p-value identical to that given in the output table. This value indicates the unlikely diagnostic that `weight` does not have a statistically significant effect on `mpg`, at any of the standard test sizes.

Now, rescale `mpg` to `mpg`/10 and rerun the same procedure.

```
. gen mpg2=mpg/10
. box mpg2 weight price, nolog
 Estimating comparison model

 Maximizing concentrated likelihood

 Maximizing the unconcentrated likelihood
                                                  Number of obs   =         74
                                                  LR chi2(3)      =     100.40
Log likelihood = -7.6775588                       Prob > chi2     =     0.0000
------------------------------------------------------------------------------
    mpg2 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
ntrans   |
  weight |  -.0001502   .0000344     -4.365   0.000      -.0002177   -.0000828
   price |  -7.66e-06   3.61e-06     -2.123   0.034      -.0000147   -5.88e-07
   _cons |   1.042532   .1589603      6.558   0.000       .7309758    1.354089
---------+--------------------------------------------------------------------
Ancillary|
   theta |  -.7568511   .2902993     -2.607   0.009      -1.325827   -.1878749
   sigma |   .0755057   .0169982      4.442   0.000       .0421898    .1088216
---------+--------------------------------------------------------------------
```

Note that the Wald test in the table now indicates that `weight` is indeed a statistically significant predictor of `mpg`/10 at all of the conventional sizes.

Since there is a constant in the model, estimates of the Box–Cox transform parameter and its variance are invariant to any rescaling of the variables. Schlesselman (1971) demonstrated this point analytically. Using this fact, a few lines of algebra show that a rescaling of the dependent variable in a left-hand-side only Box–Cox regression model results in a simple nonlinear transformation of the parameters. Specifically, write the original model as

$$\frac{\mathbf{y}^{\theta}-\mathbf{1}}{\theta}=\mathbf{Xb}+\mathbf{e}$$

where $\mathbf{y}$ is an $N\times 1$ vector of observations on the dependent variable, θ is the parameter of the Box–Cox transform, $\mathbf{X}$ is the $N\times k$ matrix of observations on the independent variables, $\mathbf{b}$ is the $k\times 1$ vector of coefficients on the independent variables and $\mathbf{e}$ is the $N\times 1$ vector of normally distributed errors. $\mathbf{X}$ contains a vector of ones in its first column.

Now, rescale the regression.

$$\frac{\left(\frac{\mathbf{y}}{\mathbf{c}}\right)^{\theta}-\mathbf{1}}{\theta}=\mathbf{Xb_s}+\mathbf{e_s}$$

where $\mathbf{b_s}$ is the $k\times 1$ vector of coefficients in the scaled model. Since estimates of θ are invariant to the transform, it needs no subscript. Solving this equation for

$$\frac{\mathbf{y}^{\theta}-\mathbf{1}}{\theta}$$

implies that

$$\frac{\mathbf{y}^{\theta}-\mathbf{1}}{\theta}=\frac{\mathbf{c}^{\theta}-\mathbf{1}}{\theta}+\mathbf{c}^{\theta}\mathbf{X}\mathbf{b_s}+\mathbf{c}^{\theta}\mathbf{e_s}$$

the implication being that, except for the constant,

$$\mathbf{b}=\mathbf{c}^{\theta}\mathbf{b_s}$$

It is easy to verify that this formula works for the example at hand.

```
. scalar b1=_b[ntrans: weight]*10^(_b[theta: _cons])

. di b1
-.0000263
```

This situation is now similar to the example of β_{weight} and β^2_{weight} discussed above. Logically, except for the constant, $\mathbf{b}=\mathbf{0}$ if and only if $\mathbf{b_s}=\mathbf{0}$. However, the Wald test on `weight` in the scaled regression indicates that `weight` is significant, while `weight` is not significant in the unscaled regression. Now, note that the test of significance of `weight` in the unscaled regression is obtainable as a nonlinear test on `weight` in the scaled regression.

```
. testnl _b[ntrans: weight]*10^(_b[theta: _cons])=0
  (1)  _b[ntrans: weight]*10^(_b[theta: _cons])=0
                chi2(1) =        1.28
            Prob > chi2 =        0.2579
```

Hence, the lack of invariance to scale in the Box–Cox regression model is just an example of the more general lack of invariance of Wald tests to nonlinear transformations.

Scaling to elasticities

If all data were scaled in natural units, then there would be no issue here and researchers would always analyze data in their natural units. Of course, most data does not have any natural units and one scale is as arbitrary as another. Spitzer (1984) argues that the solution to the lack of invariance in the Box–Cox regression model is to always analyze data normalized by its geometric mean. The following example illustrates that if all data are scaled by their geometric means, then the coefficients on the independent variables from a left-hand-side only Box–Cox regression are like elasticities. Let $G(x)$ be the geometric mean of x. The elasticity of the transformed dependent variable with respect to an independent variable evaluated at their geometric means would be

$$\frac{\frac{\partial y^{(\theta)}}{(G(y))^{(\theta)}}}{\frac{\partial x}{G(x)}}=\frac{\partial y^{(\theta)}}{\partial x}\frac{G(x)}{(G(y))^{(\theta)}}$$

Even with the data scaled as Spitzer suggests, this computation will not produce the coefficients on the independent variables in a left-hand-side only Box–Cox regression. The formula that will reproduce these coefficients is

$$\frac{\frac{\partial y^{(\theta)}}{G(y)^{\theta}}}{\frac{\partial x}{G(x)}}=\frac{\partial y^{(\theta)}}{\partial x}\frac{G(x)}{G(y)^{\theta}}$$

The denominators are the same in the two computations. The numerator in the first computation is the change in the transformed dependent variable as a fraction of the Box–Cox transformation of the geometric mean of the dependent variable. In the second case the numerator is the change in the transformed dependent variable as a fraction of the geometric mean of the dependent variable raised to power θ. The former equation is an elasticity. The latter equation is “like” an elasticity and is what is produced by Spitzer’s method.

Now let’s use Stata to calculate these formulas and verify that when all variables are scaled by their geometric means a left-hand-side only Box–Cox regression produces coefficient estimates that are identical to those produced by the “like elasticity” formula. We begin by computing the geometric means of the variables of interest and saving them in scalars. Several functions of these means are also calculated. In particular, `elw2` is the elasticity of the transformed dependent variable with respect to `weight` evaluated at the mean of both variables. `elw` is the “like elasticity” computed by the formula given above. Note that they are not equal.

```
. box mpg weight price , nolog
  (output omitted)

. means mpg
Variable |    Type        Obs        Mean       [95% Conf. Interval]
---------+----------------------------------------------------------
     mpg | Arithmetic      74     21.2973        19.9569   22.63769
         |  Geometric      74    20.58444       19.38034   21.86335
         |   Harmonic      74    19.92318       18.81185   21.17405
---------+----------------------------------------------------------
```

```
. scalar mdot=r(mean_g)  /* mdot is geometric mean of depvar */
. gen mpgs=mpg/r(mean_g) /* mpgs is scaled depvar */
. scalar mdot2=mdot^(_b[theta: _cons]) /* this is denominator for
                                          like elasticity */
. scalar mdot2b=(mdot^(_b[theta: _cons])-1)/(_b[theta: _cons])
       /* This is denominator for elasticity */
. means weight
Variable |    Type        Obs        Mean       [95% Conf. Interval]
---------+----------------------------------------------------------
  weight | Arithmetic      74    3019.459        2839.398   3199.521
         |  Geometric      74    2918.284         2743.65   3104.034
         |   Harmonic      74    2816.578        2649.055   3006.719
---------+----------------------------------------------------------
. scalar wdot=r(mean_g)  /* geometric mean of weight */
. gen weights=weight/r(mean_g) /* scaled weight */
. means price
Variable |    Type        Obs        Mean       [95% Conf. Interval]
---------+----------------------------------------------------------
   price | Arithmetic      74    6165.257        5481.914     6848.6
         |  Geometric      74    5656.907        5165.664   6194.865
         |   Harmonic      74    5296.672        4928.901    5723.75
---------+----------------------------------------------------------
. scalar pdot=r(mean_g) /* geometric mean of price */
. gen prices=price/r(mean_g) /* scaled price */
. scalar elw=_b[ntrans: weight]*wdot/mdot2  /* apply like elasticity formula*/
. di elw
-.75713121
. scalar elw2=_b[ntrans: weight]*wdot/mdot2b /* apply elasticity formula */
. di elw2
-.06463091
. scalar elp=_b[ntrans: price]*pdot/mdot2  /* apply like elasticity formula*/
. di elp
-.07486248
```

Now we need to verify that when all the variables are scaled by their geometric means, the coefficients on the independent variables are identical to those computed using the "like elasticity" formula. Below is the output from a Box–Cox regression of $\texttt{mpgs} = \texttt{mpg}/G(\texttt{mpg})$ on $\texttt{weights} = \texttt{weight}/G(\texttt{weight})$ and $\texttt{prices} = \texttt{price}/G(\texttt{price})$.

```
. box mpgs prices weights, nolog
 Estimating comparison model
 Maximizing concentrated likelihood
 Maximizing the unconcentrated likelihood
                                                  Number of obs   =         74
                                                  LR chi2(3)      =     100.40
Log likelihood =  45.746772                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
    mpgs |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
ntrans   |
  prices |  -.0748625   .0360538     -2.076   0.038      -.1455267   -.0041983
 weights |  -.7571312   .0680234    -11.130   0.000      -.8904547   -.6238077
   _cons |   .8398833   .0612024     13.723   0.000       .7199287    .9598378
---------+--------------------------------------------------------------------
Ancillary|
   theta |  -.7568508   .2902995     -2.607   0.009      -1.325827   -.1878742
   sigma |   .1304014   .0107189     12.166   0.000       .1093927    .1514101
---------+--------------------------------------------------------------------
```

Note that the coefficients on the independent variables are identical to those calculated by `elw` and `elp` above. These equalities illustrate the claim that when all the variables are scaled by their geometric means, the coefficients on the independent variables from a left-hand-side only Box–Cox regression are "like elasticities".

An LR test solution

Software developers must choose how to handle the manipulability issue. There are several options. One would be to automatically scale all the variables by their geometric mean and produce Wald tests of significance in a standard output table.

This alternative would force users interested in the coefficients expressed in another scale to transform them “by hand”. The fact that the coefficient estimates are not true elasticities in all the Box–Cox models further reduces the appeal of this solution. Since likelihood-ratio tests are invariant to any rescaling of the variables there is another option. Allow users to run the regression in any scale desired but only perform likelihood-ratio tests as opposed to Wald tests. I chose the latter in constructing the new `boxcox2`. Users can estimate their models in the scale most convenient for them and easily obtain scale invariant test statistics.

The following example from the auto data using the new `boxcox2` illustrates that LR statistics are invariant to the scale of the data.

```
. boxcox2 mpg weight price , nolog nologlr lrtest
Estimating comparison model
Estimating full model
Estimating comparison models for LR tests
                                                  Number of obs   =         74
                                                  LR chi2(2)      =     100.40
Log likelihood = -178.06886                       Prob > chi2     =      0.000
------------------------------------------------------------------------------
      mpg|      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  /theta |  -.7568509   .2902995     -2.607   0.009      -1.325828   -.1878744
------------------------------------------------------------------------------
Estimates of scale-variant parameters
----------------------------------------------------------
         |      Coef.   chi2(df)   P>chi2(df)  df of chi2
---------+------------------------------------------------
Notrans  |
  weight |  -.0000263    72.791     0.000         1
   price |  -1.34e-06     4.202     0.040         1
   _cons |   1.272474
---------+------------------------------------------------
  /sigma |   .0218107
----------------------------------------------------------
----------------------------------------------------------
   Test         Restricted     LR statistic      P-Value
    H0:       log likelihood      X~chi2        Pr > chi2
----------------------------------------------------------
theta = -1      -178.41823         0.70           0.403
theta =  0      -181.43399         6.73           0.009
theta =  1      -195.21698        34.30           0.000
----------------------------------------------------------
. boxcox2 mpg2 weight price , nolog nologlr lrtest
Estimating comparison model
Estimating full model
Estimating comparison models for LR tests
                                                  Number of obs   =         74
                                                  LR chi2(2)      =     100.40
Log likelihood = -7.6775588                       Prob > chi2     =      0.000
------------------------------------------------------------------------------
     mpg2|      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  /theta |  -.7568511   .2902995     -2.607   0.009      -1.325828   -.1878745
------------------------------------------------------------------------------
Estimates of scale-variant parameters
----------------------------------------------------------
         |      Coef.   chi2(df)   P>chi2(df)  df of chi2
---------+------------------------------------------------
Notrans  |
  weight |  -.0001502    72.791     0.000         1
   price |  -7.66e-06     4.202     0.040         1
   _cons |   1.042532
---------+------------------------------------------------
  /sigma |   .0869115
----------------------------------------------------------
----------------------------------------------------------
   Test         Restricted     LR statistic      P-Value
    H0:       log likelihood      X~chi2        Pr > chi2
----------------------------------------------------------
theta = -1      -8.0269351         0.70           0.403
theta =  0      -11.042697         6.73           0.009
theta =  1      -24.825684        34.30           0.000
----------------------------------------------------------
```

There are two important points to note about this output. First, the coefficient estimates exactly match those from the previous procedures. Second, the LR tests are invariant to changes in the scale of the transformed dependent variable.

Conclusion

This article has illustrated two important facts. First, the value of Wald tests of significance are not invariant to nonlinear transformations. Second, an important special case of this result is that Wald tests on the significance of an independent variable are not invariant to changes in the scale of any transformed variable in a Box–Cox regression model. This article has also illustrated that LR tests are invariant to the scale of transformed variables in a Box–Cox regression model. It has also argued that allowing researchers to choose their own scale for estimation with LR based inference is superior to boxing them into a specific scale that has some desirable properties.

References

Davidson, R. and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics.* Oxford: Oxford University Press.

Drukker, D. M. 2000. sg130: Box–Cox regression models. *Stata Technical Bulletin* 54: 27–36.

Gregory, A. W. and M. R. Veal. 1985. Formulating Wald tests on nonlinear restrictions. *Econometrica* 53: 1465–1468.

Lafontaine, F. and K. J. White. 1986. Obtaining any Wald statistic you want. *Economics Letters* 21: 35–40.

Phillips, P. C. B. and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083.

Schlesselman, J. 1971. Power Families: A note on the Box and Cox transformation. *Journal of the Royal Statistical Society*, Series B 33: 307–311.

Spitzer, J. J. 1984. Variance estimates in models with the Box-Cox transformation: Implications for Estimation and Hypothesis Testing. *The Review of Economics and Statistics* 66: 645–652.

sg132 Analysis of variance from summary statistics

John R. Gleason, Syracuse University, loesljrg@accucom.net

Abstract: `aovsum` is a command for performing analysis of variance when the data are available only in summary form; namely, as group sizes, means, and standard deviations. This is accomplished by synthesizing a dataset to match those summary values. Other linear model style analyses can then be performed using the synthetic data; for example, multiple comparisons and trend analysis.

Keywords: ANOVA, univariate summary, linear model.

Introduction

It is common practice in many scientific journals to summarize data with a table showing, for various groups, the sample size (n), mean ($\bar{y}$), and standard deviation (s) for some collection of variables; possibly, the standard error $s/\sqrt{n}$ appears in place of s. A reader might on occasion wish that the authors had also provided an analysis of variance (ANOVA) for some variable(s). That desire might spring from curiosity about the ANOVA F-statistic and its p-value, or from a need for the ANOVA's pooled estimate of error variance. Or, perhaps the groups in the table correspond to the cells of an unbalanced factorial design. To judge the size and significance of the various main and interaction effects from an inspection of a table of means and standard deviations can be a nontrivial task.

`aovsum` is a command that can compute the ANOVA summary table corresponding to a series of sample sizes, means, and standard deviations (or, standard errors). `aovsum` capitalizes upon some simple facts:

1. Any one-way ANOVA for independent groups can be computed from the values of n, $\bar{y}$, and s in the various groups.
2. Any multifactor design having only fixed factors can, without loss, be construed as a one-way design with K groups, where K is the total number of cells in the design. This tactic is the basis of the *cell means approach* to analyzing data from factorial designs.
3. Finally, let $n > 0$, $\bar{y}$, and $s > 0$ be given. Then a dataset consisting of $n-1$ copies of the value $\bar{y} + s/\sqrt{n}$ and one copy of $\bar{y} - (n-1)s/\sqrt{n}$ will have mean equal to $\bar{y}$ and standard deviation equal to s.

`aovsum` uses (3) to synthesize a dataset with the correct mean, standard deviation, and sample size in each of the various groups. `aovsum` then invokes `oneway` to present the one-way ANOVA for that synthetic dataset, in accord with (1). But `aovsum` can optionally save the synthetic data, and the user can then create variables that encode the factors of the underlying experimental design; the inverse of the tactic described in (2). The ANOVA command can then be used to examine the various main and interaction effects, as desired. The process is entirely accurate, subject only to limits imposed by the precision to which the means and standard deviations (or standard errors) have been reported. The only requirement is that the appropriate ANOVA model specify a single random term, as in fixed-effects, between-subject designs. For the sake of illustration, the examples below are situations where the raw data are in fact available; in practice, of course, `aovsum` is useful only when the raw data are not available.

Syntax

There are two forms of syntax:

`aovsum , n(`*nlist*`) mean(`*mlist*`) { sd(`*SDlist*`) | se(`*SElist*`) } [names(`*yname* [*grpname* [*freqname*]]`)`

`keep` *onewayopt*]

`aovsum ?`

Description

In the first form, *nlist* is a list of sample sizes and *mlist* is a list of the associated sample means. *SDlist* and *SElist* are lists of sample standard deviations and standard errors, respectively; exactly one of *SDlist* and *SElist* is required.

The second form displays a terse reminder of the first form, by issuing the command `which aovsum`.

Options

onewayopt is a string containing any of the options of the `oneway command`.

`keep` causes the synthetic data to be saved to three variables named (by default) `y_`, `cond_`, and `freq_`. Without the `keep` option, the synthetic data are discarded after `oneway` finishes its work.

`names` provides alternatives for the three variable names to receive the synthetic data if the `keep` option is specified.

Example 1

Consider an example having $K = 4$ groups with means and standard deviations given to three digit accuracy:

```
. aovsum, n(12 11 9 9) m(18.1 28.2 48.3 70.2) sd(8.54 14.3 12.4 14.0)
     Groups |    Summary of Response variable
    (cells) |        Mean   Std. Dev.       Obs.
------------+-----------------------------------
          1 |        18.1        8.54         12
          2 |        28.2        14.3         11
          3 |        48.3        12.4          9
          4 |        70.2          14          9
------------+-----------------------------------
      Total |    38.87561    23.29631         41
                        Analysis of Variance
    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between groups      16063.4956      3   5354.49854     35.09     0.0000
 Within groups       5645.2276     37   152.573719
------------------------------------------------------------------------
    Total           21708.7232     40    542.71808
Bartlett's test for equal variances:  chi2(3) =   2.9876  Prob>chi2 = 0.394
```

Retaining the synthetic data with `keep` enables several other possibilities. Prefacing `aovsum` with `quietly` will suppress the ANOVA output, useful when only the synthetic data are of interest.

```
. quietly aovsum, n(12 11 9 9) m(18.1 28.2 48.3 70.2) sd(8.54 14.3 12.4 14.0) keep
. list freq_ y_ cond_
        freq_          y_     cond_
  1.       11   20.565286         1
  2.       10   32.511612         2
  3.        8   52.433333         3
  4.        8   74.866667         4
  5.        1  -9.0181421         1
  6.        1  -14.916122         2
  7.        1   15.233333         3
  8.        1   32.866667         4
```

Many standard analyses can be now performed by including the variable `freq_` as an `fweight`. Suppose, for example, that `cond_` is a so-called continuous variable, and that we wish to investigate a linear trend in the mean of the response variable.

Either the **regress** or the **anova** command could be used for that purpose, but first make a copy of the variable **cond** to permit a small trick with **anova**:

```
. gen byte Cond_ = cond_
. ** Test for linear trend and deviation from linear trend:
. anova y_ cond_ Cond_ [fw=freq_], cont(cond_) seq
                           Number of obs =      41     R-squared     =  0.7400
                           Root MSE      = 12.3521     Adj R-squared =  0.7189

                  Source |    Seq. SS     df       MS           F     Prob > F
              -----------+----------------------------------------------------
                   Model |  16063.4956     3  5354.49854      35.09     0.0000
                         |
                   cond_ |  15653.4326     1  15653.4326     102.60     0.0000
                   Cond_ |  410.063011     2  205.031506       1.34     0.2733
                         |
                Residual |   5645.2276    37  152.573719
              -----------+----------------------------------------------------
                   Total |  21708.7232    40   542.71808
```

The entry for **cond_** provides the usual ANOVA test for the linear contrast in the means of the response variable, while the **Cond_** entry gives the ANOVA test for deviation from linear trend in the means.

The means and standard deviations for this example were computed from individual data given in Table 12.11.1 of Snedecor and Cochran (1980); those data are included with this insert as **ldose.dta**. Their Table 12.11.2 gives the ANOVA summary table, and their Table 12.11.3 gives the test for linear trend and deviation from linearity. The output from **aovsum** and **anova** shown above may be compared with those tables. Alternatively, "exact" results for the trend analysis tests can be obtained thus:

```
. use ldose, clear
(Snedecor & Cochran Table 12.11.1)

. describe

Contains data from ldose.dta
  obs:            41                          Snedecor & Cochran Table 12.11.1
 vars:             2                          3 Jan 2000 13:32
 size:           246 (95.5% of memory free)
-------------------------------------------------------------------------------
   1. ldose     byte   %8.0g                  Lethal dose (minus 50 units)
   2. rate      byte   %8.0g                  Injection rate,
                                                (mg/kg/min)/1045.75
-------------------------------------------------------------------------------
Sorted by:

.
. ** Create a copy of the group variable, again:

. gen byte Rate = rate

.
. ** And run -anova-, again:

. anova ldose rate Rate, cont(rate) seq
                           Number of obs =      41     R-squared     =  0.7402
                           Root MSE      = 12.3574     Adj R-squared =  0.7191

                  Source |    Seq. SS     df       MS           F     Prob > F
              -----------+----------------------------------------------------
                   Model |  16094.2817     3  5364.76055      35.13     0.0000
                         |
                    rate |  15683.5493     1  15683.5493     102.70     0.0000
                    Rate |  410.732323     2  205.366162       1.34     0.2730
                         |
                Residual |  5650.10859    37  152.705637
              -----------+----------------------------------------------------
                   Total |  21744.3902    40  543.609756
```

Example 2

Here, **aovsum** will silently create three new variables, **days**, **group**, and **freq_**:

```
. quietly aovsum, n(13 14 20 26 20 20 16 17)   /*
>    */    m(13.62 15.79 16.55 7.00 31.75 10.45 20.06 19.18)  /*
>    */    sd(12.07 17.49 14.75 6.11 22.07 10.68 14.22 17.90) /*
>    */    keep names(days group)
```

There are eight groups of Australian children and the response variable is days absent from school. The means and standard deviations were computed from Table 4 of Paul and Banerjee (1998), a set of data originally collected by Quine (1975). The groups form a 2 (race) $\times$ 4 (grade in school) cross-classification. Variables encoding those two factors could be entered in Stata's data editor, or created with code resembling

```
. gen byte race = mod(group-1, 2)
. gen byte grade = int((group-1)/2)
```

Then, the usual two-way ANOVA of the response variable can be computed as

```
. anova days race grade race*grade [fw=freq_]
                           Number of obs =     146     R-squared     =  0.2120
                           Root MSE      = 14.8357     Adj R-squared =  0.1720

                  Source |  Partial SS    df       MS           F     Prob > F
              -----------+----------------------------------------------------
                   Model |  8169.29067     7  1167.04152       5.30     0.0000
                         |
                    race |  1907.27695     1  1907.27695       8.67     0.0038
                   grade |  2259.47976     3  753.159921       3.42     0.0191
              race*grade |  2896.15272     3  965.384239       4.39     0.0056
                         |
                Residual |  30373.3948   138  220.097064
              -----------+----------------------------------------------------
                   Total |  38542.6855   145  265.811624
```

Quine's (1975) raw data are included as `absences.dta`; the "exact" two-way ANOVA is thus:

```
. use absences, replace
(Quine´s School Absences Data)

. describe

Contains data from absences.dta
  obs:           146                          Quine´s School Absences Data
 vars:             4                          3 Jan 2000 12:42
 size:         1,168 (95.2% of memory free)   (_dta has notes)
-------------------------------------------------------------------------------
   1. days      byte   %8.0g                  Days absent from school
   2. group     byte   %8.0g                  Race x Grade group
   3. race      byte   %8.0g       race       Race of child
   4. grade     byte   %9.0g       grade      Grade in school
-------------------------------------------------------------------------------
Sorted by:

.
. ** And the ANOVA:

. anova days race grade race*grade
                           Number of obs =     146     R-squared     =  0.2119
                           Root MSE      = 14.8369     Adj R-squared =  0.1720

                  Source |  Partial SS    df       MS           F     Prob > F
              -----------+----------------------------------------------------
                   Model |  8169.67223     7  1167.09603       5.30     0.0000
                         |
                    race |  1908.01254     1  1908.01254       8.67     0.0038
                   grade |  2259.85192     3  753.283973       3.42     0.0191
              race*grade |  2895.42955     3  965.143183       4.38     0.0056
                         |
                Residual |  30378.4922   138  220.134001
              -----------+----------------------------------------------------
                   Total |  38548.1644   145   265.84941
```

Saved Results

After a call to `aovsum`, the contents of `r()` will be the same as following a call to `oneway` with the options specified in *onewayopt*.

References

Paul, S. R. and T. Banerjee. 1998. Analysis of two-way layout of count data involving multiple counts per cell. *Journal of the American Statistical Association* 93: 1419–1429.

Quine, S. 1975. Unpublished Ph.D. thesis, Australian National University.

Snedecor, G. W. and W. G. Cochran. 1980. *Statistical Methods*. 7th ed. Ames, IA: Iowa State University Press.

sg133 Sequential and drop one term likelihood-ratio tests

Zhiqiang Wang, Menzies School of Health Research, Darwin, Australia, wang@menzies.edu.au

Abstract: Commands extending Stata's `lrtest` command are given for likelihood-ratio tests after maximum likelihood estimation.

Keywords: Likelihood-ratio tests, Akaike information criterion, AIC.

Syntax

`lrseq` [`, fp(`*format*`) fchi(`*format*`)`]

`lrdrop1` [`, fp(`*format*`) fchi(`*format*`)`]

Description

This insert includes two commands: `lrdrop1` and `lrseq`. Both commands perform likelihood-ratio tests after maximum likelihood estimations such as those by `stcox`, `logit`, `logistic`, `poisson`, and so on. They are extensions of the Stata command `lrtest`.

The `lrseq` command is designed to perform sequential pairwise likelihood-ratio tests. It starts with a null model. Explanatory variables are added into the model sequentially. At each step, the current model is compared with the previous one. The `lrdrop1` command performs likelihood ratio tests after dropping terms from the original model one at a time in turn, comparing each new model with the original model. Both commands report the Akaike information criterion (AIC) developed by Akaike (1974). For a model having r parameters,

$$\text{AIC} = -2\log(\text{likelihood}) + 2r$$

Analysts can use these two commands to select models interactively. Alternatively, the command `swaic` (see Wang 2000) can be used for an automatic model selection using AIC. Another relevant command is `lrtest2` introduced by Perez–Hoyos and Tobias (1999).

Options

`fp(`*format*`)` specifies the output format for p-values with default value `%9.4f`.

`fchi(`*format*`)` specifies the output format for `chi2` values, with default value `%9.2f`.

Examples

We begin with a call to `logit`:

```
. xi: logit outcome age sex i.expose hibp bmi, nolog
i.expose             Iexpos_1-3   (naturally coded; Iexpos_1 omitted)

Logit estimates                                   Number of obs   =        399
                                                  LR chi2(6)      =      42.13
                                                  Prob > chi2     =     0.0000
Log likelihood =  -104.4184                       Pseudo R2       =     0.1679

------------------------------------------------------------------------------
 outcome |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   .0349601   .0157084      2.226   0.026       .0041721     .065748
     sex |    .390423   .3878383      1.007   0.314       -.369726    1.150572
Iexpos_2 |   1.333849   .6090427      2.190   0.029       .1401477    2.527551
Iexpos_3 |   2.601842   .5930699      4.387   0.000       1.439446    3.764237
    hibp |   .0042039   .3858058      0.011   0.991      -.7519616    .7603694
     bmi |  -.1136088   .0425776     -2.668   0.008      -.1970594   -.0301582
   _cons |  -2.749143   1.460547     -1.882   0.060      -5.611763    .1134774
------------------------------------------------------------------------------
```

Now we use `lrdrop1`:

```
. lrdrop1
Likelihood Ratio Tests: drop 1 term
logit regression
number of obs = 399
------------------------------------------------------------------------
 outcome     Df      Chi2      P>Chi2    -2*log ll   Res. Df   AIC
------------------------------------------------------------------------
Original Model                            208.84      392     222.84
    -age      1       5.12      0.0237    213.95      391     225.95
    -sex      1       1.02      0.3123    209.86      391     221.86
-Iexpos*      2      27.67      0.0000    236.51      390     246.51
   -hibp      1       0.00      0.9913    208.84      391     220.84
    -bmi      1       8.11      0.0044    216.95      391     228.95
------------------------------------------------------------------------
Terms dropped one at a time in turn.
```

and then `lrseq`:

```
. lrseq
Sequential Likelihood Ratio Tests
logit regression
number of obs = 399
------------------------------------------------------------------------
 outcome     Df      Chi2      P>Chi2    -2*log ll   Res. Df   AIC
------------------------------------------------------------------------
Null Model                                250.96      398     252.96
     age      1       9.34      0.0022    241.62      397     245.62
     sex      1       0.36      0.5480    241.26      396     247.26
 Iexpos*      2      24.32      0.0000    216.95      394     226.95
    hibp      1       0.00      0.9859    216.95      393     228.95
     bmi      1       8.11      0.0044    208.84      392     222.84
------------------------------------------------------------------------
Terms added sequentially (first to last)
```

One `lrseq` or `lrdrop1` command in this example is equivalent to performing `lrtest` five times. The likelihood-ratio test for variable `sex` in the `lrseq` output; for example, compares a model with `age` and `sex` against a model with only `age`. `Iexpos*` is for the categorical variable `expose` with three categories and two degrees of freedom.

Acknowledgments

I thank Dr. John L. Moran, Queen Elizabeth Hospital, Adelaide, Australia for useful suggestions and Dr. Wendy Hoy, Menzies School of Health Research, Darwin, Australia for providing the example data.

References

Akaike, H. 1974. A new look at statistical model identification. *IEEE Transactions on Automatic Control* AC–19: 716–723.

Perez-Hoyos, S. and A. Tobias. 1999. sg111: A modified likelihood-ratio test command. *Stata Technical Bulletin* 49: 24–25.

Wang, Z. 2000. sg134: Model selection using the Akaike information criterion. *Stata Technical Bulletin* 54: 47–49.

sg134 | Model selection using the Akaike information criterion

Zhiqiang Wang, Menzies School of Health Research, Darwin, Australia, wang@menzies.edu.au

Abstract: A command for performing stepwise model selection using the Akaike information criterion is described and illustrated.

Keywords: Stepwise model selection, Akaike information criterion, AIC.

Syntax

`swaic` [`,` `fp(`*format*`)` `fchi(`*format*`)` `back` `model`]

Description

The command `swaic` in this insert is designed to perform stepwise model selection using the Akaike Information Criterion (AIC) developed by Akaike (1974) after maximum likelihood estimation. For a model having r parameters,

$$\text{AIC} = -2\log(\text{likelihood}) + 2r$$

It is an alternative approach to stepwise model selection in Stata. `swaic` starts with a null or full model. It takes a step by adding or dropping a term that produces the minimum AIC. `swaic` reports likelihood ratio tests as well as AIC values for all steps. With the `model` option, `swaic` reports the final model with the minimum AIC value. The current version of `swaic` works with `logit`, `logistic`, `stcox`, `poisson`, `probit`, and `streg`.

Options

`fp(`*format*`)` specifies the output format for p-values, with default value `%9.4f`.

`fchi(`*format*`)` specifies the output format for `chi2` values, with default value `%9.2f`.

`back` uses a backward method starting with a full model, the default is a forward method.

`model` reports a final model having the minimum AIC value.

Example

We illustrate `swaic` by first using `logit` and then using `swaic` with the `model` option:

```
. xi: logit outcome age sex i.expose hibp bmi, nolog
i.expose             Iexpos_1-3   (naturally coded; Iexpos_1 omitted)

Logit estimates                                   Number of obs   =        399
                                                  LR chi2(6)      =      42.13
                                                  Prob > chi2     =     0.0000
Log likelihood =  -104.4184                       Pseudo R2       =     0.1679

------------------------------------------------------------------------------
 outcome |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   .0349601   .0157084      2.226   0.026       .0041721     .065748
     sex |    .390423   .3878383      1.007   0.314       -.369726    1.150572
Iexpos_2 |   1.333849   .6090427      2.190   0.029       .1401477    2.527551
Iexpos_3 |   2.601842   .5930699      4.387   0.000       1.439446    3.764237
    hibp |   .0042039   .3858058      0.011   0.991      -.7519616    .7603694
     bmi |  -.1136088   .0425776     -2.668   0.008      -.1970594   -.0301582
   _cons |  -2.749143   1.460547     -1.882   0.060      -5.611763    .1134774
------------------------------------------------------------------------------

. swaic, model
Stepwise Model Selection by AIC
logit regression
number of obs = 399
------------------------------------------------------------------------
 outcome            Df     Chi2   P>Chi2    -2*ll    Df Res.   AIC
------------------------------------------------------------------------
Null Model                                  250.96    398     252.96
Step 1:  Iexpos*     2    22.91   0.0000    228.06    396     234.06
Step 2:      bmi     1    13.27   0.0003    214.79    395     222.79
Step 3:      age     1     4.85   0.0276    209.94    394     219.94
Step 4:      sex     1     1.10   0.2934    208.84    393     220.84
Step 5:     hibp     1     0.00   0.9913    208.84    392     222.84
------------------------------------------------------------------------

Logit estimates                                   Number of obs   =        399
                                                  LR chi2(4)      =      41.02
                                                  Prob > chi2     =     0.0000
Log likelihood = -104.97047                       Pseudo R2       =     0.1635

------------------------------------------------------------------------------
 outcome |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
Iexpos_2 |   1.293822   .5981625      2.163   0.031       .1214446    2.466199
Iexpos_3 |   2.535178    .580218      4.369   0.000       1.397971    3.672384
     bmi |  -.1072528   .0412597     -2.599   0.009      -.1881203   -.0263853
     age |   .0338024   .0156032      2.166   0.030       .0032206    .0643842
   _cons |  -2.608527   1.449736     -1.799   0.072      -5.449958    .2329043
------------------------------------------------------------------------------
```

Then we use both the `model` and `back` options:

```
. swaic, model back
Stepwise Model Selection by AIC
logit regression
number of obs = 399
-----------------------------------------------------------------------
 outcome           Df     Chi2   P>Chi2    -2*ll    Df Res.   AIC
-----------------------------------------------------------------------
Full Model                                 208.84    392     222.84
Step 1:    -hibp    1     0.00   0.9913    208.84    393     220.84
Step 2:     -sex    1     1.10   0.2934    209.94    394     219.94
Step 3:     -age    1     4.85   0.0276    214.79    395     222.79
Step 4:     -bmi    1    13.27   0.0003    228.06    396     234.06
Step 5: -Iexpos*    2    22.91   0.0000    250.96    398     252.96
-----------------------------------------------------------------------
Logit estimates                                   Number of obs   =        399
                                                  LR chi2(4)      =      41.02
                                                  Prob > chi2     =     0.0000
Log likelihood = -104.97047                       Pseudo R2       =     0.1635
------------------------------------------------------------------------------
 outcome |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   .0338024   .0156032      2.166   0.030       .0032206    .0643842
Iexpos_2 |   1.293822   .5981625      2.163   0.031       .1214446    2.466199
Iexpos_3 |   2.535178    .580218      4.369   0.000       1.397971    3.672384
     bmi |  -.1072528   .0412597     -2.599   0.009      -.1881203   -.0263853
   _cons |  -2.608527   1.449736     -1.799   0.072      -5.449958    .2329043
------------------------------------------------------------------------------
```

Both backward and forward methods produce the same results in this example. The AIC reaches a minimum of 219.94 when the model only includes `age`, `bmi` and `expose`.

Acknowledgments

Thanks to Dr. John L. Moran, Queen Elizabeth Hospital, Adelaide, Australia for testing and comments and Dr. Wendy Hoy of the Menzies School of Health Research in Darwin, Australia for providing examples.

Reference

Akaike, H. 1974. A new look at statistical model identification. *IEEE Transactions on Automatic Control*, AC–19: 716–723.

sxd1.2 Random allocation of treatments balanced in blocks: update

Philip Ryan, University of Adelaide, South Australia, pryan@medicine.adelaide.edu.au

Abstract: Allocation of treatments to subjects using a random method underpins the validity of a clinical trial. Blocking is a technique that helps ensure a constant ratio of treatment allocations is maintained throughout the randomization period. `ralloc` facilitates blocked randomization in a variety of experimental designs including stratified, factorial, and crossover designs. Output may be configured in several ways to suit central or distributed randomization, and to facilitate the pharmacy preparing blocks of treatments.

Keywords: Experimental design, randomized controlled trials, randomization, blocking.

Introduction

I have again updated the program `ralloc`, first described in Ryan (1998) and updated in Ryan (1999). The full syntax is

`ralloc` *BlockIdvar BlockSizevar Treatmentvar* `,` `saving(`*filename*`)` [{ `multif` | `nomultif` } `seed(`*#*`)`

`nsubj(`*#*`)` `ntreat(2|3|4)` `ratio(1|2|3)` `osize(1|2|3|4|5|6|7)` `init(`*#*`)` { `equal` | `noequal` }

`strata(`*#*`)` `using(`*filename*`)` `countv(`*varname*`)` { `tables` | `notables` } `trtlab(`*label1* [*label2* ...]`)`

`matsiz(`*#*`)` `fratio( 1 1 | 2 1 | 1 2 | 2 2 )` `factor(2*2|2*3|3*2|3*3|2*4|4*2|3*4|4*3)`

`xover(`{ `stand` | `switch` | `extra` }`)` `shape(`{ `long` | `wide` }`)`]

`ralloc ?`

New Features

The version of `ralloc` accompanying this insert has the following new features:

1. The second syntax displays the first syntax on screen.
2. The program supports stratified randomization.
3. The program supports two-treatment factorial designs.
4. The program supports a 2×2 crossover design with or without either a "switchback" or "extra-period" of treatment in a third period as described by Jones and Kenward (1989).
5. A new, more efficient treatment labeling option is used.
6. The display of informative tables is now optional.

New options

`multif` specifies that, for a stratified design, one file will be saved for the allocations in each stratum, in addition to a file storing all allocations. The *filename* specified in the `saving` option will be used as a stub to name the files according to the following schema

```
<filename>_n1[_n2_n3 ... _nk]
```

for a trial with 1 to k stratification variables. `n1` identifies the level of the stratum of the 1st stratification variable, `n2` gives the level of the stratum of the 2nd stratification variable, and so on, each stratification variable's set of suffixes being preceded by an underscore character. Suffixes are padded with leading zeros to maintain alphanumeric sort order. The default is `nomultif`.

`strata(`*#*`)` specifies the number of strata and may be calculated as the product, over all stratifying variables, of the number of levels in each variable. A new variable, `StratID`, denoting the stratum identifier, is generated. The default is `strata(1)`. `strata` is overridden by the specification of a `using` file.

`using(`*filename*`)` names a file whose data define the stratification schema. The file must consist solely of variables defining strata plus one other variable giving the number of subjects required to be randomized in each stratum (the `countv` variable, see below). Each row (observation) of the file defines a stratum. Levels of a stratification variable must be coded as consecutive positive integers $(1, 2, 3, \ldots)$. `ralloc` will check this and will also check that the product of levels over all stratification variables equals the number of rows (strata). Whether strata are defined by `strata` or by the rows of a using file, the number of strata cannot exceed 800.

`countv(`*varname*`)` specifies the variable in the using file whose values give the number of subjects requiring randomization in each stratum. `countv` is specified if and only if a using file is specified. Values of `countv` override the value of `nsubj` should this also be specified.

`tables` specifies that a frequency distribution of block sizes is displayed for all allocations and, where appropriate, for each stratum. The default is `notables`.

`trtlab(`*label1* [*label2* ...]`)` allows specification of value labels for treatments. At most four labels may be specified for a nonfactorial design. The number of labels that may be specified for a factorial design is equal to the sum of the number of possible treatments in the two randomization axes. Labels are separated by spaces and so may not themselves contain a space. A label will be truncated after the first eight characters. The default treatment labels are A, B, C and D (plus E, F and G if required for a factorial design). An older form of the syntax for nonfactorial designs, requiring an option for each label by `tr1lab(`*label*`)`, `tr2lab(`*label*`)`, and so on, is permitted but obsolete.

`matsiz(`*#*`)` sets the maximum size of a Stata matrix. This is a rarely used option, as `ralloc` chooses a matrix size appropriate to the stratification schema specified.

`fratio(`*string*`)` specifies, in the case of a 2×2 factorial design, the ratio of allocations in each axis. The string must be one of the choices given in the syntax diagram. For example, if we require a 1:2 ratio of treatments in the first randomization axis and a 1:1 ratio of treatments in the second axis, `fratio(2 1)` would be specified.

`factor(`*string*`)` specifies that the trial has a factorial design with two "axes of randomization". The string must be one of the choices in the syntax diagram. Allocation combinations are balanced within blocks, unless `fratio` is specified in a 2×2 design. The names of the two treatment variables generated will be the name specified by *Treatmentvar* followed by a 1 and a 2.

`xover(`*string*`)` specifies the design as a 2×2 crossover. The argument may be one of `stand` for the standard 2 treatment, 2 period design, `switch` for the switchback design where each subject receives the treatment assigned for period 1 in period 3, or `extra`, for the extra period design, where each subject has the treatment assigned for period 2 replicated in period 3. The names of the treatment variables generated will be that specified by *Treatmentvar* followed by a 1, 2, and if required, a 3.

Example 1

To illustrate the new `trtlab`, `strata` and `multif` options, we have

```
. ralloc blknum blksiz Rx, ns(494) osiz(2) eq ntreat(2) sav(mywide)
>       shap(wide) trtlab(Placebo Active) strata(4) multif
```

which results in the allocation of two treatments labeled “Placebo” and “Active” equally in two block sizes, 2 and 4, to 494 subjects in each of four strata (maybe a four-center trial). Data are saved in wide form to five files: `mywide.dta` holds all allocations, and four additional files named `mywide_1.dta`, `mywide_2.dta`, `mywide_3.dta` and `mywide_4.dta` hold stratum specific allocations.

```
. use mywide_4
. li in 1/7, noobs nodisp

StratID   blknum   blksiz        Rx1        Rx2        Rx3        Rx4
      4      498        2     Active    Placebo          .          .
      4      499        2    Placebo     Active          .          .
      4      500        2    Placebo     Active          .          .
      4      501        4    Placebo    Placebo     Active     Active
      4      502        2    Placebo     Active          .          .
      4      503        2    Placebo     Active          .          .
      4      504        2     Active    Placebo          .          .
```

Example 2

To illustrate the `using` option, we have a file, `raltest6.dta`, defining strata for an RCT to be conducted in 3 centers. We also seek to balance allocations within two age groups. The required numbers of allocations in each of the $3 \times 2 = 6$ strata are given by the variable `freq`.

```
. use raltest6
. list
        centre       freq    agegrp
  1.         1         50         1
  2.         1         80         2
  3.         2        140         1
  4.         2        100         2
  5.         3         70         1
  6.         3        100         2
```

Note that `ralloc` does not care about the order of variables in the data, nor of the sort order of the observations, but it is easier to check the completeness of the schema if levels are coherently nested. The command

```
. ralloc bID bsiz trt, sav(myrct) count(freq) using(raltest6)
>   nsubj(80) seed(54109) multif
```

results in the generation of seven files. Note that the option `nusbj(80)` will be overridden by the values of `freq` in `raltest.dta`. After some informative output (not shown here), the data are written to the appropriate stratum-specific files and the file with all allocations, `myrct.dta`, is in memory. The stratum identifying variables have also been written to the datasets.

```
. li in 1/8, noob nodis

 StratID    centre     agegrp      bID     bsiz   SeqInBlk   trt
       1         1          1        1        2          1     A
       1         1          1        1        2          2     B
       1         1          1        2        8          1     A
       1         1          1        2        8          2     B
       1         1          1        2        8          3     B
       1         1          1        2        8          4     B
       1         1          1        2        8          5     A
       1         1          1        2        8          6     A
```

Example 3

Consider a study that aims to test both the efficacy of a blood pressure lowering medication, called BPzap, versus a placebo, and the utility of two weight reduction exercise programs, called GymSweat and JogaBit, versus normal activity, on a specified cardiovascular endpoint. An efficient design might be a 2×3 factorial RCT. The command

```
. ralloc blknum size Rx, sav(rctfact) factor(2*3) osiz(2) eq
>  seed(4512) trtlab(BPzap Placebo GymSweat JogaBit normact)
>  nsubj(300)
```

will allocate two treatments, called Rx1 and Rx2, to each of 300 subjects in a single stratum using a 2×3 factorial design. Blocks of size 6 and 12 with equal frequency will result.

```
. list in 1/10
        StratID    blknum      size   SeqInBlk        Rx1        Rx2
  1.          1         1         6          1    Placebo   GymSweat
  2.          1         1         6          2      BPzap    normact
  3.          1         1         6          3      BPzap   GymSweat
  4.          1         1         6          4    Placebo    normact
  5.          1         1         6          5      BPzap    JogaBit
  6.          1         1         6          6    Placebo    JogaBit
  7.          1         2        12          1      BPzap    JogaBit
  8.          1         2        12          2      BPzap   GymSweat
  9.          1         2        12          3      BPzap    normact
 10.          1         2        12          4    Placebo    JogaBit
. tab Rx1 Rx2
           |                Rx2
       Rx1 |  GymSweat    JogaBit    normact |     Total
-----------+---------------------------------+----------
     BPzap |        50         50         50 |       150
   Placebo |        50         50         50 |       150
-----------+---------------------------------+----------
     Total |       100        100        100 |       300
```

and we note the balance in allocations in each axis of the study.

Example 4

We reformulate the preceding study as a 2×2 factorial design by excluding the JogaBit treatment. Let's say we wish to have twice as many on Placebo as BPzap, and also twice as many subjects on normal activity as on the GymSweat regimen.

```
. ralloc blknum size Rx, sav(rctfact2) factor(2*2) osiz(2) eq
>  seed(1131) trtlab(BPzap Placebo GymSweat normact) fratio(2 2)
>  nsubj(300)
```

This command will give blocks of sizes 9 (the minimum possible with 1:2 allocation ratios in each axis) and 18 (because `osize(2)` was specified).

```
. tab Rx*
           |          Rx2
       Rx1 |  GymSweat    normact |     Total
-----------+----------------------+----------
     BPzap |        34         68 |       102
   Placebo |        68        136 |       204
-----------+----------------------+----------
     Total |       102        204 |       306
```

Example 5

We have a 2×2 crossover design supplemented by a switchback in period 3. The trial compares a new antiarthritic drug "HipLube" versus aspirin in chronic osteoarthritis of the hip.

```
. ralloc Bnum Bsize medic, saving(chronOA) ns(28) osiz(1) init(4)
>  trtlab(HipLube aspirin) xover(switch) strata(2)
```

`medic1`, `medic2`, and `medic3` store the treatments administered in periods 1, 2 and 3 respectively:

```
. li in 1/6, noobs nodisp
```

```
StratID   Bnum   Bsize   SeqInBlk     medic1    medic2    medic3
      1      1       4          1    HipLube   aspirin   HipLube
      1      1       4          2    aspirin   HipLube   aspirin
      1      1       4          3    HipLube   aspirin   HipLube
      1      1       4          4    aspirin   HipLube   aspirin
      1      2       4          1    HipLube   aspirin   HipLube
      1      2       4          2    HipLube   aspirin   HipLube
```

Acknowledgments

I thank Liddy Griffith, Senior Data Manager, Department of Public Health, University of Adelaide, for helpful comments. John Gleason reminded me of the use of the second syntax.

References

Jones B. and M. G. Kenward. 1989. *Design and Analysis of Crossover Trials*. London: Chapman and Hall.

Ryan, P. 1998. Random allocation of treatments in blocks. *Stata Technical Bulletin* 41: 43–46. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 297–300.

——. 1999. Update to random allocation of treatments in blocks. *Stata Technical Bulletin* 50: 36–37.

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	*qs*	questions and suggestions
dt	datasets	*tt*	teaching
gr	graphics	*zz*	not elsewhere classified
in	instruction		
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 (979–845–3144) 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