STATA TECHNICAL BULLETIN

January 1998

STB-41

A publication to promote communication among Stata users

Editor

H. Joseph Newton
Department of Statistics
Texas A & M University
College Station, Texas 77843
409-845-3142
409-845-3144 FAX
stb@stata.com EMAIL

Associate Editors

Francis X. Diebold, University of Pennsylvania
Joanne M. Garrett, University of North Carolina
Marcello Pagano, Harvard School of Public Health
James L. Powell, UC Berkeley and Princeton University
J. Patrick Royston, Imperial College School of Medicine

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

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

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

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

Contents of this issue

stata51	Exact McNemar test added to mcc and mcci commands

The `mcc` and `mcci` commands in [R] **epitab** now calculate and report the exact McNemar statistic significance probability for all 2×2 tables. The exact test is obtained from the binomial distribution with $\pi = 1/2$ and conditioning on the sum of the two off-diagonal cells. See, for example, Agresti (1992).

Syntax and output

The commands' syntax has not changed; the output has been modified. This is demonstrated using the immediate test example that appears in [R] **epitab**.

```
. mcci 8 8 3 8
                           Controls
Cases              |   Exposed   Unexposed  |     Total
-------------------+------------------------+----------
         Exposed   |         8           8  |        16
       Unexposed   |         3           8  |        11
-------------------+------------------------+----------
           Total   |        11          16  |        27

McNemar´s chi2(1) =      2.27        Pr>chi2 = 0.1317
Exact McNemar significance probability       = 0.2266

Proportion with factor
        Cases       .5925926
        Controls    .4074074     [95% conf. interval]
                    ---------    --------------------
        difference  .1851852     -.0822542   .4526246
        ratio       1.454545       .891101   2.374257
        rel. diff.     .3125     -.0243688   .6493688
        odds ratio  2.666667      .6400699   15.60439   (exact)
```

The output is the same as before, except, that the new test "Exact McNemar significance probability" now prints below the McNemar's χ^2 tests.

The result from the exact test is saved in the `S_24` macro.

References

Agresti A. 1992. A survey of exact inferences for contingency tables. *Statistical Sciences* 7: 131–177.

dm53	Detection and deletion of duplicate observations

Thomas J. Steichen, RJRT, FAX 910-741-1430, steicht@rjrt.com
Nicholas J. Cox, University of Durham, UK, FAX (011)-44-91-374 2456, n.j.cox@durham.ac.uk

Syntax

`dups` [*varlist*] [`, drop` `expand`(*varname*) `key`(*varlist2*) `unique` `terse`]

Description

`dups` provides information about unique and duplicate observations in the data file and, optionally, drops all duplicate observations.

varlist is an optional variable list that determines which observations are duplicates: observations must match exactly on all variables in the list to be duplicates. If no *varlist* is given, then all variables in the data file are used to determine duplicates.

Options

`drop` causes duplicate observations to be dropped from the data file. `drop` must be spelled out completely. `drop` creates an expand variable to allow dropped data to be recreated. The default expand variable name is `_expand`. If `_expand` exists, an error message is reported and no data are dropped. The expand variable will contain the number of duplicate copies of the observations in the original data file. A subsequent `expand` command will completely resurrect the original data only if *varlist* was not specified in the `dups` command (or, equivalently, if *varlist* contains all variables in the data file), or if the unspecified variables are constant within the subgroups formed by the specified variables. The data can be partially, but not fully, resurrected if a limited *varlist* was used—unique information from the variables not in *varlist* cannot be recovered.

`expand(`*varname*`)` specifies a *varname* to be used as the expand variable in place of the default name, `_expand`. (This option has no effect unless option `drop` is also included.) If `drop` is included and the specified *varname* exists, an error message is given and no data are dropped.

`key(`*varlist2*`)` causes the value of the variables in *varlist2* to be added to the output displayed for each group. *varlist2* should be the same as, or a subset of, *varlist*. If *varlist2* is assigned value `*` then *varlist2* will be set the same as *varlist*.

`unique` causes the default display and option `key()` to list information for unique observations also.

`terse` limits the default display output. When specified, only the number of duplicate groups, total observations, number of observations in duplicates, and number of unique observations are shown. Without `terse`, `dups` will number the duplicate groups and provide the observation count in each group, and will do the same for unique observations, if any, when `unique` is specified. Specifying `terse` cancels both `key()` and `unique`.

Remarks

Data files generated from multiple sources, by accretion over time, or by merging of smaller data files occasionally have *duplicate* observations that need to be identified and eliminated. Other data files are expected to have only *unique* observations defined by all or a subset of the variables in the file. The existence of duplicates on the defining variables usually indicates data entry errors that require detection and correction. Alternatively, some data files are expected to have multiple observations on key variables in the file. In these files, the existence of one or more unique observations would indicate the data entry errors that require detection and correction.

`dups`, which is implemented in this insert, provides a simple method to detect and report information about unique and duplicate observations in the data file and, if needed, to drop duplicate observations.

Examples

Using the `auto.dta` dataset, `dups` provides the following output:

```
. dups
group by: make price mpg rep78 hdroom trunk weight length turn displ gratio for
> eign
groups formed: 0
```

In this example `dups` defaulted to the full variable list in `auto.dta` because *varlist* was not supplied. As expected, no duplicates were found and no duplicate groups were formed.

```
. dups foreign, key(*) unique
group by: foreign
groups formed: 2
groups of duplicate observations:
  pt_group   pt_count   foreign
         1         22   Foreign
         2         52  Domestic
no unique observations found
```

In this example `dups` is asked to define duplicates based on variable `foreign`. Option `key(*)` specifies that `dups` should include, as an identifier in the listing of duplicate groups, the value of `foreign` (i.e., the only variable specified in *varlist*). (`*` tells `dups` to use *varlist* as the value of *varlist2*.) Option `unique` requests `dups` to report on unique observations—but none were found.

```
. replace make = "Chev. Impala" in 16
. dups make, key(*)
group by: make
groups formed: 1
groups of duplicate observations:
  pt_group   pt_count                make
         1          2        Chev. Impala
```

In this example we first modify `auto.dta` to create a partially duplicated observation, then run `dups` on variable `make`. As expected, one group of duplicates is found. The same analysis is shown below with option `terse` specified.

```
. dups make, key(*) terse
group by: make
groups formed: 1
total observations:    74
```

```
in duplicates        2
in unique           72
```

Since detection and reporting of duplicates is the default mode of `dups`, no observations were dropped from the data file in any of the preceding examples. The user must actively specify option `drop` to change the data file, and like other destructive options in Stata, `drop` must be fully spelled out. In this final example we return to the original `auto.dta` file and select option `drop`.

```
. dups foreign, key(*) unique drop
group by: foreign
groups formed: 2
groups of duplicate observations:
  pt_group   pt_count   foreign
         1         22   Foreign
         2         52  Domestic
no unique observations found
(72 observations deleted)
observations remaining: 2
```

For this example, `dups` provided the usual report but appended notes, `(72 observations deleted)`, that indicates that 72 (of the 74) observations in the data file were deleted and that, as expected, only two observations remain, `observations remaining: 2`. Because the *varlist* in this example (i.e., `foreign`) is not the complete *varlist*, it will be impossible to fully recover the original data file. (Command `expand _expand` will recreate a file with 22 'Foreign' and 52 'Domestic' observations, but all 'Foreign' observations will be identical, as will all 'Domestic' observations.)

gr16.2 Corrections to condraw.ado

J. Patrick Gray, University of Wisconsin-Milwaukee, jpgray@csd.uwm.edu
Nicholas J. Cox, University of Durham, UK, FAX (011) 44-91-374-2456, n.j.cox@durham.ac.uk

Nicholas Cox discovered that the `condraw.ado` program published in STB-23 as *gr16* does not plot convex hulls correctly when multiple rightmost or leftmost points exit. The problem does not affect the calculation of the hulls done by `conhull.ado`. The `condraw.ado` program has been updated to correct the problem. This update should be copied over the original `condraw.ado` file. An updated `condraw.hlp` file is also included.

The `chplot.ado` program written by Nick Cox (STB-36, *gr16.1*) calls `condraw.ado` and needed a minor modification to work with the updated program. The modified `chplot.ado` should be copied over the original `chplot.ado` file. The minor changes in `chplot.ado` do not require a new `chplot.hlp` file.

gr27 An adaptive variable span running line smoother

Peter Sasieni, Imperial Cancer Research Fund, London, p.sasieni@icrf.icnet.uk

Syntax

`autosmoo` *yvar* [*xvar*] [*weight*] [`if` *exp*] [`in` *range*] [`, kmin(#) kmax(#)`
`repeat(#) gen(`*newvar*`) genk(`*kvar*`) logit nograph` *graph_options*]

`aweight`s are allowed; see [U] **18.1.6 weight**.

Description

`autosmoo` smooths *yvar* on *xvar*. It is designed to be completely automatic, but some options are permitted. The smooth is a running line fit with a variable span. The span is chosen at each point by cross validation on the mean squared error of prediction. The span at each point is smoothed to produce the variable span used to smooth the data. A graph of *yvar* together with its smooth is plotted against *xvar*, unless suppressed. If *xvar* is not provided, *yvar* is smoothed against the ordered observations.

Options

`kmin(#)` is the minimum number of symmetric nearest neighbors on each side of an observation that are used in smoothing. The default is chosen by a complicated formula. It is at least 2 and generally no more than the integer part of `kmax` divided by 250. The default value is slightly larger in the presence of multiple ties in the *xvar*.

`kmax(#)` is the maximum number of symmetric nearest neighbors on each side of an observation that are used in smoothing. The default is $2 \times N^{4/5}$, but no more than $N/2$, where N is the total number of observations.

`repeat(#)` specifies the number of times the data are to be smoothed using the selected number of symmetric nearest neighbors. The default is 1. Use of `repeat` does not affect the value of *kvar* selected by `autosmoo`.

`gen(`*newvar*`)` creates *newvar* containing the smoothed values of *yvar*. Note that this will be on a logit scale if `logit` is used.

`genk(`*kvar*`)` creates *kvar* containing the number of nearest neighbors on either side of each observation used in the final smooth. This can be plugged back into `running`—see *sed9.1* in this issue.

`logit` transforms the smooth and plots the y-axis on a logit scale. 0–1 observations are automatically jittered in the vertical scale and are plotted outside the range of the smoothed curve. It is not necessary that *yvar* be 0 or 1 provided it takes values in $[0, 1]$.

`nograph` suppresses displaying the graph.

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

Examples

The first example (see Figure 1) uses the motorcycle crash data (Silverman 1985) provided as a Stata dataset by Salgado-Ugarte, Shimizu, and Taniuchi (1996). Notice how the smooth fit follows the first bend reasonably well and is not excessively variable at longer time values.

```
. label var time "Time (ms)"
. label var accel "Acceleration (g)"
. local options "yline(0) ylab(0,-50,-100,-150,50) xlab"
. autosmoo accel time, `options´
```

The option `repeat(2)` used in Figure 2 smooths out the slight jaggedness in the first fit, but does slightly less well at the first bend.

```
. autosmoo accel time, `options´ repeat(2) t1("Repeat = 2")
```

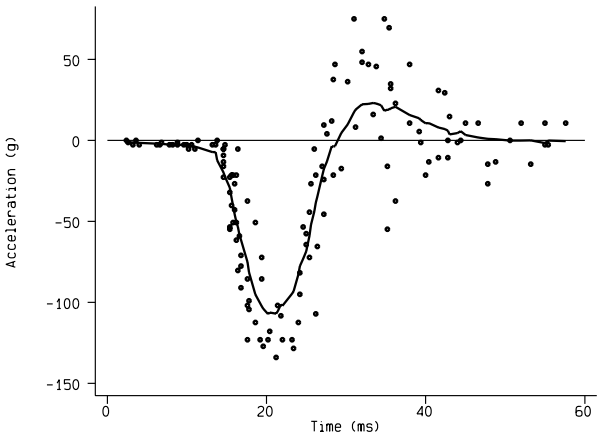

Figure 1. Smoothing the motorcycle data.

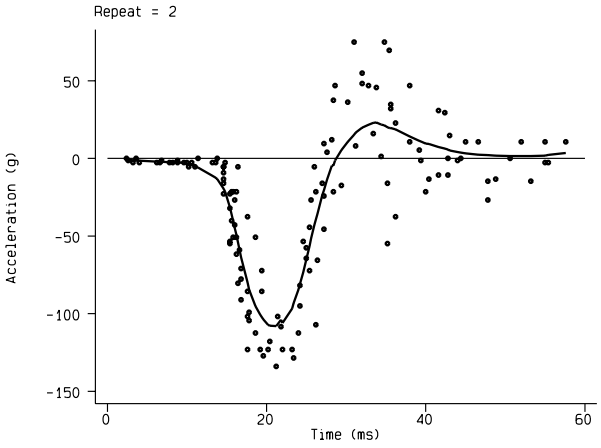

Figure 2. Smoothing the motorcycle data using the `repeat` option.

The second example uses simulated data suggested by Friedman (1984) as a difficult test for a variable span smoother. We have used 200 observations, but you might try using as few as 20 or as many as 20,000 (the latter will probably take between 3 seconds and 4 minutes depending on the speed of the computer).

```
. set obs 200
. gen x = uniform()
. gen y0 = sin(2*3.14159*(1-x)^2)
. gen y = y0 + x*invnorm(uniform())
. autosmoo y x, genk(k1) gen(f1)
. graph f1 y0 x , c(ll) s(ii) sort
```

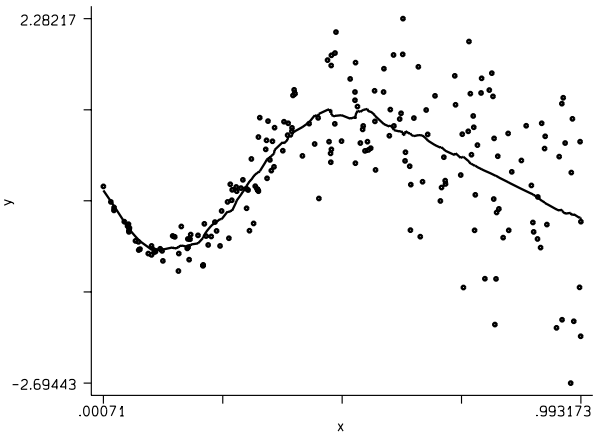

Figure 3. Smoothing the simulated data.

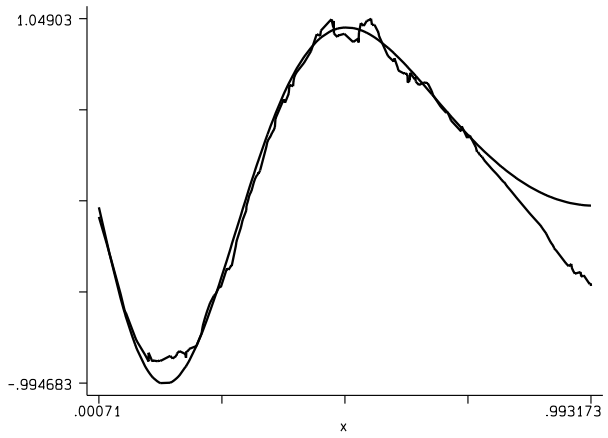

Figure 4. Plotting the smoothed values and the smooth function.

Finally, in Figure 5 we plot both the variable number of symmetric nearest neighbors and the true underlying function (on different scales) against `x`, which is done by

```
. graph k1 y0 x, s(ii) c(ll) sort rescale
```

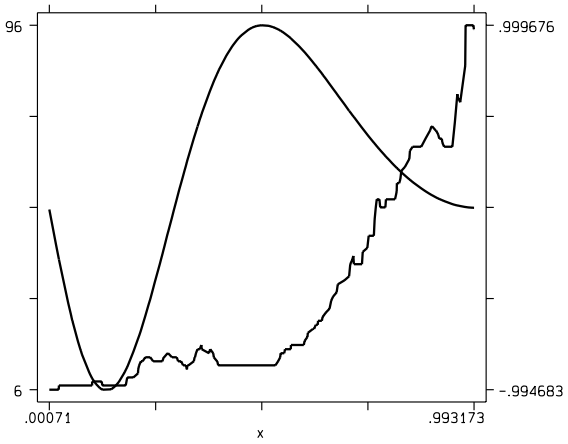

Figure 5. The underlying signal and the number of nearest neighbors used by `autosmoo` plotted against `x`.

Note how the number of neighbors increases from 6 (for small values of `x`) to 96 (for large values of `x`) corresponding to the increasing variance of the "noise" which is proportional to `x`-squared. The theoretical optimal number of nearest neighbors also depends on the curvature of the signal (`y0`). It is also of interest to look at how the selected span varies over simulations. This can be done with the following program:

```
program define simsmoo
        local j 1
        while `j´<=`1´ {
                replace y=y0+x*invnorm(uniform())
                autosmoo y x ,genk(_k`j´) nogr
                local j = `j´ +1
        }
end
```

followed by

```
. egen kmean=rmean(_k*)
. graph kmean x
```

Methods and Formulas

Friedman (1984) proposed a variable span smoother that was computable in order N operations. The program `autosmoo` is similar, but not identical to Friedman's proposal (often called "supersmoother"). The basic idea is to smooth the data using a

few different spans and to compute the cross validated mean squared error (MSE) of prediction at each point for each span. The span with the smallest MSE is identified at each point. These spans are then smoothed and the result is used as the "selected span." The selected span is used to obtain the final smooth. `autosmoo` differs from "supersmoother" in several details.

References

Friedman, J. H. 1984. A variable span smoother. Technical report LCS5, Department of Statistics, Stanford University.

Salgado-Ugarte, I. H., M. Shimizu, and T. Taniuchi. 1996. snp10: Nonparametric regression: Kernel, WARP and k-NN estimators. *Stata Technical Bulletin* 30: 15–30. Reprinted in *Stata Technical Bulletin Reprints* vol. 5, pp. 197–218.

Sasieni, P. 1995. sed9: Symmetric nearest neighbor linear smoothers. *Stata Technical Bulletin* 24: 10–14. Reprinted in *Stata Technical Bulletin Reprints* vol. 4, pp. 97–101.

Silverman, B. W. 1985. Some aspects of the spline smoothing approach to nonparametric regression curve fitting (with discussion). *Journal of the Royal Statistical Society, Series B* 47: 1–52.

ip23 Expansion and display of if expressions

Thomas J. Steichen, RJRT, FAX 910-741-1430, steicht@rjrt.com

Syntax

```
ifexp [if exp] [, novarlabel novallabel nounabbrev nospace
      color({ blue | white | green | yellow | red }) ]
```

Description

`ifexp` expands the tokens in an `if` expression and displays the fully-expanded expression. *exp* contains the expression to be expanded and is required (as is the `if` before it). By default, the following are done:

- variable names are expanded to variable labels (if they exist),
- numerical values are expanded to value labels (if they exist),
- variable names are unabbreviated (if displayed), and
- a space is placed between tokens.

Although `ifexp` is intended primarily for use by ado-file programmers, it can be invoked directly to interactively display an expanded `if` expression.

Options

`novarlabel` suppresses replacement of variable names by variable labels.

`novallabel` suppresses replacement of numerical values by value labels.

`nounabbrev` suppresses unabbreviation of variable names.

`nospace` suppresses insertion of a space between tokens in the output string.

`color()` specifies the display color for the expanded expression. Blue is the default color when this option is not specified or is misspecified.

Remarks

`if` expressions that appear optionally as part of a Stata command can become quite cryptic and uninformative after a very short elapsed time. For example, the following expression appeared in an analysis I did a few months ago: `if expos1==1 & expos8==0`. I have, at best, only a vague recollection of the meaning of the variable names, `expose1` and `expose8`, and I have no idea of the meaning of the numeric values, `1` and `0`. In this instance, though, I had enough foresight to define variable and value labels—thus I could go back and determine the subset of data that the command acted upon. Nonetheless, it would have been much easier if the printout had provided an informative text string such as:

```
if Any Home Exposure? == Yes & Any Work Exposure? == No
```

to go along with the command line expression. Then there would be no question about the subset of data used in that analysis.

`ifexp`, which is implemented in this insert, provides a tool to generate this information. `ifexp` allows ado-file programmers to automatically display the expanded text of an `if` expression. It can also be invoked directly by users who wish to examine or document an expanded `if` expression. Although Stata's style is generally quite terse (ignoring variable and value labels in many instances), display of this output has proven to be surprisingly useful to this author.

Non-Programming examples

Interactive, non-programming use of `ifexp` is straightforward. For the above example, the interactive command and its result would be the following:

```
. ifexp if expos1==1 & expos8==0
  if Any Home Exposure? == Yes & Any Work Exposure? == No
```

If the user prefers to display the string in white with numeric values, the command and results are

```
. ifexp if expos1==1 & expos8==0, noval color(w)
  if Any Home Exposure? == 1 & Any Work Exposure? == 0
```

`ifexp` will correctly process arbitrarily long `if` expressions, wrapping the displayed output at natural breaks between elements, and will process all of the standard delimiters and operators (*except* quotation marks—a limitation of Stata's parser). Here is an example of a long expression which is wrapped when displayed:

```
. ifexp if (expos1==1 | expos8==1) & hfa12 == 1 & hsagei >= 18 & hsagei <= 65
  if ( Any Home Exposure?~== Yes | Any Work Exposure?~== Yes ) &
>    Marital Status == MwSpous & Age at Interview (years) >= 18 &
>    Age at Interview (years) <= 65
```

`ifexp` calls on Stata's parser to determine if a legal `if` expression is present on the command line; therefore the expression must follow all standard rules for `if` expressions.

Programming examples

Programming use of `ifexp` is also straightforward. In programs, the `if` expression is stored in local macro `` `if´ ``. Therefore the programmer need insert only the following line into the program at an appropriate spot (I prefer to put it after any preliminary error-check code and before the start of the procedure's output):

```
if "`if´" != "" {ifexp "`if´"}
```

If options are desired, the options precede the closing bracket:

```
if "`if´" != "" {ifexp "`if´", noval color(w)}
```

Technical note

Local macro `` `if´ `` in the calling program *must* contain the `if` prefix or the programmer must provide a work-around. Without the `if` prefix, `ifexp` will not detect the `if` expression and an error will occur. Therefore, the calling program

1. must not use the `noprefix` descriptor when declaring the `if` local macro (`prefix` is the default), or
2. must manually insert the `if` in the call to `ifexp`.

Below is an example of method 2 (with the required, manually inserted `if` shown underlined):

```
if "`if´" != "" {ifexp if "`if´"}
```

Technical note

The inability of the Stata ado-file parser to process quote marks does not *directly* limit the programming usage of `ifexp`. This seemingly inconsistent assertion is true because the calling program *itself* will fail when attempting to parse an `if` expression that contains quote marks. Since this failure occurs during initialization of the calling program, no subsequent call to `ifexp` will occur; therefore, usage of `ifexp` imposes no new limitations on the calling program.

ip24 Timing portions of a program

Frederic Zimmerman, Stanford University, zimmer@leland.Stanford.edu

Programmers frequently want to know how long portions of a program last. This insert automates the procedure of timing a piece of code or an entire program. `elapse` allows you to name the portion of code you're timing if desired, and also stores the result so that logical operations may be performed on it. `elapse` is accurate over midnight, but not over several days.

Syntax

`elapse` *start_time name_of_operation*

where *start_time* is a previously defined macro equal to the global system macro `S_TIME` at some earlier point, and *name_of_operation* is a string.

`elapse` displays a string with the result. It also creates a global `S_elap`, which is a numerical macro of the form `hhmmss`, where `hh` is the number of hours, `mm` the number of minutes, and `ss` the number of seconds.

Examples

```
local st = "$S_TIME"
...
elapse "`st´" "Bob´s your uncle"
- Bob´s your uncle took 1 minute, 11 seconds.
local oper "Maximum likelihood estimation"
elapse "`st´" "`oper´"
- Maximum likelihood estimation took 7 minutes, 15 seconds.
elapse `st´
- Elapsed time was 1 hour, 10 minutes, 32 seconds.
quietly elapse `st´
if $S_elap > 4500 {
mat xx = startxx
}
```

sbe19	Tests for publication bias in meta-analysis

Thomas J. Steichen, RJRT, FAX 910-741-1430, steicht@rjrt.com

Syntax

The syntax of `metabias` is

`metabias` {*theta* {*se_theta* | *var_theta*} | *exp*(*theta*) *ll ul* [*cl*]} [`if` *exp*] [`in` *range*]
[, `by`(*by_var*) `graph`({`begg` | `egger`}) `level`(#) {`var` | `ci`} *graph_options*]

where the syntax construct $\{a \mid b \mid \ldots\}$ means choose one and only one of $\{a, b, \ldots\}$.

Description

`metabias` performs the Begg and Mazumdar (1994) adjusted rank correlation test for publication bias and performs the Egger et al. (1997) regression asymmetry test for publication bias. As options, it provides a funnel graph of the data or the regression asymmetry plot.

The Begg adjusted rank correlation test is a direct statistical analogue of the visual funnel graph. Note that both the test and the funnel graph have low power for detecting publication bias. The Begg and Mazumdar procedure tests for publication bias by determining if there is a significant correlation between the effect estimates and their variances. `metabias` carries out this test by, first, standardizing the effect estimates to stabilize the variances and, second, performing an adjusted rank correlation test based on Kendall's τ.

The Egger et al. regression asymmetry test and the regression asymmetry plot tend to suggest the presence of publication bias more frequently than the Begg approach. The Egger test detects funnel plot asymmetry by determining whether the intercept deviates significantly from zero in a regression of standardized effect estimates against their precision.

The user provides the effect estimate, *theta*, to `metabias` as a log risk ratio, log odds ratio, or other direct measure of effect. Along with *theta*, the user supplies a measure of *theta*'s variability (i.e., its standard error, *se_theta*, or its variance, *var_theta*). Alternatively, the user may provide the exponentiated form, *exp*(*theta*), (i.e., a risk ratio or odds ratio) and its confidence interval, (*ll*, *ul*).

The funnel graph plots *theta* versus *se_theta*. Guide lines to assist in visualizing the funnel are plotted at the variance-weighted (fixed effects) meta-analytic effect estimate and at pseudo confidence interval limits about that effect estimate (i.e., at $theta \pm z \times se\_theta$, where z is the standard normal variate for the confidence level specified by option `level()`). Asymmetry on the right of the graph (where studies with high standard error are plotted) may give evidence of publication bias.

The regression asymmetry graph plots the standardized effect estimates, *theta*/*se_theta*, versus precision, $1/$*se_theta*, along with the variance-weighted regression line and the confidence interval about the intercept. Failure of this confidence interval to include zero indicates asymmetry in the funnel plot and may give evidence of publication bias. Guide lines at $x = 0$ and $y = 0$ are plotted to assist in visually determining if zero is in the confidence interval.

`metabias` will perform stratified versions of both the Begg and Mazumdar test and the Egger regression asymmetry test when option `by`(*by_var*) is specified. Variable *by_var* indicates the categorical variable that defines the strata. The procedure reports results for each strata and for the stratified tests. The graphs, if selected, plot only the combined unstratified data.

Options

`by(`*by_var*`)` requests that the stratified tests be carried out with strata defined by *by_var*.

`graph(begg)` requests the Begg funnel graph showing the data, the fixed-effects (variance-weighted) meta-analytic effect, and the pseudo confidence interval limits about the meta-analytic effect.

`graph(egger)` requests the Egger regression asymmetry plot showing the standardized effect estimates versus precision, the variance-weighted regression line, and the confidence interval about the intercept.

`level()` sets the confidence level, in percent, for the pseudo confidence intervals; the default is 95%.

`var` indicates that *var_theta* was supplied on the command line instead of *se_theta*. Option `ci` should not be specified when option `var` is specified.

`ci` indicates that *exp*(*theta*) and its confidence interval, (*ll*, *ul*), were supplied on the command line instead of *theta* and *se_theta*. Option `var` should not be specified when option `ci` is specified.

graph_options are those allowed with `graph, twoway`. For `graph(begg)`, the default *graph_options* include `connect(lll.)`, `symbol(iiio)`, and `pen(3552)` for displaying the meta-analytic effect, the pseudo confidence interval limits (two lines), and the data points, respectively. For `graph(egger)`, the default *graph_options* include `connect(.ll)`, `symbol(oid)`, and `pen(233)` for displaying the data points, regression line, and the confidence interval about the intercept, respectively. Setting `t2title(.)` blanks out the default `t2title`.

Input variables

The effect estimates (and a measure of their variability) can be provided to `metabias` in any of three ways:

1. The effect estimate and its corresponding standard error (the default method):

   `. metabias` *theta se_theta* `...`

2. the effect estimate and its corresponding variance (note that option `var` must be specified):

   `. metabias` *theta var_theta*`, var ...`

3. the risk (or odds) ratio and its confidence interval (note that option `ci` must be specified):

   `. metabias` *exp*(*theta*) *ll ul*`, ci ...`

   where *exp*(*theta*) is the risk (or odds) ratio, *ll* is the lower limit and *ul* is the upper limit of the risk ratio's confidence interval.

When input method 3) is used, *cl* is an optional input variable that contains the confidence level of the confidence interval defined by *ll* and *ul*:

`. metabias` *exp*(*theta*) *ll ul cl*`, ci ...`

If *cl* is not provided, `metabias` assumes that each confidence interval is at the 95% confidence level. *cl* allows the user to provide the confidence level, by study, when the confidence intervals are not at the default level or are not all at the same level. Values of *cl* can be provided with or without a decimal point. For example, 90 and .90 are equivalent and may be mixed (e.g., 90, .95, 80, .90 etc.).

Explanation

Meta-analysis has become a popular technique for numerically synthesizing information from published studies. One of the many concerns that must be addressed when performing a meta-analysis is whether selective publication of studies could lead to bias in the meta-analytic conclusions. In particular, if the probability of publication depends on the results of the study—for example, if reporting large or statistically significant findings increase the chance of publication—then the possibility of bias exists.

An initial approach used to assess the likelihood of publication bias was the funnel graph (Light and Pillemer 1984). The funnel graph plotted the outcome measure (effect size) of the component studies against the sample size (a measure of variability). The approach assumed that all studies in the analysis were estimating the same effect, therefore the estimated effects should be distributed about the unknown true effect level and their spread should be proportional to their variances. This suggested that, when plotted, small studies should be widely spread about the average effect and the spread should narrow as sample sizes increase. If the graph suggested a lack of symmetry about the average effect, especially if small, negative studies were absent, then publication bias was assumed to exist.

Evaluation of a funnel graph was a very subjective process, with bias—or lack of bias—being in the eye of the beholder. Begg and Mazumdar (1994) noted this and observed that the presence of publication bias induced a skewness in the plot

and a correlation between the effect sizes and their variances. They proposed that a *formal test for publication bias*, which is implemented in this insert, could be constructed by examining this correlation. The proposed test evaluates the significance of the Kendall's rank correlation between the standardized effect sizes and their variances.

Recently, Egger et al. (1997) proposed an alternative, regression-based test for detecting skewness in the funnel plot and, by extension, for detecting publication bias in the data. This numerical measure of funnel plot asymmetry also constitutes a *formal test for publication bias* and is implemented in this insert. The proposed test evaluates whether the intercept deviates significantly from zero in a regression of standardized effect estimates against their precision. The test is motivated by the observation that, under assumptions of a nonzero underlying effect and a lack of publication bias, 1) small studies would have both a near-zero precision (since precision is predominantly a function of sample size) and a near-zero standardized effect (because of division by a correspondingly *large* standard error), while 2) large studies would have both a large precision and a large standardized effect (because of division by a *small* standard error). Therefore the standardized effects would scatter about a regression line (approximately) through the origin that has a slope which estimates both the size and direction of the underlying effect. Under conditions of publication bias and asymmetry in the funnel plot, the sub-sample of small studies will differ systematically from the sub-sample of larger studies and the regression line will fail to go through the origin. The size of the intercept provides a measure of asymmetry—the larger the deviation from zero the greater the asymmetry. The direction of the intercept provides information on the form of the bias—a positive intercept indicates that the effect estimated from the smaller studies is greater than the effect estimated from the larger studies. Conversely, a negative intercept indicates that the effect estimated from the smaller studies is less than the effect estimated from the larger studies.

Begg's test

This section paraphrases the mathematical development and discussion in the Begg and Mazumdar paper (the paper also includes a detailed examination of the operating characteristics of this test and examples based on medical data).

Let $(t_i, v_i),\ i = 1, \ldots, k$, be the estimated effect sizes and sample variances from k studies in a meta-analysis. To construct the adjusted rank correlation test, calculate the standardized effect sizes

$$t_i^* = \frac{(t_i - \overline{t})}{(v_i^*)^{1/2}}$$

where

$$\overline{t} = \frac{\sum_{j=1}^{k} t_j v_j^{-1}}{\sum_{j=1}^{k} v_j^{-1}}$$

is the variance-weighted average effect size, and

$$v_i^* = v_i - \left(\sum_{j=1}^{k} v_j^{-1} \right)^{-1}$$

is the variance of $t_i - \overline{t}$.

Correlate the standardized effect sizes, t_i^*, with the sample variances, v_i, using Kendall's rank correlation procedure and examine the p value. A significant correlation is interpreted as providing strong evidence of publication bias.

In their examples, Begg and Mazumdar use the normalized Kendall rank correlation test statistic for data that have no ties, $z = (P - Q)/[k(k-1)(2k+5)/18]^{1/2}$, where P is the number of pairs of studies ranked in the same order with respect to t^* and v and Q is the number of pairs ranked in the opposite order. This statistic does not apply a continuity correction. The authors remark that the denominator should be modified if there are tied observations in either t_i^* or v_i but, instead, apparently break ties in their sample data by adding a small constant. The `metabias` procedure implemented in this insert invokes a modification of Stata's `ktau` procedure to calculate the correct statistic, whether ties exist or not, and presents the z and p values with and without the continuity correction.

Begg and Mazumdar report that the principal determinant of the power of this test is the number of component studies in the meta-analysis (as opposed to the sample sizes of the individual studies). Additionally, the power will increase with a wider range in variance (sample size) and with a smaller underlying effect size. The authors state that the test is fairly powerful for a meta-analysis of 75 component studies, only moderately powerful for one of 25 component studies, and weak when there are few component studies. They advise that "the test must be interpreted with caution in small meta-analyses. In particular, [publication] bias cannot be ruled out if the test is not significant."

A *stratified test* can also be constructed. Let $P_l - Q_l$ be the numerator of the unstratified test statistic for the lth subgroup and d_l be the square of the corresponding denominator (i.e., the variance of $P_l - Q_l$). The stratified test statistic, without continuity correction, is defined as

$$z_s = \frac{\sum_l (P_l - Q_l)}{\left(\sum_l d_l\right)^{1/2}}$$

The `metabias` procedure implemented in this insert calculates the correct stratified statistic, whether ties exist or not, and presents the z_s and p_s values with and without the continuity correction.

Begg and Mazumdar assume that the sampling distribution of t is normal, i.e., $t \sim N(\delta, v_i)$, where δ is the common effect size to be estimated and the v_i are the variances, which depend on the sample sizes of the individual component studies. They argue that the normality assumption is reasonable because t is "invariably a summary estimate of some parameter, and as such will possess an asymptotic normal distribution in most circumstances." The subsequent asymptotic-normality assumption for z_s inherently follows from this argument.

Egger's test

This section paraphrases the method development and discussion in the Egger et al. paper. (The paper also provides an empirical evaluation, based on only eight examples from the medical literature, of the ability of the regression asymmetry test to correctly predict whether a meta-analysis of smaller studies will be concordant with the results of a subsequent large trial.)

Let $(t_i, v_i),\ i = 1, \ldots, k$, be the estimated effect sizes and sample variances from k studies in a meta-analysis. Define the standardized effect size as $t_i^* = t_i / {v_i}^{1/2}$, the precision as $s^{-1} = 1/{v_i}^{1/2}$, and the weight as $w_i = 1/v_i$. (In this form of standardization, t^* is a *standard normal deviate* and is designated as such in the Egger paper.) Fit t^* to s^{-1} using standard weighted linear regression with weights w and linear equation: $t^* = \alpha + \beta s^{-1}$. A significant deviation from zero of the estimated intercept, $\widehat{\alpha}$, is interpreted as providing evidence of asymmetry in the funnel plot and of publication bias in the sampled data.

Egger et al. fit both weighted and unweighted regression lines and select the results of the analysis yielding the intercept with the larger deviation from zero. This insert implements only the weighted analysis.

Egger et al. do not provide a formal analysis of coverage (i.e., nominal significance level) or power for this test, though they do provide a number of assertions about power. First, they state that "[i]n contrast to the overall test of heterogeneity, the test for funnel plot asymmetry assesses a specific type of heterogeneity and provides a more powerful test in this situation." Second, they state that "[i]n some situations... power is gained by weighting the analysis." Lastly, in a comparison to the Begg and Mazumdar test, they state that "the linear regression approach may be more powerful than the rank correlation test." Egger et al. note, though, that "any analysis of heterogeneity depends on the number of trials included in a meta-analysis, which is generally small, and this limits the statistical power of the test."

Although the paper provides no formal analysis in support of these assertions, an empirical evaluation based on eight examples from the medical literature is reported. This evaluation assessed the ability of the regression asymmetry test to correctly predict whether a meta-analysis of smaller studies will be concordant with the results of a subsequent large trial. For these eight examples, the test detected bias in 3 of 4 cases where a meta-analysis disagreed with a subsequent large trial and indicated no bias in all 4 cases where the meta-analysis agreed with the subsequent large trial. In contrast, the Begg and Mazumdar test was significant for only 1 of the 4 discordant cases (but like Egger's test, for none of the concordant cases). Nonetheless, eight example cases are too few to be statistically convincing and the test remains unvalidated. Further, the lack of coverage analysis leaves open the question of false-positive claims of asymmetry and publication bias. Interestingly, if the Egger's publication bias test is too liberal (a concern that the author of this insert holds), that translates into conservativeness at the meta-analysis level since the bias test will suggest too frequently that caution is needed in interpreting the results of the meta-analysis.

An approximate *stratified test* can be constructed using logic similar to that of Begg and Mazumdar (although Egger et al. did not do so). Let a_l be the intercept from the regression equation for the lth subgroup and v_l^a be the variance of a_l. The stratified test statistic is defined as

$$z_s = \frac{\sum_l a_l / v_l^a}{\left(\sum_l 1/v_l^a\right)^{1/2}}$$

and is assumed to be distributed asymptotically normal. In this form, the stratified estimate is simply the variance-weighted, fixed effect meta-analysis of the intercepts. This stratified test is implemented in this insert.

Examples

Begg and Mazumdar illustrated their method with examples from the literature. The first example examined the association between Chlamydia trachomatis and oral contraceptive use derived from 29 case-control studies (Cottingham and Hunter 1992). `metabias` is invoked as follows:

```
. metabias logor varlogor, var graph(egger)
```

Option `var` is used because the data were provided as log-odds ratios and variances and this avoids the, admittedly, small step of generating the standard errors. The optional Egger graph is also requested. `metabias` provides the following analysis:

```
Tests for Publication Bias

Begg´s Test
  adj. Kendall´s Score (P-Q) =     85
          Std. Dev. of Score =  53.30 (corrected for ties)
           Number of Studies =     29
                           z  =   1.59
                     Pr > |z| =  0.111
                           z  =   1.58 (continuity corrected)
                     Pr > |z| =  0.115 (continuity corrected)

Egger´s Test
------------------------------------------------------------------------------
 Std_Eff |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   slope |   .5107122   .0266415     19.170   0.000       .4560484     .565376
    bias |   .8016095   .2961195      2.707   0.012       .1940226    1.409196
------------------------------------------------------------------------------
```

The non-continuity-corrected test statistic, $z = 1.59$ ($p = 0.111$), *differs substantially* from that reported by Begg and Mazumdar, $z = 1.76$ ($p = 0.08$). It differs for two reasons: first, the `metabias` procedure corrected the standard deviation of Kendall's score for ties; and second, Begg and Mazumdar apparently carried out their calculation on data that differs slightly from the data they report in their appendix.

The difference in data is apparent when comparing the funnel graph in the published paper to that generated by `metabias`. The published graph suggests that the observation at $(\textit{logor}, \textit{varlogor}) = (0.41, 0.162)$ incorrectly overlays observation $(0.41, 0.083)$; that it, it was incorrectly entered as $(0.41, 0.083)$. Recalculation of the test statistic with ties broken, and with the data modified to match the published graph, yields the published results.

Begg and Mazumdar report that their p of 0.08 is "strongly suggestive of publication bias." Correction of the data and calculation of the test statistic to account for the ties, as shown above, weakens this conclusion. Application of the continuity correction further weakens the conclusion. Nonetheless, with only 29 component studies, the test is expected to have only moderate power at best, and the existence of publication bias cannot be ruled out.

In contrast, the Egger's bias coefficient, *bias* $= 0.802$ ($P > |t| = 0.012$), strongly indicates the presence of asymmetry and publication bias. Further, the sign of the coefficient (positive) suggests that small studies overestimate the effect (or, alternatively, that negative and/or nonsignificant small studies are not included in the Cottingham and Hunter dataset). The slope coefficient, 0.511, which is an estimate of *theta* (that in a weak sense might be considered to be adjusted for the effects of publication bias), is smaller than the effects estimated from meta-analysis of these data using either fixed-effects (*theta* $= 0.655$) or random-effects (*theta* $= 0.716$). These differences in effect estimates are consistent with those expected when small, negative studies are excluded.

The Egger plot (Figure 1), requested via the `graph(egger)` option, graphically shows this test and points out that the analysis is dominated by one large, very precise study. The plot also shows that the data near the origin are systematically elevated.

The Begg funnel graph of the data (Figure 2), which could have been selected with the `graph(begg)` option, provides additional support for this interpretation.

(*Figures on next page.*)

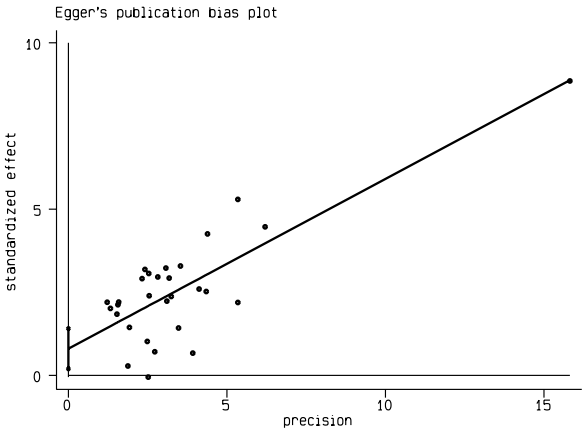

Figure 1

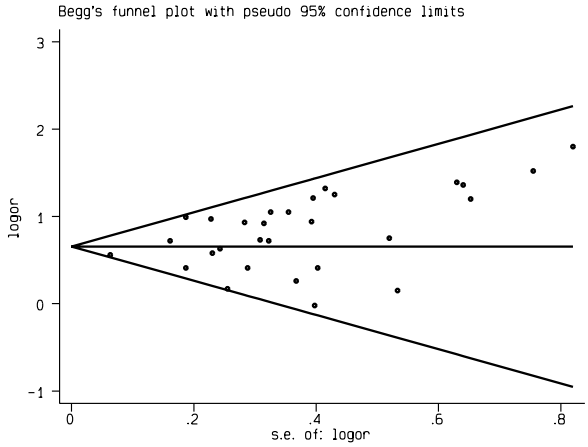

Figure 2

Most of the data points in the Begg plot fall above the meta-analytic effect estimate and there is a visible void in the lower right of the funnel, that is, in the region of low effect and high variance. This is the region where studies most likely to be subject to publication bias would appear. It is notable, though, that since the meta-analytic effect estimate and most of the individual component effect estimates are substantially above zero, the effect of publication bias, if any, would be to inflate the estimate rather than to lead to an incorrect conclusion about the existence of an effect.

Begg and Mazumdar's third example called for the use of the stratified test. These data examined the association between chlorination by-products in drinking water and cancer occurrence, with studies stratified by the site of the cancer (Morris et al. 1992). `metabias` is invoked as follows:

```
. metabias effect variance, var by(site)
```

Use of option `by(site)` informs `metabias` that the stratified tests are to be carried out and that variable `site` is to be used to define the strata. Results are provided in table format, presenting the statistics for each strata and then for the overall stratified tests:

```
Tests for Publication Bias
------------------------------------------------------------------------------
            |       |    Begg´s           Begg´s       cont. corr.   |    Egger´s
       site |     n | score    s.d.       z      p        z      p    | bias     p
------------+-------+-----------------------------------------------+-------------
   Bladder  |     7 |   7     6.658     1.05   0.293    0.90   0.368 |  0.07   0.928
     Brain  |     2 |   1     1.000     1.00   0.317    0.00   1.000 |  4.71       .
    Breast  |     4 |   2     2.944     0.68   0.497    0.34   0.734 |  4.13   0.002
     Colon  |     7 |  -1     6.658    -0.15   0.881    0.00   1.000 |  4.36   0.003
  ColoRect  |     8 |   0     8.083     0.00   1.000   -0.12   1.000 |  5.33   0.273
  Esophagu  |     5 |   4     4.082     0.98   0.327    0.73   0.462 |  1.85   0.456
    Kidney  |     4 |   2     2.944     0.68   0.497    0.34   0.734 |  2.31   0.426
     Liver  |     4 |   2     2.944     0.68   0.497    0.34   0.734 | -0.78   0.727
      Lung  |     5 |   6     4.082     1.47   0.142    1.22   0.221 |  1.06   0.324
  Pancreas  |     6 |   5     5.323     0.94   0.348    0.75   0.452 |  1.55   0.001
    Rectum  |     6 |   1     5.323     0.19   0.851    0.00   1.000 |  4.39   0.103
   Stomach  |     6 |   5     5.323     0.94   0.348    0.75   0.452 |  2.02   0.042
------------+-------+-----------------------------------------------+-------------
   overall  |    64 |  34    17.301     1.97   0.049    1.91   0.056 |  2.51   0.000
------------------------------------------------------------------------------
```

The Begg and Mazumdar results provide no evidence of publication bias in any of the small site-specific strata, yet the stratified test statistic, $z_s = 1.97$ ($p = 0.049$) provides strong evidence that publication bias exists in the chlorinated drinking water and cancer literature. (These results also differ slightly from those published by Begg and Mazumdar in that the published score for the Pancreas strata is 6, leading to an overall score of 35 and slightly different test statistics for this strata and the overall statistic. Results for all other strata agree.) Again, the Egger test provides a stronger indication of the possible presence of publication bias in this literature. Four site-specific strata (Breast, Colon, Pancreas and Stomach) reach statistical significance and the p value for the overall test is more significant than that of Begg's test, 0.000 versus 0.049. All but one of the individual bias coefficients are positive, as is the overall bias coefficient, suggesting that the small studies in this Morris et al. dataset are overestimating the effect (or that the negative and/or nonsignificant small studies are not included).

Saved Results

`metabias` saves

`S_1`	number of studies	`S_5`	Begg's p value, continuity corrected
`S_2`	Begg's score	`S_6`	Egger's bias coefficient
`S_3`	s.d. of Begg's score	`S_7`	Egger's p value
`S_4`	Begg's p value	`S_8`	overall effect (log scale)

References

Begg, C. B. and M. Mazumdar. 1994. Operating characteristics of a rank correlation test for publication bias. *Biometrics* 50: 1088–1101.

Cottingham, J. and D. Hunter. 1992. Chlamydia trachomatis and oral contraceptive use: A quantitative review. *Genitourinary Medicine* 68: 209–216.

Egger, M., G. D. Smith, M. Schneider, and C. Minder. 1997. Bias in meta-analysis detected by a simple, graphical test. *British Medical Journal* 315: 629–634.

Light, R. J. and D. B. Pillemer. 1984. Summing up: The science of reviewing research. Cambridge, MA: Harvard University Press.

Morris, R. D., A. M. Audet, I. F. Angelillo, T. C. Chalmers, and F. Mosteller. 1992. Chlorination, chlorination by-products, and cancer: A meta-analysis. *American Journal of Public Health* 82: 955–963.

sbe20 | Assessing heterogeneity in meta-analysis: the Galbraith plot

Aurelio Tobias, Institut Municipal d'Investigacio Medica (IMIM), Spain, atobias@imim.es

Graphical methods are frequently used in meta-analysis to complement the statistical analysis of clinical and epidemiological data. If the number of studies evaluated in a meta-analysis is small the assessment of heterogeneity is complicated. A range of tests to assess heterogeneity are available (Fleiss 1981), but they tend to have low power against the alternative (Laird and Mosteller 1990). Moreover, it is difficult to have a visual impression of the amount of heterogeneity from common meta-analysis diagrams (Gladen and Rogan 1983, Galbraith 1988). Hence, graphical methods are particularly important to check and to explore potential sources of heterogeneity.

The command `galbr` performs the Galbraith plot (Galbraith 1988), which has been more recommended (Thompson 1993) than other graphical methods to investigate heterogeneity in meta-analysis. This command can be useful to complement the results and graphical displays produced by the `meta` command (Sharp and Sterne 1997).

The Galbraith plot

Following the notation by Sharp and Sterne, let us assume that $\hat{\theta}_i$ is the estimated treatment effect θ_i in a trial i, and v_i the variance of the estimated treatment effect. Then, for each trial i the z statistic $\hat{\theta}_i/\sqrt{v_i}$ is plotted against the reciprocal standard error $1/\sqrt{v_i}$. Different log rate ratios, log odds ratios or log hazard ratios are therefore represented on the diagram by straight lines to the origin for different gradients. In particular, it could be verified that the (unweighted) regression line constrained through the origin has a slope equal to the overall log odds ratio in a fixed effects meta-analysis. Heterogeneity may be assessed by the contribution of each trial i to the overall Q statistic (DerSimonian and Laird 1986) for heterogeneity. This investigation can also be performed visually from a Galbraith plot. The position of each trial on the horizontal axis gives an indication of the weight allocated in the meta-analysis. The vertical axis gives the contribution of each trial to the Q statistic, that is, to say the distance between each trial point and the regression line is equal to q_i^2, where $q_i^2 = w_i(\hat{\theta}_i - \hat{\theta})^2$ and $Q = \sum q_i^2$. Points outside the confidence bounds (positioned 2 units above and below the regression line) are these trials which have a major contribution to heterogeneity. In the absence of heterogeneity we could expect all points within the confidence limits. The Galbraith plot can also be used to investigate possible sources of heterogeneity by labeling the points in the graph by different covariates, for example type of trial, duration of treatment, or drug differences. We should note that this is a post-hoc investigation and interpretation should be made with caution (Thompson 1993).

Syntax

As for the command `meta`, the command `galbr` works on a dataset containing the estimate effect, *theta*, and its standard error, *setheta*, for each trial. The syntax is as follows:

`galbr` *theta setheta* [`if` *exp*] [`in` *range*] [, `id(`*labelvar*`)` *graph_options*]

Options

`id(`*labelvar*`)` supplies a variable which is used to label the studies. If the data contains a labeled numeric variable, it can also be used.

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

`yline(0)` is useful to check the direction and intensity of the overall effect estimated in a fixed effects meta-analysis by the slope of the (unweighted) regression line constrained through the origin.

Although allowed, `ylabel()`, `yscale()`, `xscale()` and, `symbol()` are not suggested.

Example

We illustrate the use of `galbr` with data from seven placebo-controlled studies on the effect of aspirin in preventing death after myocardial infarction. Fleiss (1993) published an overview of these data, and I will focus on the assessment of heterogeneity.

```
. use fleiss
. describe
Contains data from fleiss.dta
  obs:             7
 vars:             7
 size:           231
-------------------------------------------------------------------------------
   1. study       byte   %8.0g                 study identity number
   2. studyid     str8   %8s                   study identity label
   3. rr          float  %8.4f                 odds ratio
   4. logrr       float  %8.4f                 log odds ratio
   5. logse       float  %8.4f                 standard error log odds ratio
   6. wf          float  %8.2f                 fixed effects weights
   7. wr          float  %8.2f                 random effects weights
-------------------------------------------------------------------------------
. list, noobs
study    studyid        rr     logrr     logse        wf        wr
    1      MCR-1    1.3894    0.3289    0.1972     25.71     20.58
    2        CDP    1.4701    0.3853    0.2029     24.29     16.66
    3      MRC-2    1.2451    0.2192    0.1432     48.77     33.11
    4       GASP    1.2497    0.2229    0.2545     15.44     13.43
    5      PARIS    1.2537    0.2261    0.1876     28.41     22.28
    6       AMIS    0.8826   -0.1249    0.0981    103.91     51.77
    7     ISIS-2    1.1176    0.1112    0.0388    664.26     89.28
```

Heterogeneity was tested using the `meta` command, without any statistical evidence of its presence ($Q = 9.968$, $df = 6$, $p = 0.126$). However, this test has been criticized due to its lack of sensitivity to detect heterogeneity (Spector and Thompson 1991). For this reason, Fleiss recommends to use a large significance level α, say 0.10 to 0.20, rather the usual 0.05. The output of the Galbraith plot presents strong visual evidence of heterogeneity between the studies. There is a clear influence of the largest study, ISIS-2, becoming the major weight contributor to the overall fixed-effects estimate. The second largest study, AMIS, becomes the major contributor to the heterogeneity. The other five studies are strikingly homogeneous.

```
. galbr logrr logse, id(studyid)
```

See Figure 1 below.

When the analysis without ISIS-2 and AMIS studies was performed, there was strong evidence of absence of heterogeneity ($Q = 0.627$, $df = 4$, $p = 0.960$), also confirmed by the Galbraith plot. Hence, a random effects model should be more appropriate, reducing the influence of the ISIS-2 study and the amount of heterogeneity produced by the AMIS study.

```
. galbr logrr logse if study!=6 & study!=7, id(studyid)
```

See Figure 2 below.

(*Figures on next page.*)

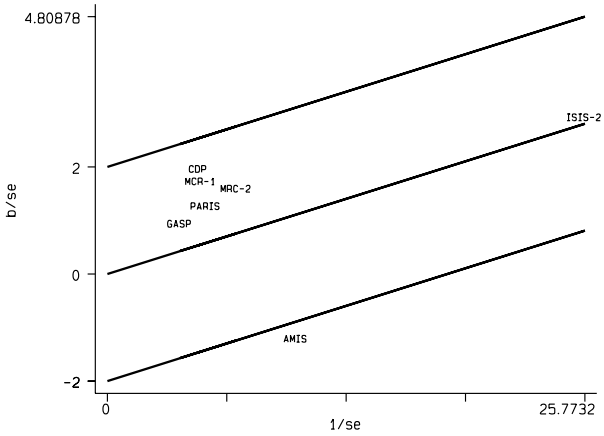

Figure 1. The Galbraith plot for the seven studies evaluated.

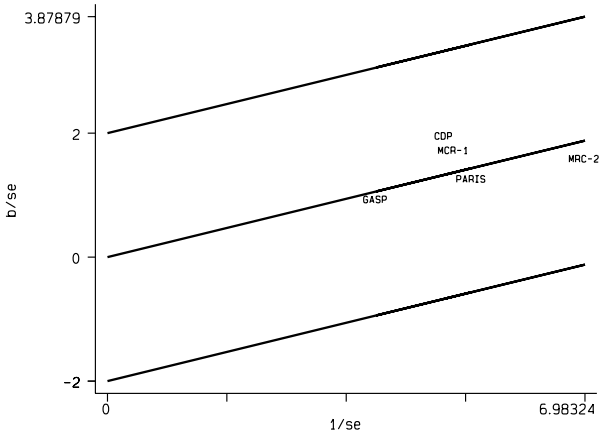

Figure 2. The Galbraith plot, excluding ISIS-2 and AMIS studies.

Individual or frequency records

As in the `meta` command, `galbr` works on data contained in frequency records, one for each study or trial. If we have primary data, that is individual records, we must combine into frequency records using the `collapse` and `byvar` Stata commands.

Acknowledgments

I am grateful to Michael Hills for introducing me to and helping me with Stata programming language, and to Joan B. Soriano for his technical assistance.

References

DerSimonian, R. and N. M. Laird. 1986. Meta-analysis in clinical trials. *Controlled Clinical Trials* 7: 177–188.

Fleiss, J. L. 1981. Statistical Methods for Rates and Proportions, New York: John Wiley & Sons.

——. 1993. The statistical basis of meta-analysis. *Statistical Methods in Medical Research* 2: 121–149.

Galbraith, R. F. 1988. A note on graphical presentation estimated odds ratios from several clinical trials. *Statistics in Medicine* 7: 889–894.

Gladen, B. C. and W. J. Rogan. 1983. On graphing rate ratios. *American Journal of Epidemiology* 118: 905–908.

Laird, N. M. and F. Mollester. 1990. Some statistical methods for combining experimental results. *International Journal of Technology Assessment and Health Care* 6: 5–30.

Sharp, S. and J. Sterne. 1997. sbe16: Meta-analysis. *Stata Technical Bulletin* 38: 9–14.

Spector, T. D. and S. G. Thompson. 1991. The potential uses and limitations of meta-analysis. *Journal of Epidemiology and Community Health* 45: 89–92.

Thompson, S. G. 1993. Controversies in meta-analysis: the case of the trials of serum cholesterol reduction. *Statistical Methods in Medical Research* 2: 173–192.

sed9.1	Pointwise confidence intervals for running

Peter Sasieni, Imperial Cancer Research Fund, UK, p.sasieni@icrf.icnet.uk
Patrick Royston, Imperial College School of Medicine, UK, proyston@rpms.ac.uk

`running` is a symmetric nearest neighbor linear smoother that was introduced in STB-24 (Sasieni 1995). The new version of `running` provides a substantial improvement over the old. The syntax is unchanged except where new options are available. The new version offers the following additional features.

1. **Analytic weights** are allowed. The weights do not affect the "windows" i.e., the observations considered to be "nearest neighbors." Rather it is assumed that the variance of *yvar* is inversely proportional to the weights. Frequency weights would require a new algorithm to select the "neighbors" and are therefore not allowed. Observations with zero, negative or missing weights are treated as missing.
2. **Standard errors** and confidence bands can be calculated and added to the graph for simple smoothers. When the fit is by locally weighted least squares, it is easy to estimate the standard errors of the fitted values. The program uses a local estimate of the variance of an observation with unit weight.
3. **Twicing** is permitted. Thus one may smooth the residuals from the initial fit and add the two smooths to give the final fit. The same smoother must be used for both the raw data and the residuals.

4. **The maximum span** for a running line smoother has been increased from 1 to 2. A span of 2 yields the usual straight line (weighted) least squares fit.

We have also taken the opportunity to correct a few minor bugs and to refine the Stata code. The new program has been extensively tested on small and large datasets with and without weights. Care has been taken so that `running` will work even when there are tied values in *xvar* and between 1 and $N-1$ neighbors on each side of the fitted data point.

Syntax

```
running  yvar [xvar] [weight] [if exp] [in range] [, knn(string) span(#)
         mean repeat(#) double twice ci gen(newvar) gense(sevar)
         genb(bvar) logit nograph graph_options]
```

Description

`running` smooths *yvar* on *xvar*. By default, the smoothed version is a running line: a running mean is also available. A graph of *yvar* together with its smooth is plotted against *xvar*, unless suppressed. If *xvar* is not provided, then *yvar* is smoothed against the ordered observations.

Options

`knn(`*string*`)` specifies the number of nearest neighbors on each side to be used. The argument of `knn` can either be an integer or the name of an integer value variable. `knn` is stored in `$S_1`. The greater the value, the greater the smoothing. You may not specify both `span` and `knn`. The formula for calculating the default value of `knn` is not the same as was used in the previous version of `running`.

`span(#)` specifies the span or proportion of the data to be used in the symmetric nearest neighbors. If `span` is specified `knn` is defined to be $(N * \texttt{span} - 1)/2$, where N is the number of observations. You may not specify both `span` and `knn`. `span` must be be in the range $(0, 2]$. (It must be less than 1 when using `mean`.) Span 2 corresponds to fitting a straight line. `span()` is stored in `$S_2`.

`mean` specifies running-mean least-squares smoothing; default is running-line.

`repeat(#)` specifies the number of times the data is to be smoothed. The default is 1. Increasing `repeat` increases the time taken to calculate the smooth, but should improve the result.

`double` doubles the value of `repeat`. If `repeat` is not specified, `double` is equivalent to `repeat(2)`.

`twice` carries out Tukey's "twicing" procedure whereby residuals from the original fit are smoothed and added back to the fit to obtain the final smooth ("smoothing the rough" or "reroughing" in Tukey's terminology). The result is somewhat rougher than would have been obtained without the application of twicing, but may be a better fit to the data.

`ci` produces a pointwise confidence interval for the smoothed values of *yvar*. The width is determined by the current value of the macro `$S_level`. Not available with `twice`, `repeat`, or `logit`.

`gen(`*newvar*`)` creates *newvar* containing the smoothed values of *yvar*. Note that this will be on a logit scale if `logit` is used.

`gense(`*sevar*`)` creates *sevar* containing the pointwise standard error of smoothed values of *yvar*. Not available with `twice`, `repeat` or `logit`.

`genb(`*bvar*`)` creates *bvar* containing the local slope estimates. They constitute a local estimate of the derivative of the smoothed values of *yvar* with respect to *xvar*. Not available with `mean`, `twice`, or `logit`.

`logit` transforms the smooth and plots the y-axis on a logit scale. 0–1 observations are automatically jittered in the vertical scale and are plotted outside the range of the smoothed curve. It is not necessary that *yvar* is 0–1 provided it takes values in $[0, 1]$.

`nograph` suppresses displaying the graph.

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

Examples

The following examples use Stata's automobile dataset.

```
. running foreign mpg [aw=price], ylab xlab span(2) ci gen(fit1) gense(se1)
. running foreign mpg, ylab(-4,-2,0,2) xlab logit yline(0) r(3)
```

Note that the fitted values and standard errors of prediction with a span of two are the same as those obtained by `fit`.

```
. fit foreign mpg [aw=price]
. predict fit2
. predict se2, stdp
. gen fitd=fit1-fit2
. gen sed=se1-se2
. sum fitd sed
```

Methods and Formulas

The default value of `knn` is defined to be $N^{0.8}/2$ where N is the number of nonmissing observations. The `span` of the smooth is defined as $(2 * \texttt{knn} + 1)/N$. The number of neighbors on each side of the observation used on each pass of the smoother is $\texttt{int}(\texttt{knn}/\sqrt{r} + 0.5)$ where r is the value of `repeat` and `int` is the function yielding the integer part of its argument. Thus the actual span is $r * \texttt{int}\{(N * \texttt{span} - 1)/(2\sqrt{r}) + 0.5\}/N$.

Standard errors are calculated based on the local weighted least squares fit. Confidence intervals take account of sampling error, but not bias. Thus, the smaller the span, the wider will be the confidence interval. The error variance is calculated locally. Thus `running` uses the weighted mean residual sum of squares for the neighbors of the observation. The nominal coverage of the confidence interval is determined by the value of the global macro `$S_level`.

Further examples

Figure 1 illustrates the use of the `logit` option with data that are binomial outcomes with sample sizes of between 1 and 5. The binomial sample sizes are used as weights.

```
. set obs 400
. gen N=min(1+int((invnorm(uniform()))^2),5)
. gen x=uniform()+uniform()
. gen p=1/(1+exp(-3*sin(3*x)-4*(x-1)))
. gen y= uniform()<p
. for 1-4, lty(num) : replace y = y+ (uniform()<p) if N>@
. replace y = y/N
. running y x [aw=N] ,logit gen(lfit)
```

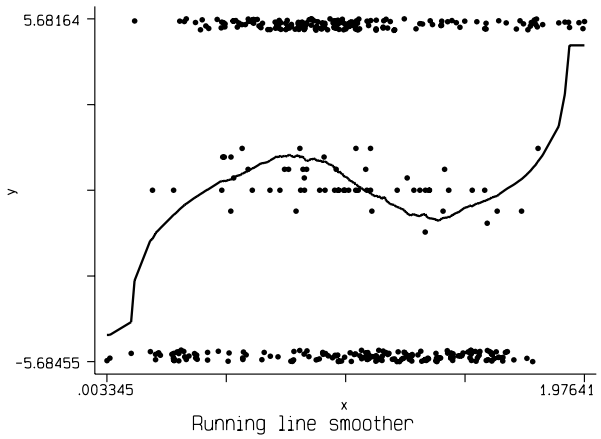

Figure 1. Smoothing binomial outcomes.

The fit is actually much better than it appears in Figure 1 as can be seen in Figure 2. Note that the use of the `jitter` option overides the weighted symbol size of graph.

```
. running y x [aw=N] ,logit jit(2)
. gen lp=log(p/(1-p))
. graph lfit lp x, s(oi) c(.l) sort
```

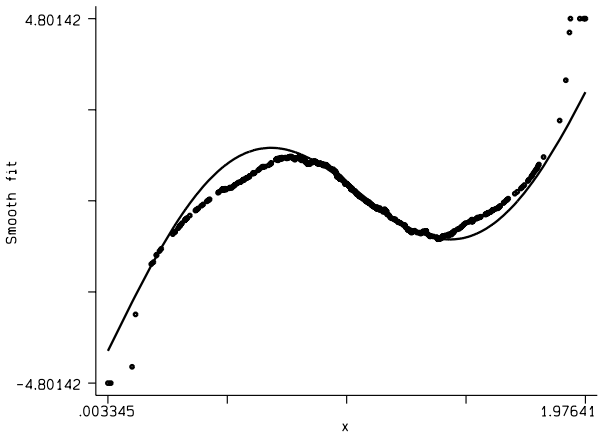

Figure 2. Comparison of the fit to the logit function from which the data were generated.

If the `logit` option is not used, the fitted probabilities are not constrained to be in the interval $[0, 1]$ (Figure 3).

```
. running y x , ylab(0,.5,1) xlab yli(0,1)
```

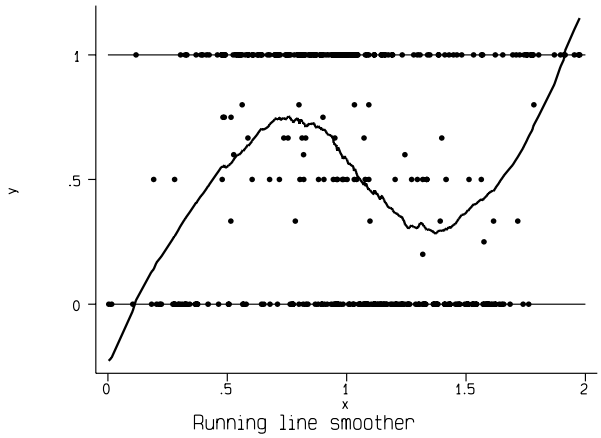

Figure 3. Binomial data smoothed without using the `logit` option.

Figure 4 illustrates how the confidence intervals may be misleading due to bias.

```
. running p x , ci
```

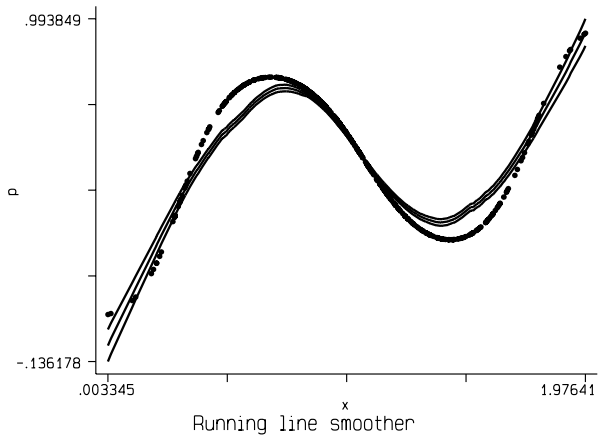

Figure 4. Illustrating possible bias.

In Figure 5 (see below) the effect of twicing is to reduce the bias.

```
. running p x , twi ti"(Running line with twicing")
```

By reducing the span and using `repeat(2)`, we are able to fit the curve almost exactly (Figure 6).

```
. running p x , twi rep(2) knn(12)
```

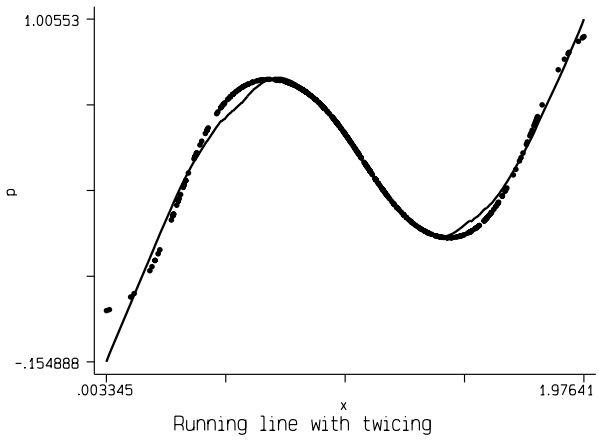

Figure 5. Reducing bias by twicing.

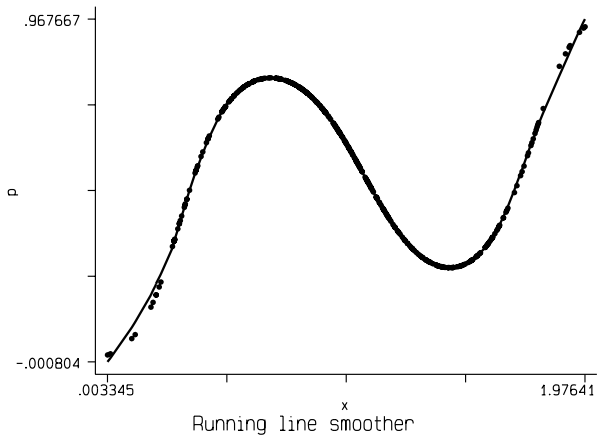

Figure 6. Bias can be greatly reduced by using a smaller span.

Figures 7–10 show the effect of the number of nearest neighbors on the confidence intervals. The default value of `knn` (60.3 in this case) is too big for such noiseless data and the smoother performs poorly, particularly where the signal has a turning point (Figure 7).

```
. gen xg=round(x,.1)
. running p xg, ci ylab t1(default span)
```

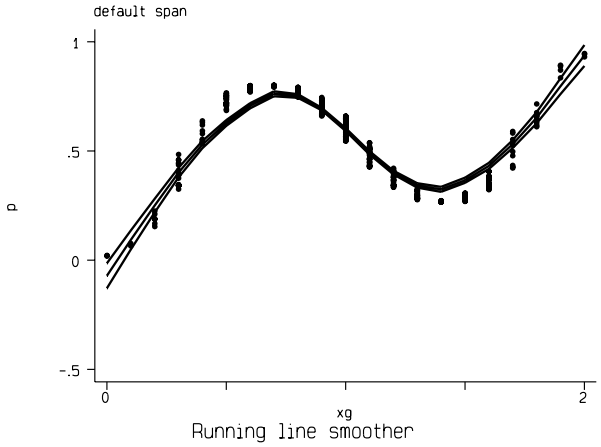

Figure 7. Using the default span.

`running` will not produce confidence intervals when `knn`=1. When `knn` is 2, the fit is excellent to this noiseless data, but the confidence interval is very wide (Figure 8).

```
. running p xg, knn(2) ci ylab t1(knn=2)
```

(*Figure 8 on next page.*)

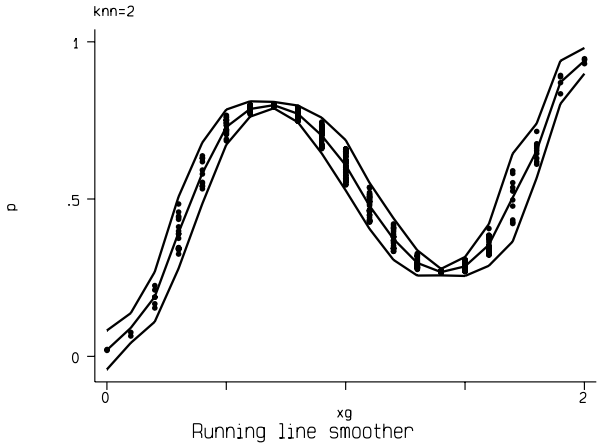

Figure 8. Wide confidence intervals when knn is too small (knn=2).

The situation with knn=3 is already greatly improved (Figure 9).

```
. running p xg, knn(3) ci ylab t1(knn=3)
```

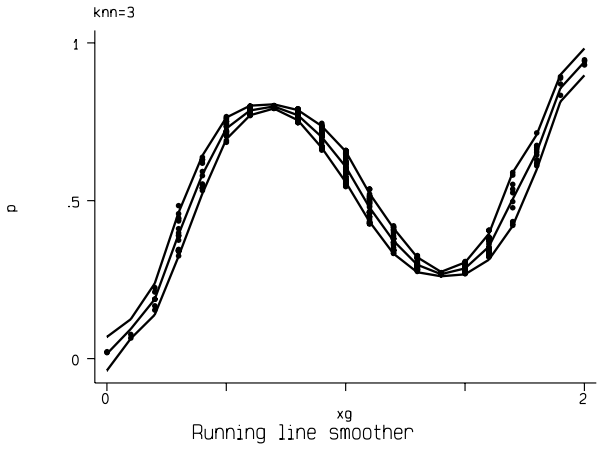

Figure 9. Confidence intervals are still too wide (knn=3)

The choice of knn=20 seems about right for this example (Figure 10).

```
. running p xg, knn(20) ci ylab t1(knn=20)
```

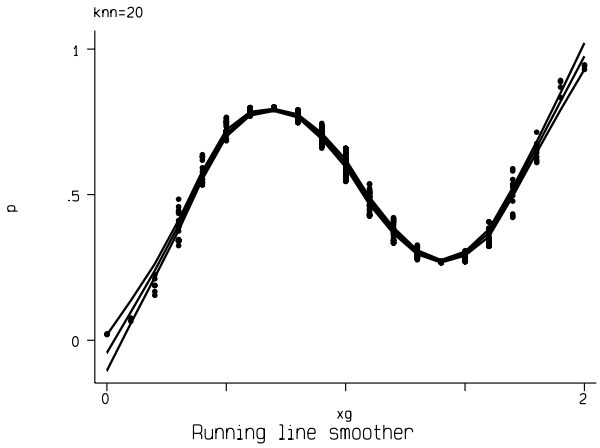

Figure 10. A good choice of knn (20).

References

Sasieni, P. 1995. sed9: Symmetric nearest neighbor linear smoothers. *Stata Technical Bulletin* 24: 10–14. Reprinted in *Stata Technical Bulletin Reprints* vol. 4, pp. 97–101.

Tukey, J. W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison–Wesley.

sg44.1 Correction to random number generators

Joseph Hilbe, Arizona State University, atjmh@asuvm.inre.asu.edu
Walter Linde-Zwirble, Health Outcomes Technologies, walter122@aol.com

It has been called to my attention that the `rndpoi.ado` program that Walter Linde-Zwirble and I wrote for STB-28 does not work from mean values such as 100. I discovered that the largest allowable value for the mean was 83. The reason is the value of the exponentiated negative mean. In order to eliminate the problem one must include a `set type double` into the program. The program has been so amended.

Also, Guy van Melle provided code to reduce computation time by nearly 50%. The problem was in my use of `egen`. Using `count` is much better. I have made the necessary accommodations to the program.

Since `rndpoix.ado`, a program to allow synthetic Poisson modeling, is based on `rndpoi.ado`, I have made necessary changes to it as well. Included are revised `rndpoi.ado` and `rndpoix.ado` programs that incorporate the above enhancements. My thanks, of course, go to Professor van Melle and to another who discovered the mean limitation (I'm embarrassed that I don't recall his name—and apologize).

sg53.2 Stata-like commands for complementary log-log regression

Joseph Hilbe, Arizona State University, atjmh@asuvm.inre.asu.edu

A maximum-likelihood complementary log-log regression program was published in the July 1996 STB-32 (Hilbe 1996). The main program, called `cloglog.ado`, includes a variety of options including a host of residual statistics. Moreover, both ungrouped and grouped data situations can be modeled from within the same program. However, the old cloglog program was written somewhat outside the standard Stata style for ML routines. I have prepared a new version of `cloglog` to more exactly correspond with Stata's internal `logit` and `probit` commands. `cloglog` in general has the same options as `logit` and `probit`, including `robust`, `score`, and `cluster`. In addition, `cloglog` has an ancillary program called `bcloglog`, which models grouped data in a manner similar to that of `blogit` and `bprobit`.

The stepwise command, `sw`, can be used for `cloglog` when StataCorp makes the additional setting from within the `sw` command. It is an easy task, but I chose not to change StataCorp's own program. Perhaps it will be amended when this program is published.

Reference

Hilbe, J. 1996. sg53: Maximum-likelihood complementary log-log regression. *Stata Technical Bulletin* 32: 19–20.

sg75 Geometric means and confidence intervals

John Carlin, University of Melbourne and Royal Children's Hospital, Australia, j.carlin@medicine.unimelb.edu.au
Suzanna Vidmar, University of Melbourne and Royal Children's Hospital, Australia
Carlos Ramalheira, Coimbra University Hospital, Portugal, cramal@cygnus.ci.uc.pt

The geometric mean is a natural summary statistic for a log-normal distribution, since it is the back-transform or anti-log of the mean of the log-transformed values, which (by definition) have a symmetric normal distribution. More generally, many right-skewed distributions are approximately log-normal and are better summarized by the geometric mean than by the mean, since estimates of the latter may be strongly influenced by a small number of outlying values. Several textbooks point out that the geometric mean is usually close to the median in distributions with this sort of skewness (e.g. Altman 1991, Armitage and Berry 1994). A helpful review of the important role of log transformations in analyzing biological data is provided by Healy (1993).

Stata's `ci` command provides confidence intervals for estimated means and proportions and the `means` command extends `summarize` to give geometric and harmonic means, but it does not provide any inferential statistics. The new command offered here, `gmci`, provides geometric means with confidence intervals in a very similar format to `ci`. The command is an improved version of a couple of simple prototypes offered on Statalist by William Gould and Nick Cox.

Syntax

gmci *varlist* [if *exp*] [in *range*] [, level(#) add(#) only]

Options

level(#) sets the desired confidence interval level to #. When specified, # must be an integer from 10 to 99. The default is to use the value in the global macro S_level (set at 95 when Stata starts up).

add(#) adds the value # to all values of *varlist* before the geometric mean and confidence interval are calculated. This is sometimes useful when analyzing variables with nonpositive values.

only has no effect unless the add option is used. When both options are specified, the add option will only be applied to variables that have nonpositive values.

Example

The dataset eg_gmci.dta contains the titres of 3 different antibodies from 188 babies after a primary course of immunization. Various graphical inspections (in particular qnorm) reveal that all three antibody levels have strongly skewed distributions that are made approximately symmetric, if not exactly normal, by log transformation. The geometric mean and confidence interval for ab1 may be obtained simply:

```
. gmci ab1
  Variable  |    Obs     Geometric Mean           [95% Conf. Interval]
------------+------------------------------------------------------------
       ab1  |    188          14.57392          12.73919      16.67289
```

The same method fails for ab2 because there are two zero values:

```
. gmci ab*
Nonpositive values encountered in variable ab2.
Minimum value of ab2 is  0.0, minimum positive value of ab2 is  1.9
r(411);
```

This may be overcome with the add option:

```
. gmci ab*, add(1)
  Variable  |    Obs     Geometric Mean           [95% Conf. Interval]
------------+------------------------------------------------------------
       ab1  |    188          16.01538          14.14699      18.13052 *
       ab2  |    188          46.62022          38.43774      56.54456 *
       ab3  |    188          16.15642          13.36500      19.53086 *
(*) 1 was added to the variable(s) prior to calculating the results
```

Finally, the only option may be used to restrict the adding of 1 to the variables that do not have all positive values:

```
. gmci ab*, a(1) o
  Variable  |    Obs     Geometric Mean           [95% Conf. Interval]
------------+------------------------------------------------------------
       ab1  |    188          14.57392          12.73919      16.67289
       ab2  |    188          46.62022          38.43774      56.54456 *
       ab3  |    188          13.54571          10.84699      16.91586
(*) 1 was added to the variable(s) prior to calculating the results
```

Remark

Although the device of adding a constant to all values in order to remove zeros or even negative values, before taking logs, appears to be widely used, it should be approached with caution. In particular, it is often recommended to add 1 (which means that the minimum log value becomes 0, if the raw data have minimum 0), but this is appropriate only if the original scale of measurement is such that most values are greater than 1. It is not appropriate to add a "very small" value (such as 0.1, 0.001), because this is likely to create outliers relative to the variation in the remaining values, since the definition of "very small" is always relative (it is a long way from any positive value to 0 on a log scale since the log of 0 can be regarded as negative infinity). Minimal distortion seems to result if a value of about one half of the minimum positive value in the data is added. Even then, however, the interpretation may be somewhat awkward, since the resulting geometric mean is a summary of the shifted distribution, not the original. Finally, if there are a large number of zeros, no transformation will achieve a symmetric distribution, so the use of a geometric mean as a summary may not be very helpful.

References

Altman, D. G. 1991. *Practical Statistics for Medical Research*. London: Chapman & Hall.

Armitage, P. and G. Berry. 1994. *Statistical Methods in Medical Research*. 3d ed. Oxford: Blackwell Scientific Publications.

Healy, M. J. 1993. Data transformations. *Archives of Diseases in Childhood* 69: 260–264.

ssa10.1	Update to analysis of follow–up studies with Stata 5.0

David Clayton, MRC Biostatistical Research Unit, Cambridge, david.clayton@mrc-bsu.cam.ac.uk
Michael Hills, London School of Hygiene and Tropical Medicine (retired), mhills@regress.demon.co.uk

All routines in ssa10 have been updated to version 2.0 in which a number of small bugs have been fixed.

Reference

Clayton, David and Michael Hills. 1997. ssa10: Analysis of follow–up studies with Stata 5.0. *Stata Technical Bulletin* 40: 27–39.

ssa11	Survival analysis with time-varying covariates

Jeroen Weesie, Utrecht University, Netherlands, weesie@weesie.fsw.ruu.nl

One of the most demanding tasks in the analysis of survival time data involves data management in constructing time-varying covariates. In an undergraduate course on "event-history modeling" for the social sciences that I teach using Stata, a large fraction of the time and inspiration of students evaporates in perspiration over trying to understand a relatively long series of arcane, intimidating Stata statements in `expand`, `by`, and the system identifiers `_n` and `_N`. In a desperate attempt to become more popular with my students, and, more seriously, to redirect the student's attention to the more substantial and statistical issues (and, of course, to reduce data manipulation errors in my own research), I wrote a series of new commands (programs) that I hope will facilitate survival analysis with time-varying covariates. With these new programs, performing survival analyses with time-varying covariates will resemble much more closely how one uses other software for survival time analyses. These new programs employ the facilities of the new `st` package for survival time data that was introduced in Stata 5.0. In this insert I describe my collection of new `st` programs, with a series of examples. My description presupposes that the reader is already familiar with the `st` package and is familiar with survival time modeling. In particular, if you haven't read the section of the Reference Manual on `stset`, you should probably read it before reading on here.

Let me start with some more or less formal descriptions of the five new programs. After these descriptions, we illustrate the commands with numerous examples. All these commands are for use with survival time data. Thus, you must have `stset` your data before using the commands.

Time-varying covariates: Discrete transitions

We first discuss 4 new commands that are especially useful for a more or less "declarative" way to construct transition-type time-varying covariates.

```
stegen [type] newvar [if exp] [in range], at(exp-t) [from(exp-0) to(exp-1)
       censor noshow preserve]
strepl oldvar [if exp] [in range], at(exp-t) { from(exp-0) | to(exp-1) }
       [censor noshow preserve]
stsplit[type] newvar [if exp] [in range], { at(numlist | exp-list) | every(# | exp) }
       [expr noshow preserve]
sttvc  [type] newvar = v(0) t(1) v(1) ... t(k) v(k) [if exp] [in range]
       [, zero(exp-t) noshow preserve]
```

Description

`stegen` generates a new time-varying covariate that changes `at` time *exp-t* `from` value *exp-0* `to` value *exp-1*. `stegen` performs episode-spitting if necessary.

`strepl` modifies an existing time-varying covariate so that is takes value *exp-0* for $t>$*exp-t* or *exp-1* for $t\leq$*exp-t*. `strepl` performs episode-spitting if necessary.

`stsplit` splits the time axis at time points $n_1, n_2, \ldots, n_k$ specified by a *numlist* into $k+1$ intervals $[\,0, n_1\,], (\,n_1, n_2\,], \ldots$, creating an indicator variable that has value i on the ith interval. `stsplit` can be used to create a "person-period file."

`sttvc` generates a time-varying covariate that is $v(0)$ for $t \leq t(1)$, is $v(1)$ for $t(1) < t \leq t(2), \ldots$, is $v(k-1)$ for $t(k-1) < t \leq t(k)$, and is $v(k)$ for $t > t(k)$. The list $v(0)\ t(1) \ldots t(k)\ v(k)$ consists of an odd number of expressions, separated by blanks. Moreover, it is enforced that the transition times $t(j)$ are strictly increasing, or that $t(j) ==.$ implies that $t(j') ==.$ for $j' > j$.

These commands can of course be invoked more than once in a job, in any combination, to create and manipulate one or more time-varying covariates. Episode splitting will always be performed transparently. If necessary, these commands generate a case identification variable, entry times, and a failure/censor indicator. See help for `stset` for details on these variables; and help for `st_aux` for details on the names.

Options for stegen and strepl

`at(`*exp-t*`)` is not optional; it specifies the time, expressed as the elapsed time since time 0, at which the TVC changes from *exp-0* to *exp-1*. The time should be nonmissing.

`from(`*exp-0*`)` specifies the value of the TVC before the change (transition). In `stegen`, `from` defaults to 0. In `strepl` one should specify either `from()` or `to()`.

`to(`*exp-1*`)` specifies the value of the TVC after the change (transition). In `stegen`, `to` defaults to 1. In `strepl` one should specify either `from()` or `to()`.

`censor` specifies that missing values of the at-expression indicate censoring, e.g., the event did not happen during $(\,t(0), t(1)\,)$, and "thus" the TVC should be set to the from-value (`stegen`) or left unchanged (`strepl`).

Options for stsplit

`at(`*numlist* | *exp-list*`)` specifies the time points, expressed as the elapsed time since time 0, at which the episodes have to be split. See `help numlist`. For example, `at(5 20)` performs episode splitting at times 5 and 20. Either `at()` or `every()` must be specified.

`every(`*#* | *exp*`)` specifies that episodes are generated at multiples of a (positive) constant *#* or of an expression that may vary between cases.

`expr` specifies that `at()` should be interpreted as an expression-list rather than as a numeric list.

Options for sttvc

`zero(`*exp*`)` specifies the zero-point for the transition times $t(j)$. The expression should evaluate to nonmissing.

Time-varying covariates in Cox regression

To facilitate analyses with the Cox regression model with covariates that vary "continuously" in time, we include the command `stcoxtvc`:

`stcoxtvc` *varlist* [`if` *exp*] [`in` *range*] [, `eps(`*real 1E-6*`)` `strata(`*varnames*`)` `noshow` `preserve`]

Description

`stcoxtvc` creates the risk-pool expanded data in st-format, so that time-varying covariates can be generated in terms of the survival time variable. These time-varying covariates may change at discrete time or "continuously." Note that `stcoxtvc` is suitable only for semi-parametric Cox-regression, not for the fully parametric models `stereg` and `stweib`.

`stcoxtvc` may require huge amounts of memory. Invoking `stcoxtvc` with the list of variables that you will use as time-constant covariates, or in the construction of time-varying covariates helps to reduce memory requirements. If you want to invoke `stcox` only on a selection of cases (with `if` or `in` phrases) specify these phrases already here. Similarly, if you want to use stratification with `stcox`, `stcoxtvc` will require less memory if these variables are known already here.

Options

`eps(#)` specifies a typically small number so that a case that is in the risk set at time t is represented in the expanded data by an "infinitesimal" episode $(\,t - \texttt{eps}, t\,]$. Infinitesimal episodes rather than n death-on-arrival-episodes are required in Stata because of the particular order in which Stata orders failures, censors, and entries that occur "at" some time t. `eps` should be set to a number that is small compared to the measurement unit of time. `eps` defaults to $1E-6$.

`strata(`*varnames*`)` specifies up to five strata variables. In `stcox`, observations with equal values of the variables are assumed to be in the same stratum. Stratified estimates (equal coefficients across strata but baseline hazard unique to each stratum)

are then estimated. `strata` are used in risk-pool expansion so that records are replicated only at all times at which failures occur within the stratum, and so memory requirements are reduced when strata are specified. Performing a stratified analysis with `stcox` on `stcoxtvc` expanded data that do not account for the strata wastes memory but produces otherwise correct results.

General options

All five commands discussed above support the following options.

`show` and `noshow` change whether the other `st` commands are to display the identities of the key `st` variables at the top of their output.

`preserve` specifies that the data are preserved before data manipulation, so that the data will be restored in original form after pressing the break key.

These programs expand and may substantially modify the data in memory. As a consequence, you can't undo the modifications to the data. Before using the programs you should thus save your data. Beware that the programs do not obey the Stata convention that destructive programs should be invoked on modified data with the option `clear` to protect a user from (unintentional) destruction of data. Survival analysis with time-covariates is not for the faint-hearted anyway.

These programs support multi-record and recurrent event data, data with late entry, with gaps, etc. The programs automatically manipulate the entry times (`t0`), exit times (`t`), failure/censor indicator (`died`), and the case identifier (`id`). If these variables have not already been set by `stset`, the programs will generate these variables automatically with the following names and values. If a case identifier was not specified, one is created with values `_n`, and named `id, caseid,...` whichever did not yet occur in the data. If an entry time variable was not specified, one is created with value 0, meaning no late entry, and named `t0`, `etime`, or `st_t0`, whichever was available as the name for a new variable. Finally, if a censor variable was not specified, one named `died`, `failure`, or `st_d` is created with value 1, meaning that all cases died.

Example 1

The exponential regression model (without time-covariates) assumes that the hazard rate h is constant in time t,

$$\log h(t, x_i) = \beta_0 + x'_{\{1i\)}\beta_1 + \ldots + x'_{\{1k\)}\beta_k$$

where the β are the regression coefficients to be estimated. Note that the exponential model fits into the class of proportional hazards models with constant baseline hazard β_0. Here it is assumed that the baseline hazard is *piecewise constant* on a predefined set of time intervals. For ease of presentation we consider the case of two time intervals $[\,0, T\,)$, and $[\,T, \infty\,)$. Then the model reads

$$\log h(t, x_i) = \begin{cases} \alpha_0 + x'_{\{1i\)}\beta_1 + \ldots + x'_{\{1k\)}\beta_k, & \text{for } t < T \\ \alpha_1 + x'_{\{1i\)}\beta_1 + \ldots + x'_{\{1k\)}\beta_k, & \text{for } t \geq T \end{cases}$$

This model can be estimated in Stata in a somewhat different parameterization (deviation contrast) by conceiving of the model as one with k time-constant covariates, and one time-varying dummy covariate. This involves some data manipulation. Consider a simple example with three cases, no late entry, and no recurrent data:

```
t       died    X
10       0      x1
8        1      x2
4        1      x3
```

With episode splitting at time $T = 5$, this has to be turned into a new data structure with a case identifier (`id`), an entry-time variable (`t0`), and a time-varying covariate `Late`:

```
id      t0      t       died    Late    X
1       0       5       0       0       x1
1       5       10      0       1       x1
2       0       5       0       0       x2
2       5       8       1       1       x2
3       0       4       1       0       x3
```

This can be achieved with the following commands:

```
. gen id = _n                          (create case identifier)
. expand if t > 5                      (episode splitting if survival time longer than 5)
. sort id                              (by requires sort)
. by id: replace died = 0 if _n<_N     (newly created first episodes are censored)
. by id: replace t  = 5 if _n<_N       (newly created first episodes end at time 5)
. by id: gen     t0 = 0 if _n<_N       (entry in first episodes at time 0)
. by id: replace t0 = 5 if _N==2       (entry in second episode at time 5)
. by id: gen     Late = _N==2          (time-varying dummy)
```

While the logic of these commands is fairly simple, it is a lot of work, it requires relatively intricate understanding of Stata syntax, and it is easy to make mistakes. With the new command `stegen` it becomes very simple to manipulate data in constructing a time-varying dummy.

```
. stegen Late, at(5)
```

`stegen` has created `id` and `t0`, has performed episode splitting for the required cases, and has adopted `t0`, `t`, and `died` appropriately. We illustrate the command on a subset of the cancer data distributed with Stata.

```
. use cancer
(Patient Survival in Drug Trial)
. gen S = _N
. recode S 1 14 25 45 46 = 1 * = 0
. list stu died drug age if S, nodisplay
     studytim  died  drug   age
  1.        1     1     1    61
 14.       11     1     1    55
 25.       10     0     2    49
 45.       33     1     3    60
 46.       34     0     3    62
```

`stset` provides the setup for the `st` package.

```
. stset studytim died,
   data set name:  cancer.dta
              id:  --                    (meaning each record a unique subject)
      entry time:  --                    (meaning all entered at time 0)
       exit time:  studytim
  failure/censor:  died
```

For a comparison below, we estimate the standard exponential regression model.

```
. stereg drug age, nolog
    failure time:  studytim
  failure/censor:  died
Exponential regression -- entry time 0
log relative hazard form
No. of subjects =           48                  Log likelihood =  -48.837598
No. of failures =           31                  chi2(2)        =       25.01
Time at risk    =          744                  Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    drug |   .3625496   .0883076     -4.165   0.000       .2249247    .5843833
     age |   1.081641   .0347794      2.441   0.015       1.015578    1.152001
------------------------------------------------------------------------------
```

We will now estimate a piecewise constant exponential regression model with episode splitting at the median survival time. This can be estimated nonparametrically with the standard `st` command `stsum`.

```
. stsum
    failure time:  studytim
  failure/censor:  died
         |                incidence       no. of    |------ Survival time -----|
         | time at risk     rate        subjects        25%       50%       75%
---------+--------------------------------------------------------------------
   total |          744   .0416667            48          8        17        33
```

Thus, the median survival time is 17. As before, we create a time-varying dummy `Late`.

```
. stegen Late, at(17)
number of episode splits : 18
st key variables were created
   data set name:  d:cancer.dta
              id:  id           defined to _n
      entry time:  t0           defined to 0, meaning all entered at time 0
       exit time:  studytim
  failure/censor:  died
```

`stegen` reported that in 18 cases episodes had to be split. In 18 cases survival time exceeded 17. In addition, `stegen` reports that new key variables `t0` and `id` were created. Let us look in more detail at the selected "interesting cases."

```
. list id t0 stu died drug age Late if S, nodisplay noobs
    id    t0  studytim  died  drug   age  Late
     2     0         1     1     1    61     0
    19     0        10     0     2    49     0
    22     0        11     1     1    55     0
    45     0        17     0     3    60     0
    45    17        33     1     3    60     1
    46     0        17     0     3    62     0
    46    17        34     0     3    62     1
```

We see that episodes were split for cases 45 and 46; they survived after time 17.

For comparison, we can estimate an exponential regression model on the expanded data, but without the time-varying covariate `Late`.

```
. stereg drug age, nolog
    failure time:  studytim
      entry time:  t0
  failure/censor:  died
              id:  id
Exponential regression -- entry time t0
log relative hazard form
No. of subjects =           48                  Log likelihood =  -58.518083
No. of failures =           31                  chi2(2)        =       25.01
Time at risk    =          744                  Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    drug |   .3625496   .0883076     -4.165   0.000       .2249247    .5843833
     age |   1.081641   .0347794      2.441   0.015       1.015578    1.152001
------------------------------------------------------------------------------
```

We see that the `stereg` output for the unexpanded and expanded data are identical. This should be, since the expanded data organization is just a less economical way for describing the same information. The careful reader may have noticed that the "Log likelihood" values are not the same. This is due to the fact that, as explained in *Stata Reference Manual*, vol. 3 (p. 355), Stata does not display the true log likelihood.

We can now estimate the piecewise constant exponential regression model.

```
. stereg drug age Late, nolog noshow
Exponential regression -- entry time t0
log relative hazard form
No. of subjects =           48                  Log likelihood =  -55.983059
No. of failures =           31                  chi2(3)        =       30.08
Time at risk    =          744                  Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    drug |    .295467   .0767466     -4.694   0.000       .1775874    .4915932
     age |   1.095681   .0372252      2.690   0.007       1.025097    1.171125
    Late |   2.994553   1.358982      2.417   0.016       1.230387    7.288238
------------------------------------------------------------------------------
```

We see that `Late` is very significant: the death after time 17 is about three times higher than before time 17. Thus, we can reject the exponential regression model that assumes time-constancy.

To me, a piecewise constant model is not very attractive because its baseline function is so discrete. Does it make sense to assume that death rate just before 17 may be totally different than right after 17? Thus, we may, in this case, actually prefer a Weibull model to test the time constancy assumption. However, the Weibull model

$$\log h(t, x(t)) = b \log t + LP$$

assumes that the baseline hazard is monotone, i.e., increasing or decreasing. In many situations it may be more reasonable to assume that survival is inverse-U shaped. This cannot be tested with the Weibull specification. (We may of course inspect the Kaplan–Meier estimate of the survival rates, adjusted for the covariates, for a visual test.) Currently, Stata does not estimate survival models with non-monotone hazard rates such as the log-normal and log-logistic models. However, with the piecewise constant model with more than 2 episodes, a fairly crude test can be made.

Continuing the example with the cancer data, we note that the 25th percentile of survival is at time 8. We construct a further time-varying dummy `Early` that is 1 at times $[0, 8)$, and 0 otherwise. The default behavior of `stegen` is to construct time-varying dummies that are 1 *after* some time point. Using the options `from()` and `to()` this can be modified.

```
. stegen Early, at(8) from(1) to(0)
number of episode splits : 32
```

We see that in 32 cases, survival lasted at least till time 8. Let us look at the "interesting special cases."

```
. list id t0 stu died drug age Late Early if S, nodisplay noobs
   id    t0  studytim  died  drug   age  Late  Early
    2     0         1     1     1    61     0     1
   19     0         8     0     2    49     0     1
   19     8        10     0     2    49     0     0
   22     0         8     0     1    55     0     1
   22     8        11     1     1    55     0     0
   45     0         8     0     3    60     0     1
   45     8        17     0     3    60     0     0
   45    17        33     1     3    60     1     0
   46     0         8     0     3    62     0     1
   46     8        17     0     3    62     0     0
   46    17        34     0     3    62     1     0
```

We can now estimate a piecewise constant exponential regression model, with two time-varying dummies, `Early` (time $\leq$ 8) and `Late` (time$>$17). Thus, 8$<$time$\leq$17 is the reference category.

```
. stereg drug age Early Late , nolog noshow
Exponential regression -- entry time t0
log relative hazard form
No. of subjects =           48                  Log likelihood =  -64.920965
No. of failures =           31                  chi2(4)        =       31.11
Time at risk    =          744                  Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    drug |   .2832666   .0742996     -4.809   0.000        .169406    .4736545
     age |    1.10172   .0380458      2.805   0.005       1.029619    1.178869
   Early |   .6365223   .2784854     -1.033   0.302       .2700248    1.500457
    Late |    2.31135   1.165007      1.662   0.096       .8606529    6.207308
------------------------------------------------------------------------------
```

We see that the multiplicative effect on the hazard rate of `Early` is below 1 and of `Late` larger than 1. Thus, the three-period discretization of the baseline hazard supports the claim that death rates increase with survival time.

With many time points it becomes laborious and error prone to issue many `stegen` commands to construct the time-varying dummies. The command `stsplit` provides a simpler and faster alternative.

```
. use cancer, clear
(Patient Survival in Drug Trial)
. stset studytim died, noshow
```

`stsplit` takes as an argument `at()`, a numeric list of values at which episodes have to be split. Thus `at(8 17)` invokes episode splitting at 8 and 17. `stsplit` created a time-varying variable that identifies the respective episodes. In the example, `tvc` takes value 1 on the episode $[0, 8)$, 2 on the interval $[8, 17)$, and 3 on $[17, .)$.

Actually, `stsplit` can also be called with the option `every(#)` for episode splitting at regular intervals. Below, we will discuss how this can be used for person-period analyses.

```
. stsplit Epi, at(8 17)
number of episodes generated : 50
. tab Epi, gen(e)
       Epi |      Freq.     Percent        Cum.
-----------+-----------------------------------
         1 |         48       48.98       48.98
         2 |         32       32.65       81.63
         3 |         18       18.37      100.00
-----------+-----------------------------------
     Total |         98      100.00
```

We conclude that there are 48 patients who survived after time 8, and 32 who survived after time 3. We used the `gen()` option of the `tab` command to generate time-varying dummies. (Of course, we could also have used the prefix command `xi:`). Again, we inspect our "interesting cases."

```
. list id t0 stu died drug age Epi e1 e2 e3 if S, nodisplay noobs
   id    t0  studytim  died  drug   age   Epi    e1    e2    e3
    1     0         1     1     1    61     1     1     0     0
   14     0         8     0     1    55     1     1     0     0
   14     8        11     1     1    55     2     0     1     0
   25     0         8     0     2    49     1     1     0     0
   25     8        10     0     2    49     2     0     1     0
   45     0         8     0     3    60     1     1     0     0
   45     8        17     0     3    60     2     0     1     0
   45    17        33     1     3    60     3     0     0     1
   46     0         8     0     3    62     1     1     0     0
   46     8        17     0     3    62     2     0     1     0
   46    17        34     0     3    62     3     0     0     1
```

Finally, we estimate an exponential regression model with a piecewise-constant baseline model.

```
. stereg drug age e2 e3, nolog noshow
Exponential regression -- entry time t0
log relative hazard form
No. of subjects =           48                   Log likelihood =  -64.920965
No. of failures =           31                   chi2(4)        =       31.11
Time at risk    =          744                   Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    drug |   .2832666   .0742996     -4.809   0.000        .169406    .4736545
     age |    1.10172   .0380458      2.805   0.005       1.029619    1.178869
      e2 |   1.571037   .6873457      1.033   0.302       .6664637    3.703363
      e3 |   3.631216   1.801366      2.600   0.009       1.373377    9.600955
------------------------------------------------------------------------------
```

Example 2

We now illustrate the commands for the construction of time-varying covariates in an example in which the transition times for a covariate varies between subjects. The example is based on the Coleman–Hoffer research project "High school and Beyond;" the data are copied from Yamaguchi (1991). We want to test whether marital status affects school drop outs. These are the variables:

`caseid`	Observation id
`time`	Time till school drop-out (number of months since entry)
`died`	Censoring (0) vs drop-out (1)
`sex`	gender, 0=males, 1=females
`grades`	high-school grades 1=high 5=low
`parttime`	part-time student 1=yes, 0=no
`lag`	time lag high-school graduation – entry in college (in months)
`mrg`	time of marriage; 99=never (number of months since jan 1980)
`stm`	time of entry college (number of months since 1980)

```
. stset time died, id(caseid) noshow
note:  making entry-time variable t0
       (within caseid, t0 will be 0 for the 1st observation and the
       lagged value of exit time time thereafter)
```

We check how many people married before entering school?

```
. count if mrg<stm
 4
```

and how many people married during school

```
. count if mrg>=stm & mrg<=stm+time
 10
```

Now create the time-varying covariate `married`, with the associated data manipulation. Note that `mrg` − `stm` provides the time-of-marriage relative to entry at college. For students who never married, `mrg` = 99.

```
. stegen married if mrg~=99, at(mrg-stm) from(0) to(1)
number of episode splits : 10
```

Alternatively, we could have recoded the missing value code 99 to Stata's missing value code ".". Invoking `stegen` with the option `censor` would then imply that the transition has not yet occurred.

```
. replace mrg = . if mrg==99
. stegen married, at(mrg-stm) from(0) to(1) censor
```

Let us verify the number of episodes in which a student is married. This should equal the number of students who married before college, and those who married during college.

```
. tab married
    married |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        261       94.91       94.91
          1 |         14        5.09      100.00
------------+-----------------------------------
      Total |        275      100.00
```

We can now fit a survival time model.

```
. stcox married sex grades parttime lag, nolog
Cox regression -- entry time t0
No. of subjects =          265                     Log likelihood =  -546.62949
No. of failures =          107                     chi2(5)        =       40.81
Time at risk    =       8086.1                     Prob > chi2    =      0.0000
------------------------------------------------------------------------------
    time |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 married |   3.859737    1.64504      3.169   0.002        1.67406    8.899064
     sex |   1.295683   .2588462      1.297   0.195       .8758891    1.916674
  grades |   1.398652   .1241915      3.779   0.000       1.175245    1.664528
parttime |   3.493112   .9206143      4.746   0.000       2.083902    5.855282
     lag |   .9925116   .0073259     -1.018   0.309       .9782565    1.006974
------------------------------------------------------------------------------
```

We conclude that married students have a 4 times higher rate of drop out than non-married students. One may wonder whether the effects of marriage for men and women are the same. This can be easily tested via an interaction effect between the time-constant variable `sex` and the time-varying variable `married`. Note that we can now use the normal Stata command `generate`.

```
. gen msex = sex * married
. stcox married sex grades parttime lag msex, nolog
Cox regression -- entry time t0
No. of subjects =          265                     Log likelihood =  -546.62631
No. of failures =          107                     chi2(6)        =       40.81
Time at risk    =       8086.1                     Prob > chi2    =      0.0000
------------------------------------------------------------------------------
```

```
    time |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+-------------------------------------------------------------------
 married |   3.679398   2.731387      1.755   0.079       .8587974    15.76387
     sex |   1.290915   .2646478      1.246   0.213        .863763    1.929304
  grades |   1.397946   .1244145      3.764   0.000       1.174181    1.664354
parttime |   3.504788   .9348628      4.702   0.000       2.077846    5.911671
     lag |   .9924686   .0073433     -1.022   0.307       .9781799    1.006966
    msex |   1.075106   .9792943      0.080   0.937       .1803463    6.409073
-------------------------------------------------------------------------------
```

Example 3

Many survival time data are very discrete. For instance, in research on labor market mobility, career changes are often only recorded in units of a month, reflecting both the institutional organization of the labor market (labor contracts usually expire by the end of the month) and "crude" measurement ("interval-censoring" to use the technical term) of careers in interviews. Discrete time methods are thus fairly popular, especially among sociologists. In this approach, we analyze the conditional probability π_{it} of "death" in a period, conditional on being "alive" at the beginning of the period. This conditional probability π_{it} is modeled in some binary response model using covariates that depend only on time (compare the baseline hazard in proportional hazards models) and covariates that depend on personal characteristics (and possibly time; these are of course time-varying covariates).

The discrete time approach requires the creation of a so-called "person-period" file in which a person (case) is described via as many records as the number of periods that he was at risk until failure or censoring. Such a file can easily be created with the `stsplit` command with the option `every(#)`, where `every()` specifies the degree of coarseness of the required time discretization. This will be illustrated now with our subset of the cancer data.

```
. stset studytim died
. gen S = _n
. recode S 1 14 25 45 46 = 1 * = 0
```

For comparison with the discrete-time version below, we estimate the continuous time models that assume that the hazard rate is time-constant ("exponential regression") or is additive in log(waiting time) ("Weibull regression").

```
. stereg age drug, nolog noshow
Exponential regression -- entry time 0
log relative hazard form
No. of subjects =           48                  Log likelihood =  -48.837598
No. of failures =           31                  chi2(2)        =       25.01
Time at risk    =          744                  Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.081641   .0347794      2.441   0.015       1.015578    1.152001
    drug |   .3625496   .0883076     -4.165   0.000       .2249247    .5843833
------------------------------------------------------------------------------

. stweib age drug, nolog noshow
Weibull regression -- entry time 0
log relative hazard form
No. of subjects =           48                  Log likelihood =  -42.662839
No. of failures =           31                  chi2(2)        =       35.92
Time at risk    =          744                  Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.117297   .0391618      3.164   0.002       1.043119    1.196751
    drug |   .2583268   .0681501     -5.131   0.000        .154032    .4332393
------------------------------------------------------------------------------
    ln p |   .5639891   .1404668      4.015   0.000       .2886792    .8392989
       p |    1.75767                                    1.334664    2.314744
     1/p |    .568935                                    .4320133    .7492525
------------------------------------------------------------------------------
```

Note that in these data, the survival time is an integer variable with measurement unit equal to 1 month. We can create a person-period file with periods of 1 original unit as

```
. stsplit Period, every(1)
number of episodes generated : 696
```

The variable `period` numbers periods consecutively. We want to stress that `stsplit` returns st-data that are equivalent to the original data, but much less economical in terms of memory requirements. This becomes clear from the following selected list of cases:

```
. sort id Period
. by id: list Period t0 studytim died age drug, noobs, if S
-> id=       1
   Period        t0  studytim      died       age      drug
        1         0         1         1        61         1

-> id=      14
   Period        t0  studytim      died       age      drug
        1         0         1         0        55         1
        2         1         2         0        55         1
        3         2         3         0        55         1
        4         3         4         0        55         1
        5         4         5         0        55         1
        6         5         6         0        55         1
        7         6         7         0        55         1
        8         7         8         0        55         1
        9         8         9         0        55         1
       10         9        10         0        55         1
       11        10        11         1        55         1

-> id=      25
   Period        t0  studytim      died       age      drug
        1         0         1         0        49         2
        2         1         2         0        49         2
        3         2         3         0        49         2
        4         3         4         0        49         2
        5         4         5         0        49         2
        6         5         6         0        49         2
        7         6         7         0        49         2
        8         7         8         0        49         2
        9         8         9         0        49         2
       10         9        10         0        49         2
```

(*output omitted*)

Consider case `id == 14` with entry time 0 and survival until time (month) 11 at which she died of cancer. This history is represented by 11 cases associated with survival months. All but the first period (month) have late entry, and all but the last period are censored. Case `id == 14` has a survival time of 10 periods (months) and all periods are censored. Note that in the expanded data, the variables `age` and `drug` are time-constant.

We can estimate exponential regression models to these data. Invoking `stereg` simply reproduces the output shown above, but takes more time because of the non-economical representation in person-period data. We can also use the data to estimate a discrete time analog of an exponential regression model. In this model, the failure/censor variable becomes the dependent variable, while the survival time variable itself need not be directly included. Here, we use the logistic model.

```
. logistic died age drug
Logit Estimates                                     Number of obs =    744
                                                    chi2(2)       =  26.37
                                                    Prob > chi2   = 0.0000
Log Likelihood = -115.67875                         Pseudo R2     = 0.1023
------------------------------------------------------------------------------
    died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.088742   .0369092      2.508   0.012       1.018752     1.16354
    drug |   .3412381   .0861308     -4.260   0.000       .2080695    .5596371
------------------------------------------------------------------------------
```

The odds ratios of the variables in the discrete time model are indeed very similar to the hazard ratios of the variables in the continuous time model. Also, the standard errors are comparable. You may actually prefer robust standard errors. It can be shown that you have to specify the variant with clustering on the case identifier (`id`) to obtain the correct results for the robust/sandwich estimator.

```
. logistic died age drug, cluster(id)
```

```
Logit Estimates                                         Number of obs =    744
                                                        chi2(2)       =  29.24
                                                        Prob > chi2   = 0.0000
Log Likelihood = -115.67875                             Pseudo R2     = 0.1023

                              (standard errors adjusted for clustering on id)
------------------------------------------------------------------------------
             |               Robust
        died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   1.088742   .0235287      3.934   0.000       1.043589    1.135848
        drug |   .3412381   .0761123     -4.820   0.000       .2203937    .5283428
------------------------------------------------------------------------------
```

While in most applications scholars use the binary regression model with the logit-link, the complementary log-log link (in Stata: `cloglog` or `glm ..., fam(bin) link(c)`) is actually more appropriate if the time-discreteness is due to crude measurement of a continuous survival time that follows a proportional hazards model (Collett 1994). Generally, however, the results will be close.

```
. glm died age drug, fam(bin) link(c) eform nolog
Residual df  =       741                                No. of obs =       744
Pearson X2   =  759.0694                                Deviance   =  231.2681
Dispersion   =  1.024385                                Dispersion =  .3121027
Bernoulli distribution, cloglog link
------------------------------------------------------------------------------
        died |       ExpB   Std. Err.       z     P>|z|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   1.085625   .0349721      2.550   0.011         1.0192    1.156379
        drug |   .3503073   .0856836     -4.288   0.000       .2168948    .5657822
------------------------------------------------------------------------------
```

The Weibull regression assumes that the hazard rate $h(t)$ takes the form

$$h(t) = pt^{p-1}\exp(x'\beta) \quad t \geq 0$$

or

$$\text{logit } \pi(t) \approx \log \pi(t) \approx \log h(t) = \log p + (p-1)\log t + x'\beta$$

We can thus estimate a discrete time analog to the Weibull regression model by incorporating a covariate `log(time)`. Since $\log(0)$ is not well defined here, we arbitrarily define $\log(0) = 0$.

```
. gen lntime = cond(t0>0,ln(t0),0)
. logistic died age drug lntime, nolog
Logit Estimates                                         Number of obs =    744
                                                        chi2(3)       =  35.60
                                                        Prob > chi2   = 0.0000
Log Likelihood = -111.06296                             Pseudo R2     = 0.1381
------------------------------------------------------------------------------
        died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   1.121709   .0413951      3.112   0.002       1.043441    1.205848
        drug |   .2447847    .068893     -5.001   0.000       .1410001    .4249611
      lntime |   2.137748   .5785327      2.807   0.005       1.257761    3.633413
------------------------------------------------------------------------------
```

The effects of `age` and `drug` are roughly the same for the continuous and discrete-time version. But what about the shape parameter p? Above, we have seen that the maximum-likelihood estimator of p is 1.758. According to the output of `logistic`, we have $\exp(\widehat{p}-1) = 2.134$, or $\widehat{p} = 1.786$ which is again close to the maximum-likelihood estimator.

It is also possible to estimate discrete time analogs to survival time models that are currently not supported by Stata. An example is the Gompertz/Bailey regression model that assumes that the (log)-probability of failure depends linearly on the survival time. The discrete-time variant of this model can be estimated by simply including the `t0`-variable in the model.

```
. logistic died age drug t0
```

```
Logit Estimates                                     Number of obs =    744
                                                    chi2(3)       =  37.84
                                                    Prob > chi2   = 0.0000
Log Likelihood = -109.94389                         Pseudo R2     = 0.1468
------------------------------------------------------------------------------
    died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.125063   .0421818      3.143   0.002       1.045353    1.210851
    drug |   .2085663   .0646071     -5.060   0.000         .11365    .3827531
      t0 |   1.097513   .0299863      3.406   0.001       1.040286    1.157887
------------------------------------------------------------------------------
```

Finally, Efron has suggested discrete time methods with a flexible model for the pure time dependency as an alternative to Cox regression. This can be accomplished with a linear spline.

```
. mkspline spt5 5 spt10 10 spt20 20 sptover2 = t0
. logistic died age drug spt*
Logit Estimates                                     Number of obs =    744
                                                    chi2(6)       =  38.13
                                                    Prob > chi2   = 0.0000
Log Likelihood = -109.79985                         Pseudo R2     = 0.1479
------------------------------------------------------------------------------
    died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.126712   .0425537      3.159   0.002        1.04632     1.21328
    drug |   .2083371   .0656665     -4.977   0.000       .1123243      .38642
    spt5 |   1.203756   .2143015      1.042   0.298       .8491831    1.706379
   spt10 |   1.034258   .1539526      0.226   0.821       .7725472    1.384628
   spt20 |   1.107736   .0889251      1.275   0.202       .9464653    1.296485
sptover2 |   1.094208   .0929205      1.060   0.289       .9264363    1.292362
------------------------------------------------------------------------------
```

According to the estimated model, we have little reason to assume that the (logit-) conditional probability of failure depends nonlinearly on survival time. The disadvantage of linear splines is, of course, that the knots have to be specified exogenously. Fractional polynomials provide an alternative flexible specification of the baseline that do not suffer the same disadvantage.

```
. fracpoly logistic died t0 age drug
.........
-> gen t0_1 = x^2
-> gen t0_2 = x^3
where: x = (t0+1)/10
Logit Estimates                                     Number of obs =    744
                                                    chi2(4)       =  37.91
                                                    Prob > chi2   = 0.0000
Log Likelihood = -109.91189                         Pseudo R2     = 0.1471
------------------------------------------------------------------------------
    died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    t0_1 |   2.059963   .7031101      2.117   0.034       1.055179    4.021546
    t0_2 |   .8628029   .0936377     -1.360   0.174        .697482    1.067309
     age |   1.123725   .0421692      3.108   0.002       1.044041    1.209491
    drug |   .2104672   .0655291     -5.005   0.000       .1143305     .387442
------------------------------------------------------------------------------
Deviance:  219.824. Best powers of t0 among 44 models fit: 2 3.
```

With `fracplot` the pure time effect can be assessed graphically. According to this plot, the discrete hazard is approximately S-shaped in survival time.

So far, we have included `age` as a time-constant variable in the analyses. However, `age` is clearly time-dependent—some of the pure time effects that we saw before, but thus is interpretable in terms of changing age. A time-varying covariate `Dage` is easily constructed. We have to be aware that `age` is in years, while the survival time `studytim` is measured in months.

```
. gen Dage = age + t0/12
```

Again, we can estimate a discrete analog to exponential regression.

```
. logistic died Dage drug
```

```
Logit Estimates                                     Number of obs =    744
                                                    chi2(2)       =  28.16
                                                    Prob > chi2   = 0.0000
Log Likelihood = -114.78687                         Pseudo R2     = 0.1092
------------------------------------------------------------------------------
    died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    Dage |   1.103106    .038183      2.835   0.005       1.030751     1.18054
    drug |   .3274351    .083553     -4.375   0.000       .1985731    .5399208
------------------------------------------------------------------------------
```

One can construct other time-varying covariates in terms of `t0`. Since the cancer data does not provide any meaningful data here, we leave this point to the reader.

In this discussion, we have chosen to consider discrete time in units of one month. This is clearly somewhat arbitrary. Why not choose smaller or larger periods? One can specify a shorter period length, e.g., of days, as

```
. stsplit X, every(12/365)
```

It can be shown mathematically that with decreasing episode length, the estimates of, for instance, the discrete time Weibull model converge to the estimates of the continuous time model. In practice, we often have to move in the other direction: larger periods. The reason is that person-period files may consume a fair amount of memory if the dataset comprises many cases, or the observation times are quite large. The `stsplit` command can also be used to create a person-period file in which a period lasts a quarter (3 months).

```
. use cancer, clear
. stset studytim died
. stsplit Period, every(3)
number of episodes generated : 218
```

We show again some interesting cases in the expanded format.

```
. sort id Period
. by id: list Period t0 studytim died age drug if S, noobs
-> id=       1
   Period        t0  studytim      died       age      drug
        1         0         1         1        61         1

-> id=      14
   Period        t0  studytim      died       age      drug
        1         0         3         0        55         1
        2         3         6         0        55         1
        3         6         9         0        55         1
        4         9        11         1        55         1

-> id=      45
   Period        t0  studytim      died       age      drug
        1         0         3         0        60         3
        2         3         6         0        60         3
        3         6         9         0        60         3
        4         9        12         0        60         3
        5        12        15         0        60         3
        6        15        18         0        60         3
        7        18        21         0        60         3
        8        21        24         0        60         3
        9        24        27         0        60         3
       10        27        30         0        60         3
       11        30        33         1        60         3

-> id=      46
   Period        t0  studytim      died       age      drug
        1         0         3         0        62         3
        2         3         6         0        62         3
        3         6         9         0        62         3
        4         9        12         0        62         3
        5        12        15         0        62         3
        6        15        18         0        62         3
        7        18        21         0        62         3
        8        21        24         0        62         3
        9        24        27         0        62         3
       10        27        30         0        62         3
       11        30        33         0        62         3
       12        33        34         0        62         3
```

While in the expanded format generated by `every(1)`, the variable `Period` and `studytim` were identical, this is no longer the case with expansion using a different period length. Now `t0` and `studytim` are still measured in the original time scale (months), while `Period` has scale "quarter."

We can still easily estimate a discrete analog to Gompertz regression.

```
. logistic died age drug t0, nolog
Logit Estimates                                   Number of obs =    266
                                                  chi2(3)       =  36.78
                                                  Prob > chi2   = 0.0000
Log Likelihood = -77.365507                       Pseudo R2     = 0.1920
------------------------------------------------------------------------------
    died | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.124787    .045369     2.915   0.004       1.039289    1.217318
    drug |   .1881332   .0654217    -4.804   0.000       .0951629    .3719318
      t0 |   1.100035   .0329813     3.180   0.001       1.037256    1.166614
------------------------------------------------------------------------------
```

Comparing the output of the discrete Gompertz models with data in months or time-aggregated to quarters, we see that the differences are very small indeed. In my experience, this is generally the case.

Example 4: Continuously varying covariates in Cox regression

We will now discuss how to estimate Cox regression models with continuously varying covariates. We want to stress that the method outlined below is not applicable to parametric survival time models. It is important to recall a theoretical property of the estimation method of Cox regression: the partial-likelihood inference method used to estimate the Cox model depends on survival data only via the composition of the risk pools at the failure times, i.e., at the times at which at least one of the subjects in the sample failed (died). Thus, in partial-likelihood estimation of a Cox regression model, covariates are only "evaluated" at the failure times; it does not matter what happened with covariates between failure times. It follows that "continuously varying covariates" in Cox regression are essentially covariates that change at failure times. Note, again, that these time points depend on the sample, and are not defined exogenously. In particular, and somewhat oddly, these time points involve what happened to other subjects, and do not depend on characteristics of the subjects themselves. Moreover, inference is not affected if a subject "disappears" from observations between failure times, as if a subject is only at risk during an infinitesimal period at the failure times.

How can these theoretical properties of partial-likelihood inference be put to use for Cox regression analyses with time-varying covariates? The command `st_rpool` (Gould 1997) does the more tricky part of expanding a subject to all event times at which a subject was at risk. `st_rpool` modifies the failure/censor indicator ("died"), and returns a variable that numbers the risk pools sequentially in time. Unfortunately, `st_rpool`-modified data are designed to be suitable for use by `clogit` for an "exact" treatment of ties (see `stcoxe`) and are no longer in an appropriate st-format: `st_rpool` does not modify the entry times `t0` and the failure times `t`. The new command `stcoxtvc` is a fairly simple shell around `st_rpool` that forms the risk pools, but leaves the data intact in st-format. Before one starts creating time-varying covariates, one simply invokes `stcoxtvc`. Time-varying covariates can now be constructed in terms of the time variable t, i.e., the name of the survival time variable. Readers familiar with the survival modules of other statistical programs will be used to defining time-varying covariates in terms of a "system variable," labeled TIME (BMDP), t_ (Statistica), T_ (SPSS), (time) (LIMDEP), TIME (TDA), etc. For instance, in some demographic analysis, you want to include a time-varying covariate "marital status" based on a variable `tmarried`, where time is already defined relative to time at risk. After invoking `stcoxtvc`, this is achieved by simply

```
. gen mstatus = t > tmarried
```

By this simple application of `stcoxtvc` (or, really, `st_rpool`), Stata users now can define time varying covariates using the name of the failure time variable! The only real distinctions with other programs are operational: `stcoxtvc` (`st_rpool`) may require a huge amount of memory. Thus, users should only keep the cases that are used, and drop all variables that will not be used as time-constant covariates, or are used to construct the time-varying covariates. If you invoke `stcoxtvc` with a varlist, it will keep only the selected variables, along with the variables that are used by the `st` package (the entry and survival times, etc.) On the other hand, if `stcoxtvc` (`st_rpool`) can form the risk pools, Stata is really *very* much faster than these other programs.

As a first illustration, we consider again the data on school drop-out. In Example 2 we saw that being married increases the drop-out rate four-fold. We want to study whether the duration of marriage affects the drop-out rate even further. After loading the data and declaring the key variables with `stset`, we prepare for Cox regression with time-varying covariates via risk-pool expansion.

```
. stcoxtvc
number of episodes increased from 265 to 6707
```

To define the time-varying dummy variables `married`, we simply have to compare the (calendar) time at an episode with the calendar time of marriage.

```
. gen married = cond(mrg~=99, time+stm > mrg, 0)
```

Recall that in the risk-pool expanded data format, the dependent variable (here: `time`) has become dynamic. We can now refit the Cox regression model shown above.

```
. stcox married sex grades parttime lag, nolog
Cox regression -- entry time t0
No. of subjects =          265                     Log likelihood =  -546.62949
No. of failures =          107                     chi2(5)        =       40.81
Time at risk    =      .006707                     Prob > chi2    =      0.0000
------------------------------------------------------------------------------
    time |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 married |   3.859737    1.64504      3.169   0.002        1.67406    8.899064
     sex |   1.295683   .2588462      1.297   0.195       .8758891    1.916674
  grades |   1.398652   .1241915      3.779   0.000       1.175245    1.664528
parttime |   3.493112   .9206143      4.746   0.000       2.083902    5.855282
     lag |   .9925116   .0073259     -1.018   0.309       .9782565    1.006974
------------------------------------------------------------------------------
```

Apart from the "time at risk" nothing has changed—as it should be. The "time at risk" has become so small because in the risk-pool expanded format, subjects are only at risk during the very small periods at which one of the subjects failed (dropped-out). To assess the effects of duration of marriage, a continuously varying covariate is created.

```
. gen dur = max(time+stm-mrg,0)
. stcox married dur sex grades parttime lag, nolog
Cox regression -- entry time t0
No. of subjects =          265                     Log likelihood =   -546.4297
No. of failures =          107                     chi2(6)        =       41.21
Time at risk    =      .006707                     Prob > chi2    =      0.0000
------------------------------------------------------------------------------
    time |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 married |   2.784656   1.923157      1.483   0.138       .7192982    10.78038
     dur |    1.03517   .0543132      0.659   0.510       .9340087    1.147288
     sex |   1.288459    .257883      1.266   0.205       .8703702     1.90738
  grades |   1.406567   .1257351      3.816   0.000       1.180512     1.67591
parttime |   3.508604   .9250074      4.761   0.000       2.092781    5.882269
     lag |   .9921429    .007358     -1.064   0.287       .9778258     1.00667
------------------------------------------------------------------------------
```

We find no evidence that duration of marriage affects drop-out, though we have to be careful with this conclusion since being married and the duration of marriage are relatively highly correlated in these data.

In a second illustration, we tackle the problem discussed by Bill Gould in the statistics part of the Frequently Asked Questions on Stata's internet site (`www.stata.com`). Gould explains how an interaction effect between dynamic time and a time-constant covariate can be estimated via a person-period file—the method that we discussed above—and he illustrates this method in an analysis of the treatment of cancer replicating Collett (1994).

$$\log h(t) = a(t) + .136 * \texttt{age} \; - .532 * \texttt{treatment} \; + .003 * (\texttt{treatment} \; - 1) * (t - 470)$$

With `stcoxtvc`, this becomes much easier.

```
. describe
Contains data
  obs:            26
 vars:             5
 size:           624 (100.0% of memory free)
------------------------------------------------------------------------------
   1. patient     float  %9.0g
   2. time        float  %9.0g                 survival time (days)
   3. dead        float  %9.0g                 dead
   4. treat       float  %9.0g                 1=single 2=combined
   5. age         float  %9.0g                 age
------------------------------------------------------------------------------
```

```
. stset time dead
              id:  --                    (meaning each record a unique subject)
      entry time:  --                    (meaning all entered at time 0)
       exit time:  time
  failure/censor:  dead
```

`stcoxtvc` is invoked to transform the data into the risk-pool format that is so suitable for Cox regressions with time-varying covariates.

```
. stcoxtvc
st key variables were created
              id:  id               defined to _n: each record a unique subject
      entry time:  t0               defined to 0, meaning all entered at time 0
       exit time:  time
  failure/censor:  dead
number of episodes increased from 26 to 230
```

Note that `stcoxtvc` has defined an id-variable and an entry-time variable. `stcoxtvc` reports that it added $230 - 26 = 204$ cases. The size of the risk-pool data format (230) which compares quite favorably to the size of the person-period file (with time-unit 1): This file contained over 15000 observations.

We can now construct an interaction effect between the time-constant treatment variable and the dynamic waiting time variable (here `time`).

```
. gen ctime = (treat-1)*(time - 470)
```

That is all the data manipulation required. And now we can estimate the Cox regression model:

```
. stcox age treat ctime, nohr nolog
    failure time:  time
      entry time:  t0
  failure/censor:  dead
              id:  id
Cox regression -- entry time t0
No. of subjects =           26                  Log likelihood =  -26.819386
No. of failures =           12                  chi2(3)        =       16.33
Time at risk    =       .00023                  Prob > chi2    =      0.0010
------------------------------------------------------------------------------
    time |
    dead |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |    .136387   .0463484      2.943   0.003       .0455457    .2272282
   treat |  -.5319678   .7687557     -0.692   0.489      -2.038701    .9747657
   ctime |   .0034117     .00513      0.665   0.506      -.0066429    .0134663
------------------------------------------------------------------------------
```

These results are identical to those reported by Collett (1994) and Gould (1997). (The reader who has consulted Gould's answer on FAQ may have noticed that Gould stresses that his method is only approximately right. Although Gould is right that Stata can only deal with covariates that change at discrete time points rather than continuously, this does not matter in Cox-regression in which continuously varying covariates are indistinguishable from covariates that change at properly defined time points.).

A third illustration of Cox regression with continuously varying covariates is a specification test for the proportional hazards model.

$$\log h\,(t|x(t)) = a(t) + \beta x(t)$$

Note that this model implies that β is time-constant. If $x(t)$ is itself time-constant, the "proportional" hazard model implies that the hazard rates of different subjects are proportional, i.e., do not depend on time,

$$\frac{h(t,x_1)}{h(t,x_2)} = \frac{\exp(a(t)+\beta x_1)}{\exp(a(t)+\beta x_2)} = \frac{\exp(\beta x_1)}{\exp(\beta x_2)}$$

which is independent of time. Of course, if the covariates are themselves time-dependent, the term proportional doesn't make much sense anymore. A well-known method to test the assumption that β is itself time-constant is to embed β in a time-varying model, and test the hypothesis that β is time-constant. In continuous time, we could consider the linear model

$$\log h(t) = a(t) + (\beta + \gamma t)x(t) = a(t) + \beta x(t) + \gamma(tx(t))$$

and test that there is no (linear) trend in β, i.e., $\gamma = 0$. For the cancer data this procedure looks like this:

```
. stset studytim died, noshow

   data set name:  cancer.dta
             id:  --                    (meaning each record a unique subject)
     entry time:  --                    (meaning all entered at time 0)
      exit time:  studytim
  failure/censor:  died

. stcox age drug, nolog

Cox regression -- entry time 0

No. of subjects =           48                    Log likelihood =  -81.765061
No. of failures =           31                    chi2(2)        =       36.29
Time at risk    =          744                    Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.116351   .0403379      3.046   0.002       1.040025    1.198279
    drug |   .2153648   .0676904     -4.885   0.000       .1163154    .3987605
------------------------------------------------------------------------------
```

We store the results using `lrtest`.

```
. lrtest, saving(0)
```

We go on by first invoking `stcoxtvc` to bring the data into a suitable format, and then creating the interaction effects of the covariates and the survival time variable. Please be aware that forming these interaction effects on the unexpanded data and computing the test using the unexpanded data is quite wrong.

```
. stcoxtvc, noshow

number of episodes increased from 48 to 573

. gen tdrug = studytim * drug
. gen tage  = studytim * age
```

Next we estimate the Cox regression model with the covariates and their interactions with time.

```
. stcox age drug tdrug tage, nolog

Cox regression -- entry time t0

No. of subjects =           48                    Log likelihood =  -81.739763
No. of failures =           31                    chi2(4)        =       36.34
Time at risk    =      .000573                    Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.128015   .0675927      2.010   0.044        1.00302    1.268588
    drug |   .2043147    .131922     -2.460   0.014       .0576362    .7242754
   tdrug |   1.004434    .041842      0.106   0.915       .9256836    1.089884
    tage |   .9990086   .0045143     -0.220   0.826       .9901999    1.007896
------------------------------------------------------------------------------

. lrtest, saving(1)

. * Wald test for proportional hazard assumption
. testparm tdrug tage
 ( 1)  tdrug = 0.0
 ( 2)  tage = 0.0

           chi2(  2) =    0.05
         Prob > chi2 =    0.9747

. * partial-likelihood ratio test for proportional hazard assumption
. lrtest, model(0) using(1)
Warning:  observations differ:  573 vs. 48
Cox:  likelihood-ratio test                           chi2(2)     =       0.05
                                                      Prob > chi2 =     0.9750
```

We tested the assumption that $\gamma = 0$, i.e., there is no time-linear trend in β with a Wald test and with a (partial-)likelihood ratio test. (The warning of `lrtest` that the number of observations differ can be disregarded; `lrtest` doesn't understand the difference between the number of records in a dataset and the number of observations.) These tests are asymptotically equivalent under $\gamma = 0$; I know of no studies comparing the power of these tests against meaningful alternatives. In this case, both specification tests do not reject the proportional hazards model.

A fourth and final illustration of Cox regression with time-varying covariates may be of some interest because of a conceptual failure that we will make. We want to re-analyze the `cancer` data discussed before. One of the explanatory variables for surviving some cancer treatment is `age`. This is clearly a covariate that is varying continuously in time. So let us analyze survival with "dynamic" age.

```
. use cancer
(Patient Survival in Drug Trial)
.  stset studytim died

   data set name:  cancer.dta
              id:  --                    (meaning each record a unique subject)
      entry time:  --                    (meaning all entered at time 0)
       exit time:  studytim
  failure/censor:  died

. stcox age drug, nolog

failure time:  studytim
failure/censor:  died

Cox regression -- entry time 0

No. of subjects =           48                   Log likelihood =  -81.765061
No. of failures =           31                   chi2(2)        =       36.29
Time at risk    =          744                   Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   1.116351   .0403379      3.046   0.002       1.040025    1.198279
    drug |   .2153648   .0676904     -4.885   0.000       .1163154    .3987605
------------------------------------------------------------------------------
```

We prepare for the creation of (continuously) time-varying covariates.

```
. stcoxtvc

st key variables were created

   data set name:  cancer.dta
              id:  id          defined to _n
      entry time:  t0          defined to 0, meaning all entered at time 0
       exit time:  studytim
  failure/censor:  died

number of episodes created: 525
```

Note that `stcoxtvc` automatically created `id` and `t0`, and that 525 records were added.

Note that the variable `studytim` measures time in months. We will create `tvage` as time-varying age (in years).

```
. gen tvage = age + studytim/12

. stcox tvage drug, nolog

    failure time:  studytim
      entry time:  t0
  failure/censor:  died
              id:  id

Cox regression -- entry time t0

No. of subjects =           48                   Log likelihood =   -81.76506
No. of failures =           31                   chi2(2)        =       36.29
Time at risk    =      .000573                   Prob > chi2    =      0.0000
------------------------------------------------------------------------------
studytim |
    died | Haz. Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   tvage |   1.116351   .0403379      3.046   0.002       1.040025    1.198279
    drug |   .2153648   .0676904     -4.885   0.000       .1163154    .3987605
------------------------------------------------------------------------------
```

Compare the output of Cox regressions with age-at-time-zero and time-varying-age respectively. There are no differences in the effects (hazard ratios) of `age` and `tvage`! How can this be? The proportional hazards model with time-varying age can be written as

$$\begin{aligned}\log h(t) &= a(t) + b_1[\texttt{age} + t/12] + b_2[\texttt{drug}] \\ &= (a(t) + b_1 t/12) + b_1[\texttt{age}] + b_2[\texttt{drug}] \\ &= a'(t) + +b_1[\texttt{age}] + b_2[\texttt{drug}]\end{aligned}$$

Cox's semi-parametric regression model seeks to make inferences about the regression coefficients b without making assumption about the baseline hazard. Since the (partial) likelihood does not depend on the baseline hazards $a(t)$ and $a'(t)$, inference on the regression coefficients is not affected! More generally, effects of covariates that change over time in the same way for all cases (within each stratum) are indistinguishable from time-invariant covariates when analyzed with a Cox regression model.

Dependencies

The commands in this insert use commands previously written by Weesie (1997a, 1997b) and Gould (1997). These previously written commands are included on the STB-41 disk.

Programming remarks

1. The Stata parser does not correctly match on parentheses in options. Thus, normal Stata programs (e.g., `hilite`,...) do not allow expressions that include parentheses in options. For the construction of time-varying covariates, this is often an awkward limitation. We use `parsoptp` to provide "real" expressions in the options `at()`, `from()`, `to()` and `every()` of the commands `stegen`, `strepl`, and `stsplit`. If `parsoptp` is not installed, the st-commands still (should) work, but of course you cannot invoke them with expressions with embedded parentheses.
2. By a similar stroke, the command `sttvc` needs to parse the input line into the expression lists, optional `if` and `in` phrases, and options. Expressions may of course contain commas. Thus, one cannot simply decide that all text after the first comma is the options part; rather, one has to search for the first comma that is not embedded within parentheses.
3. Originally, I had written these commands with extensive use of `expand`, `sort`, and `by` constructs. In this format, the programs were tested in teaching as well. Gould's code `st_rpool` used a different approach that depends heavily on the predictable order of records after expansion, and thereby avoids costly sorting. In a number of tests, I learned that my code was sometimes somewhat faster than Gould's type of coding, but never by more than 50%; on the other hand, Gould's coding sometimes was over 300% faster than my code, especially in risk-pool expansion `st_rpool` for large datasets with relatively few ties. I thus decided to switch to Gould's type of coding. The disadvantage is that I made these changes rather quickly, and the new programs are not as well tested as they should be. But, be merry, if something goes wrong, at least you don't have to wait so long to notice it.

Acknowledgments

This project was supported by grant PGS 50-370 of the Netherlands Organization for Scientific Research. Helpful suggestions by Wim Bernasco are gratefully acknowledged. Programming these utilities benefited greatly from inspecting the coding of `st_rpool` by W. Gould (STB-37), which led to a rewrite of the original code that depended more on time-intensive sorting.

References

Collett, D. 1994. *Modelling Survival Data in Medical Research*. London: Chapman & Hall.

Gould, W. 1997. ssa9.1: Survival analysis subroutine for programmer. *Stata Technical Bulletin* 37: 26–30.

Kalbfleisch, J. D. and R. L. Prentice. 1980. *The Statistical Analysis of Failure Time Data*. New York: John Wiley & Sons.

Yamaguchi, K. 1991. *Event History Analysis*. Canbury Park: Sage Publications.

Weesie, J. 1997a. ip14: Programming utility: Numeric lists. *Stata Technical Bulletin* 35: 14–16.

——. 1997b. ip22: Parsing options with embedded parentheses. *Stata Technical Bulletin* 40: 13–15.

sxd1	Random allocation of treatments in blocks

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

The command `ralloc` assists in the design of randomized controlled clinical trials (RCT). `ralloc` produces a data file whose observations are randomly chosen treatment allocations in blocks of varying sizes.

`ralloc` addresses 4 (of the many) objectives of the design of a RCT:

1. Random allocation of treatments to subjects. Each block represents a random permutation of the treatments specified.
2. Avoiding unnecessary imbalance in the number of subjects allocated to each treatment. Allocation within blocks of reasonable size achieves this. In the case of a trial with k treatments, even in the event of unexpected termination of the trial, the imbalance will be at most $1/k$ times the size of the largest block used.
3. Maintenance of blinding by concealing the pattern of the blocks. A sequence of block sizes is chosen at random (from 3 to 7 different sizes, with equal or unequal probabilities, may be specified). Such a scheme makes "breaking the blind" by working out the block pattern extremely difficult.
4. Ensuring that a record is kept of the randomization protocol. Good practice dictates that the randomization schema for a RCT should be able to be reproduced. Ideally, the user will save a log file when running `ralloc`, but `ralloc` also saves information relating to the trial design in notes attached to the data file itself. `ralloc` uses program fragments borrowed from Stata's `notes.ado` and `note.ado` to write a record of the options specified (seed, number of subjects requested etc) and certain other useful information (number of blocks used, number of subjects randomized) as notes into the data file. It does this by copying the options into global macros that can be used by `xmknote.ado` (a subprogram called by `notes.ado`). (The clue that this is possible—that is, saving other than just straight text as a note—comes from the fact that one can time-and-date stamp a note by specifying the `TS` option.)

Syntax

`ralloc` *BlockIdvar BlockSizevar Treatmentvar* [`,` `seed(#)` `nsubj(#)` `ntreat(2|3|4)`
`ratio(1|2|3)` `osize(3|4|5|6|7)` `equal` `init(#)` `saving(`*filename*`)`]

Description

`ralloc` requires specification of 3 variables that will appear in the dataset that the command creates and saves:

BlockIdvar is the variable identifying each block.

BlockSizevar stores the block size.

Treatmentvar stores the randomly allocated treatment; these are 1, 2, 3, 4 labeled as "A" "B" "C" and "D" respectively.

In addition, `ralloc` creates a fourth variable, named *Order*, which stores the original random allocation sequence. This variable may later be used with Stata's `sort` command to recover the sequence should this ever be disturbed.

Note that `ralloc` issues a `clear` command immediately after it is invoked, so data in memory will be lost.

Options

`seed(#)` specifies the random number seed. If unspecified, the default is 123456789.

`nsubj(#)` specifies the total number of subjects requiring a random treatment allocation. If unspecified, the default is 100. `ralloc` may yield a number greater than `nsubj` if this is required to complete the final block.

`ntreat(2|3|4)` specifies the number of treatment arms in the trial. Currently, `ralloc` supports 2, 3 or 4 arms. If unspecified, the default is 2.

`ratio(1|2|3)` specifies the ratio of treatment allocations to the arms of the trial. If unspecified, the default is 1:1 (:1(:1)). Currently, `ratio(2)` and `ratio(3)` may only be specified when `ntreat(2)` is chosen. These yield a 1:2 and 1:3 allocation, respectively.

`osize(3|4|5|6|7)` specifies the number of different size blocks. For example, if 3 treatment arms are chosen, then `osize(5)` will yield possible block sizes of 3, 6, 9, 12, and 15. The default value is 5. Note that it is quite possible not to realize some block sizes if the number of subjects requested (`nsubj`) is low.

`equal` specifies that block sizes will be allocated in equal proportions. In the example given under the `osize` option, each block would appear on roughly 20% of occasions. This may not be desirable: too many small blocks may allow breaking the blind; too many large blocks may compromise balance of treatments in the event of premature closure. The default choice is to allocate treatments in proportion to elements of Pascal's triangle. In the above example, if `equal` were not specified, or, equivalently, `noequal` appeared, allocation of block sizes would be in the ratio of 1:4:6:4:1. That is, the relative frequency of small and large block sizes is down-weighted. See the `init` option below for another way to limit the number of small blocks, albeit at the cost of increasing the number of large blocks. The number and proportions of the different block sizes is shown on-screen as a table at the end of the program.

`init(#)` specifies the initiating value of the sequence defining the block sizes. When not specified the default is the number of treatments given by `ntreat`, except when a `ratio` > 1 has been specified for a 2 treatment trial, when the default initiating value of the block size is $2 + (\texttt{ratio} - 1)$. When specified, `init(#)` must be an integer multiple of `ntreat` or $\big(2 + (\texttt{ratio} - 1)\big)$. The default may also be specified by `init(0)`. For example, in a 3 treatment trial, `init(9)` would, if the default `osize(5)` is chosen, yield block sizes of 9, 12, 15, 18 and 21. If `init` were unspecified, the block sizes would be 3, 6, 9, 12 and 15. Currently, `ralloc` does not provide a choice of the increments in block sizes.

`saving(`*filename*`)` saves the results to a Stata `.dta` file. This "option" is not optional.

Examples

```
. ralloc block size treat, seed(675) sav(mytrial)
```

allocates treatments A and B at random in a ratio of 1:1 in blocks of sizes 2 4 6 8 and 10 to 100 subjects. Block sizes are allocated unequally in the ratio 1:4:6:4:1 (Pascal's triangle). `seed` is set at 675. Sequence is saved to the file `mytrial.dta` in the variable `treat`.

```
. list in 1/12,noobs
    Order      block       size      treat
        1          1          6          B
        2          1          6          A
        3          1          6          A
        4          1          6          B
        5          1          6          A
        6          1          6          B
        7          2          4          A
        8          2          4          A
        9          2          4          B
       10          2          4          B
       11          3          8          B
       12          3          8          A
```

```
. ralloc bn bs Rx, nsubj(920) nt(2) osiz(4) ra(3) init(8) eq sav(mys)
```

allocates treatments A and B at random in ratio of 1:3 in blocks of sizes 8 12 16 and 20 to 920 subjects using Stata's default seed of 123456789. Roughly 25% of blocks will be of each size. Data are saved to `mys.dta`.

```
. ralloc blknum blksiz Rx, ns(4984) osiz(4) ntreat(4) sav(mys)
```

allocates treatments A B C and D at random in ratio of 1:1:1:1 in blocks of sizes 4 8 12 and 16 to 4984 subjects using the default seed. Block sizes are roughly in ratio of 1:3:3:1.

For this last example, the following table will appear on-screen:

```
Frequency of block sizes:
block size  |      Freq.     Percent        Cum.
------------+-----------------------------------
          4 |         62       12.50       12.50
          8 |        183       36.90       49.40
         12 |        185       37.30       86.69
         16 |         66       13.31      100.00
------------+-----------------------------------
      Total |        496      100.00
```

Ideally, this would be captured in the user's log file. Otherwise, the table could subsequently be reproduced from the saved file by issuing the following commands:

```
. sort blknum
. by blknum: keep if _n==1
. display in b "Frequency of block sizes:"
. tab blksiz
```

Do not save this reduced file over the original!

`ralloc` saves the allocation sequence into `mys.dta`, and, at its conclusion, reads this data file back into memory. If memory has not been disturbed, then the command:

```
. tab blksiz Rx
```

produces a table showing the frequency of treatment allocations:

```
           | treatment
block size |      A          B          C          D |   Total
-----------+--------------------------------------------+--------
         4 |     62         62         62         62 |     248
         8 |    366        366        366        366 |    1464
        12 |    555        555        555        555 |    2220
        16 |    264        264        264        264 |    1056
-----------+--------------------------------------------+--------
     Total |   1247       1247       1247       1247 |    4988
```

Note that 4988 subjects were allocated (compared with 4984 requested). The dataset would show that the 4984th subject was the fourth in a block of size 8. This final block's allocation was completed to yield 4988 allocations.

If the `notes` command were issued, we would see

```
_dta:
  1.  Randomisation schema created on 2 Sep 1997 23:40 using ralloc.ado v1.1.2
  2.  Seed used = 123456789
  3.  There were 4 treatments defined
  4.  The treatments were allocated in the ratio 1:1:1:1
  5.  There were 496 blocks of 4 different sizes generated
  6.  The minimum block size is 4 maximum is 16
  7.  Block sizes were allocated proportional to elements of Pascal´s triangle
  8.  There were 4984 allocations requested
  9.  There were 4988 allocations provided to maintain integrity of final block
 10.  The original order of allocation is stored in the variable ´Order´
```

tt7 Random walk tutorial

Albert Verbeek, Utrecht University, Netherlands
Jeroen Weesie, Utrecht University, Netherlands, weesie@weesie.fsw.ruu.nl

The tutorial `randwalk` displays some interesting random walks in 1 and 2 dimensions. In particular, the tutorial demonstrates that while Gaussian (normal) and Cauchy distributions are in many respects quite similar (e.g., both distributions are symmetric and bell-shaped; many observations are required to distinguish them), there is a distinct qualitative difference between Gaussian and Cauchy random walks.

To run the tutorial, you have to move (or copy) the file `randwalk.tut` into your Stata directory, where the other tutorial files (files with extension `*.tut`) are also placed. Note that it is probably not the same directory where your ado files are located. You can then issue the command

```
. tutorial randwalk
```

and read the explanations on the screen.

A first draft of the random walk tutorial was written in Stata version 1 by Albert Verbeek, professor of mathematical sociology at Utrecht University, shortly before he died. According to Verbeek's documentation, the tutorial was based on Huber's ISP random walk demo. I recently relocated this tutorial file, and made the modifications to the coding to make it fit into Stata's tutorial system.

STB categories and insert codes

Inserts in the STB are presently categorized as follows:

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

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

Guidelines for authors

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

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

A submission to the STB consists of

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

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

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

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

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

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

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

International Stata Distributors

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

Company:	Applied Statistics & Systems Consultants
Address:	P.O. Box 1169 Nazerath-Ellit 17100, Israel
Phone:	+972 66554254
Fax:	+972 66554254
Email:	sasconsl@actcom.co.il
Countries served:	Israel

Company:	Dittrich & Partner Consulting
Address:	Prinzenstrasse 2 D-42697 Solingen Germany
Phone:	+49 212-3390 200
Fax:	+49 212-3390 295
Email:	evhall@dpc.de
Countries served:	Austria, Germany, Italy

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

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

Company:	Smit Consult
Address:	Doormanstraat 19 5151 MG Drunen Netherlands
Phone:	+31 416-378 125
Fax:	+31 416-378 385
Email:	j.a.c.m.smit@smitcon.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 3 5978 8329
Fax:	+61 3 5978 8623
Email:	sales@survey-design.com.au
URL:	http://survey-design.com.au
Countries served:	Australia, New Zealand

Company:	Timberlake Consultants
Address:	47 Hartfield Crescent West Wickham Kent BR4 9DW U.K.
Phone:	+44 181 462 0495
Fax:	+44 181 462 0493
Email:	info@timberlake.co.uk
URL:	http://www.timberlake.co.uk
Countries served:	United Kingdom, Eire

Company:	Timberlake Consulting S.L.
Address:	C/ Montecarmelo No 36 Bajo 41011 Seville Spain
Phone:	+34.5.428.40.94
Fax:	+34.5.428.40.94
Teléfono móvil:	+34.39.78.64.31
Countries served:	timberlake@zoom.es Spain

Company:	Timberlake Consultores
Address:	Praceta do Comércio, N° 13–9° Dto. Quinta Grande 2720 Alfragide Portugal
Phone:	+351 (01) 4719337
Telemóvel:	0931 62 7255
Email:	timberlake.co@mail.telepac.pt
Countries served:	Portugal