STATA
TECHNICAL
BULLETIN

November 1998
STB-46

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

Nicholas J. Cox, University of Durham
Francis X. Diebold, University of Pennsylvania
Joanne M. Garrett, University of North Carolina
Marcello Pagano, Harvard School of Public Health
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

dm63	Dialog box window for browsing, editing, and entering observations.

Tony Brady, Imperial College School of Medicine, UK, tbrady@rpms.ac.uk

The facility in Stata 5.0 to program dialog boxes opens up the possibility of using Stata as a powerful and flexible data-entry system. This has many advantages, not the least eliminating the need to transfer datasets from other databases to Stata, with all the associated problems of handling dates, value labels, and missing values. It also makes the enormous variety of Stata commands instantly available to the database manager allowing, for instance, lists of inconsistent or missing data, graphs to spot outliers, and tabulations for detecting invalid codes to be produced easily.

Programming dialog boxes in Stata 5.0 is a slow and tedious task of trial and error. `winshow` was written to automate the production of a dialog box through which observations from the current dataset can be browsed or edited. The same dialog box can be used to enter new data which is appended to the end of the dataset.

A companion program, `winset`, provides a user-friendly interface for setting characteristics which are specific to `winshow`. These characteristics enable useful data-entry features such as range checks on continuous variables, default values for new observations and requirements to enter data into certain variables.

Syntax

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

`call(`*program_name*`)` `caption(`*text*`)` `nonum` `novar` `nodesc` `notype` `nopreserve`]

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

Options

`maxdisp(#)` specifies the maximum number of variables to be displayed on each page. Values greater than 15 tend to cause "too many dialog controls" errors. The default is 14.

`edit` allows the dataset to be modified by changing the values displayed in the edit boxes. By default changes made to the data are ignored.

`new` puts an extra button on the dialog which allows the user to append new observations to the end of the dataset.

`del` puts a delete button on the dialog which deletes the current observation when pressed.

`strict` prohibits the entry of values which do not have a corresponding label for any variable which has an attached value label. By default the user is warned but not prevented from entering such values.

`dateord(`*string*`)` determines how entered dates are to be interpreted, `dmy` by default. Dates can be entered into numeric fields with a `%d` format either directly (as elapsed days since 1 Jan 1960) or as strings which can be interpreted as dates. If a string is entered then the order becomes important to correctly interpret the date. See [U] **30.3.2 The date() function** for details.

`log(`*varlist*`)` requests verbose logging of changes made to the dataset. The values of the variables in *varlist* are reported for the observation being changed.

`call(`*program_name*`)` allows the user to enhance the built-in error checking of modified values. The user's program is passed a variable list of the modified variables and global macros contain their prospective values.

`caption(`*text*`)` overrides the default dialog box title with the user's own.

`nonum` prevents the variable order from being displayed.

`novar` prevents the variable name from being displayed.

`nodesc` prevents the variable label from being displayed.

`notype` prevents the variable type from being displayed.

`nopreserve` prevents `winshow` from issuing a `preserve` on start-up, with the option to `restore` the data (if modified) on closing.

`list` in `winset` displays the current `winshow` settings for *varlist* in the main results window.

Example

It is easiest to demonstrate `winshow` by example. Without arguments, `winshow` puts the user into browse mode, that is, attempts to edit the data are ignored and new observations cannot be appended. For example,

```
. use auto
(1978 Automobile Data)
. winshow
```

results in the dialog box shown in Figure 1.

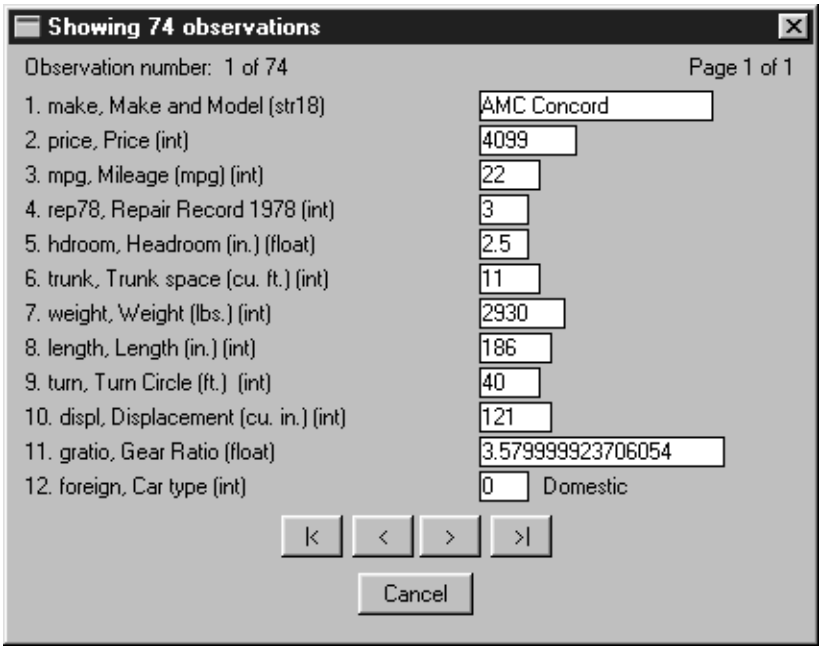

Figure 1. Example of the browse mode of `winshow`

Each variable in the dataset is shown on the left along with its order, label and type. This display can be reduced with the `novar`, `nonum`, `nodesc` and `notype` options. On the right, the values from the current observation are shown in edit boxes. Value labels, if present, are shown to the right of the edit box. In Figure 1 we can see that 0 in the foreign variable has the label "Domestic." Date variables (numeric variables with a `%d` format) are also shown formatted to the right of the edit box. To illustrate this, we will create a fictitious date of manufacture variable:

```
. gen int datemade = 4000 + int(uniform( )*2000)
. format datemade %dd_m_Y
. label var datemade "Date of manufacture"
. winshow, novar notype max(7)
```

On page 2 (use the `More -->` button), our new variable is shown in date format; see Figure 2.

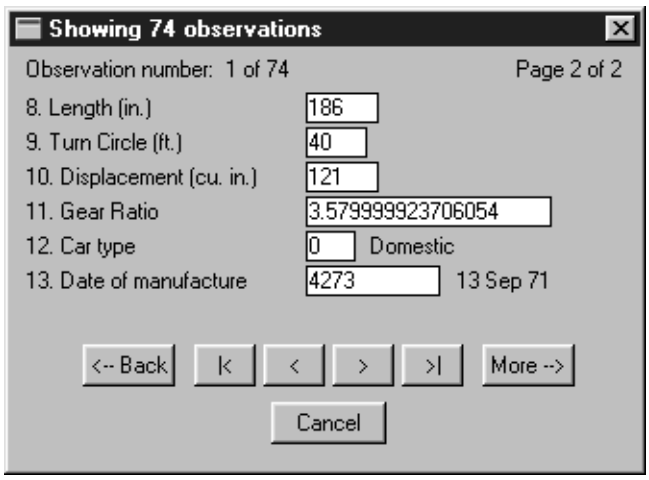

Figure 2. A new variable in date format.

The VCR style buttons towards the bottom of the dialog allow the user to move from one observation to the next, or to the first or last observation.

Data entry

The `edit`, `new`, `del`, `strict`, `dateord()`, `log()`, and `call()` options allow the user to exploit `winshow` for data entry purposes. Attempts to change the data are ignored unless the `edit` option is specified. Then changes to variables are checked against value labels, where these exist, and a warning is issued if a value is entered which does not have a corresponding label (see Figure 3, for example).

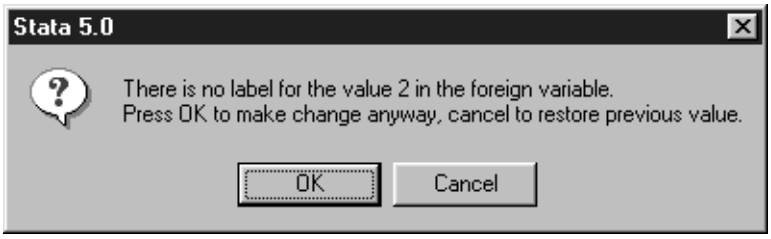

Figure 3. An example of a warning message.

Use of the `strict` option enforces these checks, preventing users from entering values which do not have a label. Similarly, range checks can be introduced for continuous variables (including dates of course) by setting the `[range]` characteristic of the variable. Dates can be entered into variables with a `%d` format in almost any recognized syntax and will be converted into elapsed dates by `winshow`. By default, dates are assumed to be in `dmy` order, but this can be changed with the `dateord( )` option.

New observations are appended to the end of the dataset only when they have been successfully submitted. This avoids the creation of multiple blank observations if the user decides to cancel the dialog box while entering a new observation. The `log( )` option requests detailed reporting of any changes made to the dataset, which may be of use to someone supervising a data-entry clerk, for instance. As an example, suppose the fuel efficiency for the Buick Electra was really 18 mpg, rather than the current value of 15 mpg. Editing this record with `log(make price)` as an option would give the following output:

```
Observation number 5 in the auto data
with the following identifiers:
make:     Buick Electra
price:    7827
was edited on 15 Jul 1998 at 14:44.
- replace mpg = 18 in 5
The variable mpg was changed from 15 to 18
```

Customizing your data-entry system

`winshow` already uses the value labels on categorical variables to check for possible data-entry mistakes. However, you can further customize `winshow` for your own purposes by setting characteristics on variables in your dataset. The characteristics `winshow` recognizes are

Characteristic	Value	Description
`default`	*# or string*	Specifies a default value that the variable is to take when entering a new observation
`len`	*#*	The length in characters of the edit box used to display values of the variable. The default length is either the length of the maximum value of the variable (for numeric) or the space allocated to a string variable (e.g., 12 for `str12`)
`noedit`	*#*	A value of 1 prohibits editing of the variable
`range`	*# # [strict]*	The valid range of input values is defined by the two numbers (lower and upper limit respectively). Optionally *strict* may be specified which prohibits entry of values outside the valid range (including missing)
`req`	*#*	A value of 1 means that the variable cannot be left blank (although missing is allowed)

Characteristics can be set in the usual way with the `char` command; for example,

```
. char price[range] 3000 16000
```

However, the `winset` command provides an easy way of setting and reviewing the characteristics listed above (see Figure 4).

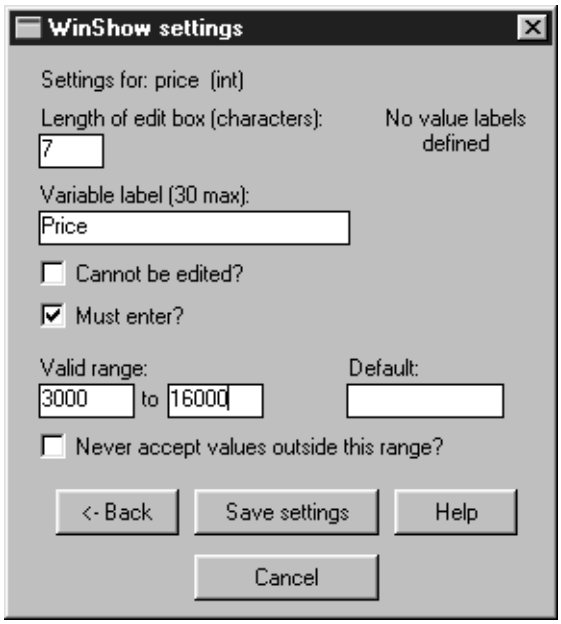

Figure 4. An example of reviewing the variable characteristics via `winset`.

Advanced data-entry control

More advanced customization is available via the `call(`*program_name*`)` option, where *program_name* is a user-written ado file which contains extra checks or data-manipulation. For example, you may wish to check that one date is after another date (such as the date of a follow-up appointment is after the date of entry into the study), or you may wish to calculate a person's age from their date of birth.

The program named by the `call( )` option is passed a list of the variables which have been modified in the current observation. Their prospective values are stored in a series of global macros of the form `$GWSvv#`, where `#` is an index number defined by the variable's order. These can be easily obtained via the `[no]` characteristic of the variable. The program should exit with a return code of one if there is a problem with the modified values. This will prevent the changes to the observation from being submitted, the previous values being restored instead.

To illustrate use of the `call( )` option, consider the following program, `wsauto.ado`, which checks that `mpg`/`displ` (the fuel efficiency per inch) is less than 0.5.

```
program define wsauto
  version 5.0
  local varlist "req exist"
* Get the modified variable names into locals 1, 2 etc...
  parse "`*´"
  parse "`varlist´", parse(" ")
  while "`1´" != "" {
* Only do check if mpg or displ has been modified:
    if "`1´" == "mpg" | "`1´" == "displ" {
* Get index number of mpg:
    local n1 : char mpg[no]
* Get index number of displ:
    local n2 : char displ[no]
    if ${GWSvv`n1´}/${GWSvv`n2´} > 0.5 {    * (mpg/displ) > 0.5
      #delimit ;
      cap window stopbox rusure "The fuel efficiency per cu
        inch is above 0.5 which is unusual."
        "Are you sure mpg and displ are correct?"
        "Press OK to continue anyway.";
      #delimit cr
      if _rc {                  /* User pressed cancel */
        exit 1
      }
      else {                    /* User pressed OK */
        exit
      }
```

```
    }
    else {
       * (mpg/displ) < 0.5
      exit
    }
  }
  mac shift
  }
end
```

This isn't a particularly useful check, but it illustrates the principle and provides a framework which can be adapted to your own situation. The result of attempting to modify the mpg of the AMC Concord to an obviously errant value is shown in Figure 5.

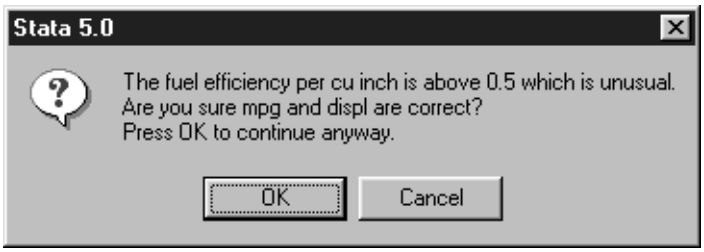

Figure 5. An example of error trapping via advanced data-entry.

Note that `winshow` makes extensive use of global macros starting with the prefix `GWS`. On exiting `winshow`, all global macros with this prefix are dropped.

Acknowledgments

This work was support by the British Heart Foundation (grant number DSVC 3358). Thanks also to Patrick Royston for constructive suggestions on an earlier version of `winshow`.

dm64	Quantiles of the studentized range distribution

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

The studentized range distribution is fundamental to several statistical techniques, including the Tukey *wsd* multiple comparison method. Studentized range quantiles are difficult to calculate and thus many statistics packages do not provide a command or function for them. Highly accurate algorithms do exist (for instance, that of Copenhaver and Holland 1988), but they usually require specialized routines and very careful coding. For example, the Copenhaver and Holland (C–H) algorithm makes iterative use of 12-point Gaussian quadrature and may require quad-precision floating point arithmetic for accuracy in extreme cases. As a result, methods such as Tukey's *wsd* are often still performed using tables of studentized range quantiles. The studentized range tables found in textbooks are, perhaps without exception, derived from H. L. Harter's table, published in Harter (1960) and Harter and Clemm (1959). However, textbooks generally reproduce only a small subset of Harter's table, and the original sources may be hard to locate. (My copy of Harter and Clemm (1959) has faded to unreadability in several places and, more than once, I have found critical pages of our library's archived journals to be missing.)

To be more precise, let $X_{(1)}$ and $X_{(r)}$ be the smallest and largest order statistics in r independent $N(\mu, \sigma^2)$ random variables. That is, $X_{(1)}$ and $X_{(r)}$ are the minimum and maximum values in a simple random sample of size r from the normal distribution with mean μ and variance σ^2. The studentized range is a variable of the form

$$Q = \frac{X_{(r)} - X_{(1)}}{S} \tag{1}$$

where $\nu S^2/\sigma^2$ is a $\chi^2(\nu)$ random variable that is independent of $X_{(r)} - X_{(1)}$. The pth quantile, $q_p(r, \nu)$, of the distribution of Q satisfies

$$\Pr(Q \leq q_p(r, \nu)) = p, \quad \text{where} \quad 0 < p < 1.$$

Thus, the area to the right of $q_p(r, \nu)$ under the density function of Q is $1 - p$.

This insert provides values of $q_p(r, \nu)$ in two forms: a large table in electronic format (`sturng.pdf`) and a Stata command (`qsturng.ado`) for rapidly and accurately approximating $q_p(r, \nu)$ as needed. `sturng.pdf` covers the entire range of Harter's

table, though with some thinning at large values of r, along with the addition of some new values of p. The command `qsturng` is nearly as accurate as Harter's table, and allows for accurate interpolation and extrapolation of quantiles not represented in `sturng.pdf`. Readers interested only in the table may ignore all but the following section. The section below on implementing the `qsturng` command is intended only for readers interested in the details of `qsturng`.

A studentized range table

Typically, values of $q_p(r,\nu)$ are required for multiple comparisons of means in analysis of variance. In this setting, ν is the degrees of freedom for an error mean square, r is the number of means, and p is rather near 1, say $p = 0.95$. Harter's table provides values of $q_p(r,\nu)$ for $p = 0.90, 0.95, 0.975, 0.99, 0.995, 0.999$, $r = 2(1)20(2)40(10)100$, and $\nu = 1(1)20, 24, 30, 40, 60, 120, \infty$. Denote by $\mathcal{T}$ the subset of Harter's table where $r = 2(1)20, 30, 40(20)100$; that is, $\mathcal{T}$ eliminates 11 large (and hence infrequently used) values of r. $\mathcal{T}$ contains $6 \times 24 \times 26 = 3744$ values of $q_p(r,\nu)$, claimed by Harter (1960) to be accurate to the four significant digits tabled. Now, denote by $\mathcal{T}^+$ the table that appends to $\mathcal{T}$ the values of $q_{0.50}(r,\nu)$ and $q_{0.75}(r,\nu)$ for $r = 2(1)20, 30, 40(20)100$, and $\nu = 2(1)20, 24, 30, 40, 60, 120, \infty$.

The included file `sturng.pdf` presents the 4944 quantiles in $\mathcal{T}^+$ as an Adobe PDF document viewable by Acrobat Reader, Version 3.0 or newer. (Visit `http://www.adobe.com/prodindex/acrobat/readstep.html` to download a free copy of Acrobat Reader.) The entries in `sturng.pdf` were computed using a double precision version of the C–H algorithm and then rounded, as in Harter's table, to four significant digits. Entries in the subset $\mathcal{T}$ were then compared with Harter's table. This revealed that about 20% of the values in Harter's table are too large by one unit in the fourth digit, apparently because of slightly conservative rounding. In those cases, `sturng.pdf` contains the value from the C–H algorithm. Discrepancies of any other kind were uncommon, and were resolved by accepting Harter's tabled value. It may be that `sturng.pdf` contains the most accurate table of $q_p(r,\nu)$ currently available.

Approximating studentized range quantiles

The command `qsturng` computes a fast, accurate approximation to $q_p(r,\nu)$. The syntax is

`qsturng` *r* *ν* *p*

where $r \geq 2, \nu \geq 1$, and $0 < p < 1$. So, for example,

```
. qsturng 2 2 .9
  4.129
```

displays the approximate 90th quantile at $r = 2$ and $\nu = 2$. `qsturng` stores its approximation $\widehat{q}_p(r,\nu)$ in global macro `S_1`, so that

```
. di $S_1
4.1294832
```

displays the computed value of $\widehat{q}_{0.90}(2,2)$.

In this particular instance, `4.1294832` is correct to the number of digits shown, and the value `4.129` is precisely the entry given in `sturng.pdf` for this case. Harter's table, however, reports `4.130`; this is an example of the conservative rounding mentioned above. In general, the value displayed by `qsturng` is only an approximation to the relevant entry in `sturng.pdf`, but it is wrong by more than one fourth-digit unit for fewer than 2% of the entries in `sturng.pdf`, and the maximum fourth-digit error is four units. For $\nu > 5$, fewer than 1% of the fourth-digit errors exceed 1. Over the 4944 cases in $\mathcal{T}^+$, the relative error in approximating $q_p(r,\nu)$, as computed by the C–H algorithm, by $\widehat{q}_p(r,\nu)$ never exceeds 0.00075 and is less than 0.0002 for 92% of the cases.

For values of p, r, and ν not represented in $\mathcal{T}^+$ (i.e., not in `sturng.pdf`), `qsturng` provides accurate interpolation routines. Harter (1960, 1145) notes that linear interpolation in ν^{-1} is quite accurate in his table, provided $\nu > 20$. The interpolation routine in `qsturng` is distinctly better, especially at small ν. To illustrate, interpolating between $\nu = 4$ and $\nu = 5$ for $\widehat{q}_p(r, 4.5)$ produces errors no larger than three units in the fourth digit, over all p and r in $\mathcal{T}^+$. Interpolation in r is also much more accurate than in Harter's table, and even extrapolation beyond $r = 100$ is quite feasible. For example, extrapolating to $r = 200$ yields values $\widehat{q}_p(200,\nu)$ that differ from the output of the C–H algorithm by at most 11 fourth-digit units, over all p and ν in $\mathcal{T}$; these errors are of the same size as for interpolating for values in the range $40 < r < 100$ in Harter's table, using the interpolation scheme he recommended. Interpolation in p is the most difficult of the three possibilities, and `qsturng` resorts to quadratic interpolation in this case. The result is quite satisfactory, good enough to permit some extrapolation above $p = 0.999$. In fact, extrapolating to $p = 0.9999$ gives approximate quantiles $\widehat{q}_{0.9999}(r,\nu)$ that are about as accurate as the extrapolations to $r = 200$: Fourth digit errors no greater than 13 units, over the range of (r,ν) values in Harter's table. Finally, while extrapolations for $p < 1/2$ are permitted, it is recommended that `qsturng` be used only for $1/2 < p < 1$.

Implementation

`qsturng` implements a new approximation to $q_p(r,\nu)$, presented in detail by Gleason (1997). The main ideas are that the distribution of Q in Equation (1) depends strongly on the estimator S, and that the denominator of a $t(\nu)$ variable contains a term formally the same as S. Quantiles of $t(\nu)$ can be quickly obtained from the `invt()` function and those quantiles have, in a sense, already captured the effects of S. The strategy is then to use the readily available and very accurate $t(\nu)$ quantiles to simplify the task of estimating $q_p(r,\nu)$.

Let $t_p(\nu)$ be the p-th quantile of the $t(\nu)$ distribution, and start with the fact that

$$q_p(2,\nu) = \sqrt{2}t_{p'}(\nu) \quad \text{where} \quad p' = (1+p)/2$$

for any $0 < p < 1$ and all $\nu \geq 1$. The approximation proceeds by defining the variable

$$y_p(r,\nu) = \frac{q_p(r,\nu)}{\sqrt{2}t_{p'}(\nu)}$$

and then modeling $y_p(r,\nu)$ as $1 + f(p,r,\nu)$ where f is a function with $f(p,2,\nu) = 0$ for all p and ν. This is helpful because while $q_p(r,\nu)$ and $t_{p'}(\nu)$ both change rapidly at small ν, their ratio y is more stable and thus easier to capture with a simple formula. Figure 1 illustrates the point by plotting $y_p(r,\nu)$ and $q_p(r,\nu)$ against $\log(r-1)$ for the entries in $\mathcal{T}^+$ at $p = 0.975$. The unmarked traces plot values of $y_p(r,\nu)$, shown on the left axis; the traces marked with the plot symbol `o` plot the corresponding values of $q_p(r,\nu)$, shown on the right axis. ν assumes the values $1(1)20, 24, 30, 40, 60, 120$, and ∞, reading from top to bottom within each set of traces. There is much more homogeneity of shape across ν in the traces for y than in those for q, and so it is easier to approximate y than q.

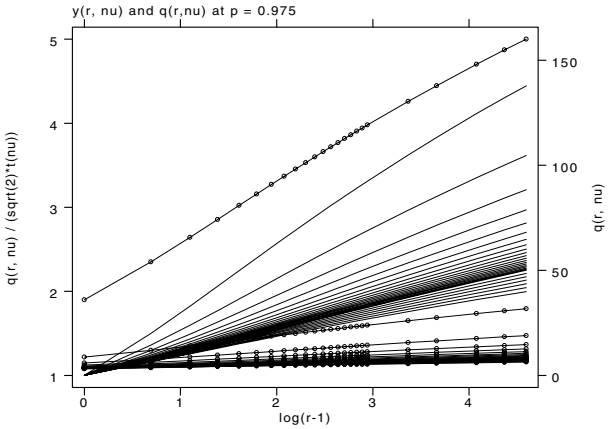

Figure 1. $y_p(r,nu)$ (left axis) and $q_p(r,nu)$ (right axis) versus $\log(r-1)$ at p=0.975.

In fact a very simple form suffices for f; namely a quartic function of $\log(r-1)$, the abscissa of Figure 1. For each of the 206 combinations of p and ν in $\mathcal{T}^+$, model $y = y_p(r,\nu)$ as

$$y = 1 + \sum_{j=1}^{4} \alpha_j \log^j(r-1) \tag{2}$$

using ordinary least squares estimation, say using the `regress` or `fit` command. This leaves a 206×4 table $\mathcal{A}$ of estimated coefficients, $a_j(p,\nu)$; $\mathcal{A}$ is included as part of the `qsturng` command. In effect, given values p and ν represented in $\mathcal{A}$, `qsturng` retrieves the appropriate set of four estimated coefficients from $\mathcal{A}$, and calculates

$$\widehat{y}_p(r,\nu) = 1 + \sum_{j=1}^{4} a_j(p,\nu)\log^j(r-1), \quad \text{and then} \quad \widehat{q}_p(r,\nu) = \sqrt{2}t_{p'}(\nu)\widehat{y}_p(r,\nu), \tag{3}$$

with $t_{p'}(\nu)$ obtained from `invt()`. At $r = 3$ only, `qsturng` adds a small correction to $\widehat{y}$ before computing the second expression in (3). See Gleason (1997) for details.

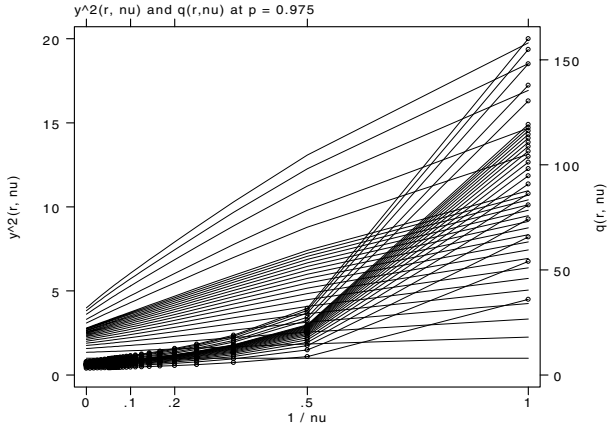

Figure 2. $y_p^2(r,nu)$ (left axis) and $q_p(r,nu)$ (right axis) versus ν^{-1} at p=0.975.

Interpolation and extrapolation in r occur transparently when (3) is evaluated; the gentle curvature in Figure 1 implies that these calculations will be rather accurate. But for (p, ν) combinations not represented in $\mathcal{A}$, explicit interpolation or extrapolation is necessary. The basic strategy is to find interpolation axes on which the relevant pairs of points plot almost as a straight line, so that linear interpolation will be accurate. Working with $y_p(r, \nu)$ rather than $q_p(r, \nu)$ again pays large dividends. Harter recommended interpolating $q_p(r, \nu)$ versus ν^{-1} for values of ν not present in his table. `qsturng` interpolates $y_p^2(r, \nu)$ against ν^{-1} instead; Figure 2 shows why. Again, the marked traces show values of $q_p(r, \nu)$ at $p = 0.975$ while the unmarked traces plot values of $y_p^2(r, \nu)$. The abscissa is ν^{-1}, and within each set of traces the uppermost trace is for $r = 100$, the lowermost for $r = 2$. The traces for y^2 are much straighter than those for q, and so `qsturng` provides improved interpolation accuracy, especially at small values of ν.

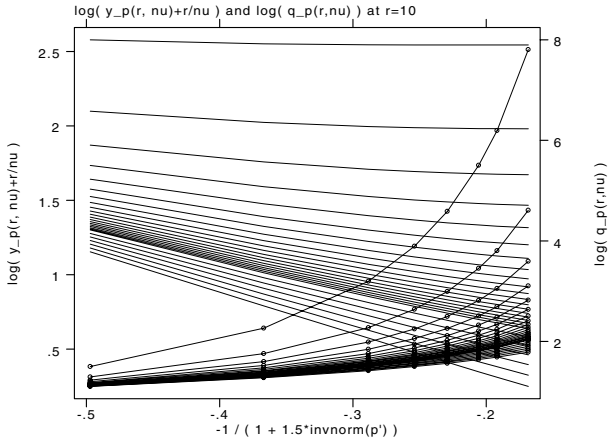

Figure 3. $\ln(y_p(r,nu)+r/nu)$ (left axis) and $\ln(q_p(r,nu))$ (right axis) versus $x=-1/(1+1.5z_{p'})$ at r=10.

Harter (1960, 1145) cautioned users of his table not to interpolate $q_p(r, \nu)$ against p, justifiably so. Still, with a bit more effort, accurate interpolation is possible for values of p not in $\mathcal{A}$. `qsturng` interpolates using $u = \log(y_p(r, \nu) + r/\nu)$ as the ordinate and $x = -1/(1 + 1.5z_{p'})$ as the abscissa, where $z_{p'}$ is obtained as `invnorm((1+p)/2)` for a given value `p`. Figure 3 demonstrates the reason for this peculiar choice of axes. The unmarked traces plot values of u against x at $r = 10$; the marked traces plot $\log(q_p(r, \nu))$, the best choice if using q rather than y. Within each set of traces, the uppermost trace is for $\nu = 1$, the lowermost for $\nu = \infty$. It is clear that the axes x and u are effective in removing curvature, and will give better interpolation accuracy than working with $q_p(r, \nu)$ itself. Still, some curvature remains at small r and ν so `qsturng` uses quadratic interpolation of u against x. This allows interpolation with respect to p to be nearly as accurate as that with respect to r or ν. For example, interpolating for $p = 0.965$ and $p = 0.9925$ yields approximations $\widehat{q}_p(r, \nu)$ that differ from the result of the C–H algorithm by at most three units in the fourth significant digit, and only about 1.5% of the errors exceed one unit in the fourth digit.

Finally, the table $\mathcal{A}$ contains 206×4 coefficients $a_j(p, \nu)$, only a few of which are required in any specific application. `qsturng` deals with this issue by dividing $\mathcal{A}$ into subsets and loading only the portion(s) required. Specifically, $\mathcal{A}$ is recreated by the subroutines `sturng1` ... `sturng8`, each of which can create two matrices containing the $a_j(p, \nu)$ for some particular p in $\mathcal{A}$; the digit in the subroutine name indexes the value of p. The matrices created by these subroutines have names of the form `StuRng`*ab* where a is the digit in the subroutine name, and b is either `0` or `1`. But the 16 matrices the subroutines can create are never used directly. Rather, at invocation, `qsturng` checks to see whether the matrices it requires are already present in

Stata's memory. If not, it issues the commands to create exactly as many matrices as it needs, usually just one. In the canonical application, performing the Tukey *wsd* method in an ANOVA setting, a single $\widehat{q}_p(r,\nu)$ is required, for a p represented in $\mathcal{A}$ (say, $p = 0.95$) and a largish value of ν (say, $\nu \geq 20$). In this very common usage, `qsturng` loads a single 7×4 subset of $\mathcal{A}$. In any case, the submatrices `StuRng`*ab* are left in memory so that subsequent calls to `qsturng` tend to be faster than initial ones.

Acknowledgment

This project was supported by a grant R01-MH54929 from the National Institute on Mental Health to Michael P. Carey.

References

Copenhaver, M. D. and B. S. Holland. 1988. Computation of the distribution of the maximum studentized range statistic with application to multiple significance testing of simple effects. *Journal of Statistical Computation and Simulation* 30: 1–15.

Gleason, J. R. 1997. An accurate, non-iterative approximation for studentized range quantiles. *Computational Statistics & Data Analysis* (submitted).

Harter, H. L. 1960. Tables of range and studentized range. *Annals of Mathematical Statistics* 31: 1122–1147.

Harter, H. L. and D. S. Clemm. 1959. The probability integrals of the range and of the studentized range—probability integral, percentage points, and moments of the range. Technical Report 58–484, Vol. I, Wright Air Development Center, Dayton, OH.

gr29.1	Correction to labgraph

Jon Faust, Federal Reserve Board, faustj@frb.gov

The `labgraph.ado` file shipped with STB-45 contained an error in the code, leftover from debugging, which led to an "x not found" error. This has now been fixed. The `labgraph.hlp` file has had a typographical error corrected.

References

Faust, J. 1998. gr29: labgraph: placing text labels on two-way graphs. *Stata Technical Bulletin* 45: 6–7.

gr32	Confidence ellipses

Anders Alexandersson, Mississippi State University, andersa@rocketmail.com

Syntax

`ellip` *indepvar1* [*indepvar2*] [`if` *exp*] [`in` *range*] [`,` `level(#)` `generate(`*newyvar newxvar*`)`

`add(`*oldyvar oldxvar*`)` `pool(#)` `fyvar(`*fmt*`)` `fxvar(`*fmt*`)` *graph_options*]

Description

`ellip` is a program for graphing confidence ellipses. The variables to be displayed on the vertical and horizontal axes, *yvar* and *xvar* respectively, are coefficient estimates of two independent variables, *indepvar1* and *indepvar2*, from an immediately preceding fit. If *indepvar2* is not specified, it is the constant.

Options

`level(#)` specifies the confidence level, in percent, for confidence ellipses. The default is `level(95)` or as set by `set level`.

`generate(`*newyvar newxvar*`)` creates 400 observations of *yvar* and *xvar*, if necessary by appending observations with missing values to the original variables.

`add(`*oldyvar oldxvar*`)` adds or overlays an old confidence ellipse to the graph. It may be specified only if `generate()` is also specified.

`pool(#)` displays a confidence ellipse labeled `bp` using all the data, a confidence ellipse labeled `b` using a theoretically unproblematic subset, and the locus and a separate table of $\# + 1$ pooled estimates where the problematic subset is discounted by fractional `iweights` at $1/\#$ intervals between 0 and 1; `pool()` works only if `add()` and `if` or `in` are specified.

`fyvar(`*fmt*`)` specifies a display format, such as `%1.0f` or `%3.2g`, for *yvar*.

`fxvar(`*fmt*`)` specifies a display format, such as `%1.0f` or `%3.2g`, for *xvar*.

graph_options are any options allowed with `graph, twoway`. Defaults are: `c(ll) s(.)` (or `c(ll) s(..)` if `add()` is not missing) `t1(" ") t2(" ") l1(`*Estimated indepvar1*`) b2(`*Estimated indepvar2*`)`.

Remarks

In linear regression, a confidence ellipse is the boundary of an elliptical joint $100(1-\alpha)\%$ confidence region for two coefficient parameters. The point representing the coefficient estimates is the center of the confidence ellipse. The larger the confidence level is, the larger the confidence ellipse is. The orientation of the confidence ellipse and the relative lengths of its axes are determined by the estimated covariance matrix. In order to use the `pool(#)` option you must have the `gphdt` programs from `gr20` that appeared in STB-34 installed (Newton and Hardin 1996). If you have installed the `parsoptp` program from `ip22` of STB-40 (Weesie 1997), you can have titles with embedded parentheses. However, then the option names cannot be abbreviated and `ellip` is slower.

Examples

The following three examples illustrate increasingly advanced uses of `ellip`. The examples use the accompanying datasets `fig1.dta`, `fig2.dta`, `fig3.dta` and produce Figures 1 through 3 below. Also included on the STB diskette is the file `fig1to3.do` that can be used to produce the figures. The first example illustrates the basic syntax of `ellip`. Montgomery and Peck (1982, 11–12, 391–392) expected that shear propellant strength linearly depends on the propellant age. Their 95% confidence ellipse for the estimated age and the estimated constant is simply displayed in Figure 1 by

```
. quietly fit strength age
. ellip age
```

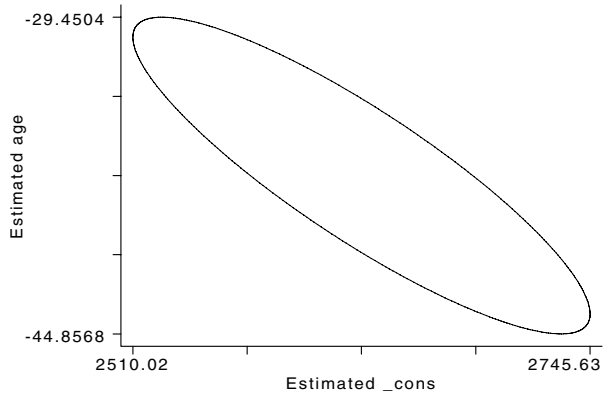

Figure 1. Confidence ellipse for Montgomery and Peck example.

The second example illustrates how `ellip` can produce confidence ellipses with varying confidence levels. Theil (1971, 102) expected that the logarithm of textile consumption per capita in a year is a linear function of the logarithm of real income per capita in that year and of the logarithm of the relative textile price in the same year. Figure 2 reproduces Theil's overlaying graph with 95% and 99% confidence ellipses.

```
. quietly fit logtext logprice loginc, l(99)
. ellip logprice loginc, l(99) g(y99 x99)
. quietly fit logtext logprice loginc
. ellip logprice loginc, g(y95 x95) a(y99 x99)
```

(*Graph on next page*)

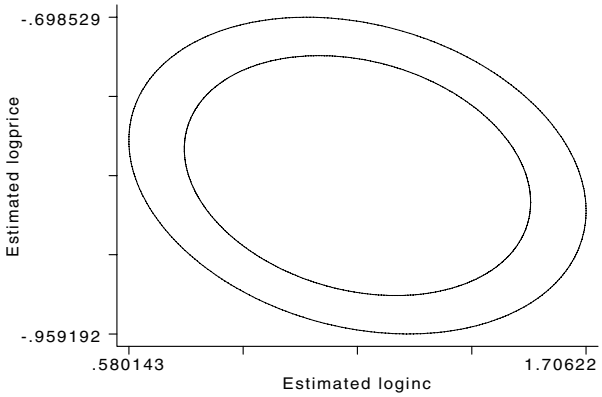

Figure 2. 95% and 99% confidence ellipses for Theil example.

The final example illustrates how `ellip` facilitates regression analysis of disparate observations as in Bartels (1996). Bartels proposed a method of discounting a theoretically problematic subset with a fractional importance weight (using a 50% confidence level), and gave three examples. Unfortunately, it seems that Bartels' Figure 1 graphs individual confidence intervals, that Table 3 for Figure 2 has wrong estimates and standard errors of the constants, and that Figure 3 has positively correlated coefficients but a negative tilt. In the lack of other examples, for my Figure 3, I chose to replicate the least complex figure that Bartels produced; namely his Figure 1 (Bartels, 1996, 924,938), which summarizes his reanalysis of alleged vote fraud in Philadelphia, Pennsylvania in 1993. The court had overturned the election result based on an analysis of the relationship between the Democratic margin in votes cast by machine ballot and the Democratic margin in votes cast by absentee ballot in 21 previous elections in seven Philadelphia Senate districts. Bartels examined the sensitivity of the conclusion by giving less importance weight to other districts than the 2nd Senate district where the disputed election actually occurred. The `pool()` option in Figure 3 discounts the problematic subset with 0.1 or 10% *iweight* intervals. The `format` and *graph_options* and `parsoptp` are useful for the final display. The locus curve is slightly different from the original because Bartels used simple iweighted variables rather than Stata's more accurate `[iweight]` method.

```
. quietly fit absentee machine, l(50)
. ellip machine, l(50) g(ybp xbp) l2title((Machine Vote Margin))
. quietly fit absentee machine if subset==1, l(50)
. ellip machine if subset==1, l(50) g(yb xb) a(ybp xbp) pool(10) /*
    */ fyvar(%4.3f) fxvar(%3.0f) ylab(0.000,0.005,0.010,0.015,0.020) /*
    */ xlab(-300,-200,-100,0,100,200,300) yscale(0.000,0.020) /*
    */ xscale(-300,300) ytick(0.001,0.002,0.003,0.004,0.006,0.007, /*
    */ 0.008,0.009,0.011,0.012,0.013,0.014,0.016,0.017,0.018,0.019) /*
    */ xtick(-280,-260,-240,-220,-180,-160,-140,-120,-80,-60,-40,-20, /*
    */ 20,40,60,80,120,140,160,180,220,240,260,280) /*
    */ l1title(Estimated Slope) l2title((Machine Vote Margin)) /*
    */ b2title(Estimated Intercept) border
```

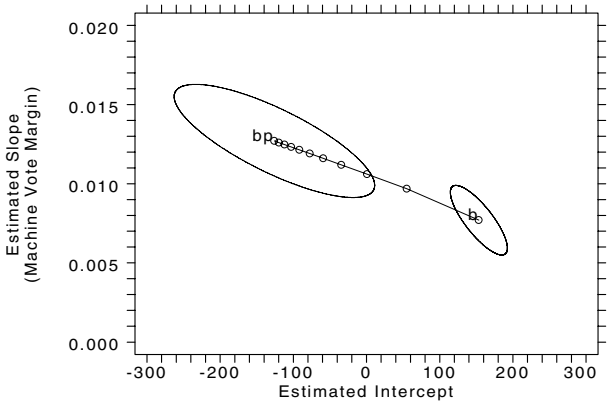

Figure 3. The Bartels' example.

Methods and Formulas

A confidence ellipse can be expressed in algebraic or parametric form. Press et al. (1992, 698) showed that the singular value decomposition can solve the algebraic equation of the confidence ellipse directly. However, because `matrix svd` only works on small datasets, an algebraic scalar version is better (see, for example, Theil, 1971, 133). To avoid having to solve the algebra, I instead used the parametric form mentioned in Douglas (1993, 44).

Acknowledgments

Privately, Nick Cox made helpful comments on an early draft, and my thanks go to William Gould for his explanation on Statalist (January 30, 1998) of how `iweight` works.

References

Bartels, L. M. 1996. Pooling disparate observations. *American Journal of Political Science* 40: 905–942.

Douglas, J. B. 1993. Confidence regions for parameter pairs. *American Statistician* 47: 43–45.

Montgomery, D. C. and E. A. Peck. 1982. *Introduction to Linear Regression Analysis.* New York: John Wiley & Sons.

Newton, H. J. and J. W. Hardin. 1996. gr20: Low-level graphics in data coordinates. *Stata Technical Bulletin* 34: 3–8. Reprinted in *Stata Technical Bulletin Reprints* vol. 6, pp. 27–34.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing.* 2d ed. London: Cambridge University Press.

Theil, H. 1971. *Principles of Econometrics.* New York: John Wiley & Sons.

Weesie, J. 1997. ip22: Parsing options with embedded parentheses. *Stata Technical Bulletin* 40: 13–15. Reprinted in *Stata Technical Bulletin Reprints* vol. 7, pp. 86–89.

gr33 Violin plots

Thomas J. Steichen, RJRT, FAX 336-741-1430, steicht@rjrt.com

Syntax

The syntax of `violin` is

`violin` *yvar varlist* [*weight*] [`if` *exp*] [`in` *range*] [`,`

{ `biweight` | `cosine` | `epan` | `gauss` | `parzen` | `rectangle` | `triangle` }

`n(#) width(#) by(`*byvar*`) truncat(#,#|*) round(#)` *graph_options*]

`fweights` and `aweights` are allowed; see `help weights`.

Description

`violin` produces violin plots: a graphical box plot–kernel density synergism. The violin plot combines the basic summary statistics of a box plot with the visual information provided by a local density estimator. The goal is to reveal the distributional structure in a variable. Much like a traditional box plot, the violin plot displays the median as a short horizontal line, the first-to-third interquartile range as a narrow shaded box, and the lower-to-upper adjacent value range as a vertical line, but it does not plot outside values. Instead, it "boxes" the data with mirrored density curves and labels the y-axis at the minimum, median and maximum observed data values.

`violin` also lists basic descriptive statistics about the data (i.e., the lower and upper adjacent values, the 25th and 75th centiles, the minimum, median and maximum of the data, and the sample size) and it provides information about the density estimation (i.e., the kernel method used, the number of points of estimation, and the resulting scale and width factors). When `by()` is specified, descriptive statistics are displayed for the combined group only. When multiple variables are included in *varlist*, statistics are displayed for the last variable only.

`violin` discards observations on a casewise basis as a function of 1) missing data and 2) the `if` (or `in`) specification (i.e., it ignores the entire observation). This behavior may lead to unexpected results when multiple variables are in *varlist*.

Options

`biweight`, `cosine`, . . . , `triangle` specify the kernel. By default, `epan`, the Epanechnikov kernel, is used.

`n(#)` specifies the number of points at which density estimates will be evaluated. The default is 50.

`width(#)` specifies the bandwidth of the kernel (i.e., the width of the density window around each point). If `width()` is not specified, then the "optimal" width is used; see [R] **kdensity**. For multimodal and highly skewed densities, the "optimal" width is usually too wide and oversmooths the density.

`by(`*byvar*`)` produces separate plots for the groups of observations defined by *byvar* and displays them in a single graph having common vertical scale. `by()` cannot be specified when there is more than one variable in *varlist*.

`truncat(#,#|*)` limits the range of the density curve, either to a range specified as `(#,#)` or to the observed data limits, specified as `(*)`. Regardless of the actual `(#,#)` specification, the maximum range of truncation honored is the observed data limits. The precise truncation points will be the most extreme points within the specified range where the density is calculated (the points of density calculation depend on `n()`, `width()` and the observed data).

`round(#)` rounds the y-axis numeric labels to the value specified. As a result, the labels and their corresponding tic marks may not be placed at the true minimum, median, or maximum values, rather they will be at the rounded values. `round()` has no effect if `ylabel` is specified without arguments but is operative if `ylabel` is not specified or is specified with arguments. The `round()` option follows the rules of Stata's `round(`x`,`y`)` function, with # being the y argument and each label value being the x argument; see [U] **20.3.5 Special functions**.

graph_options are any of the options allowed by `graph, twoway` except `b2title()`, which is ignored; see [R] **graph**. Some options are preset and, although changeable, usually should not be modified. These include `symbol(i)` and `connect(l)` for specifying the plotting symbol and point connection method for the density curve. In addition, `ylabel()` is preset to label only the minimum, median and maximum points. `t1title(Violin Plot)` is preset but can be changed—except when `by()` is specified; in this instance `t1title` is used for the variable name or label. When changeable, use of `t1title(.)` will result in a blank title. Other preset options, such as `pen(2)` for specifying the plot pen color, are intended to be freely changed. A few options, such as the left and right titles, are set (or default to) blank. If specified, they appear beside each plot in a multi-variable graph. Thus these options are generally appropriate only when a single variable is specified. Lastly, the `saving()` option differs slightly from `graph`'s in that the filename extension is always `.gph` and must not be specified.

Explanation

Many tools exist in the exploratory data analysis (EDA) toolkit, and Stata provides the capability to use most of them. However, a quite recent addition—the violin plot—has not yet been implemented in Stata. This insert provides an initial Stata implementation and paraphrases much of the development published by Hintze and Nelson (1998). These authors report that the name *violin plot* was selected because one of the first analyses with the procedure resulted in a graphic with the appearance of a violin.

At its simplest, the violin combines, in a single plot structure, the data summarization features of the box plot with the distributional shape-revealing capability of a local density estimator. Figure 1, which places a traditional box plot beside a violin plot, illustrates their common features using the `mpg` variable from Stata's automobile dataset and provides labels identifying the principal lines used to report these common features.

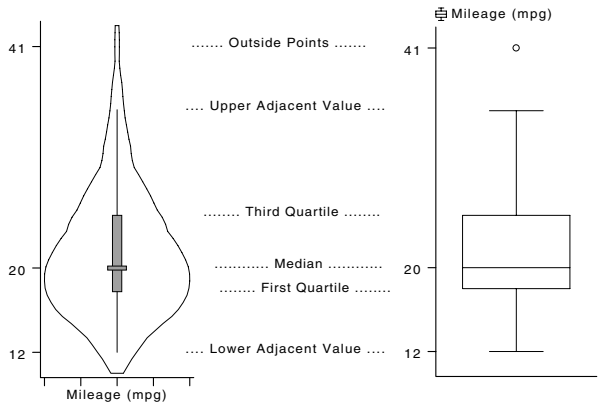

Figure 1. Common features of the box and violin plots.

In general, a box plot shows four main features about a variable: its center, spread, asymmetry, and outliers. As indicated in Figure 1, the first three of these features have direct correspondence in the violin plot. Both plots use the median to define the center, and both use the interquartile range and adjacent points to define the spread and asymmetry. The fourth feature, outliers (or in EDA jargon, *outside points*), are not explicitly identified with individual symbols in the violin plot. Instead, if present, outliers are loosely represented as a slight thickening of the extremes of the mirrored density curves. The density curve

supplements the traditional summary statistics of the box plot by graphically showing the shape of the distribution.

Hintze and Nelson elucidated the violin plot concept using the simple *density trace* procedure described in Chambers et al. (1983). The density trace is an elementary form of a *kernel density estimator*. Kernel density estimators approximate the density $f(x)$ from observations on x. The obvious precursor of a kernel density estimator is the histogram. The histogram divides the data range into nonoverlapping intervals, counts the number of data points that fall into each interval, and graphically depicts $f(x)$ by plotting side-by-side vertical bars spanning each interval with height proportional to the count (i.e., density) in the interval.

In true kernel density estimators, the range is still divided into intervals, and estimates of the density are made at the center of each interval. The key difference is that the intervals are now allowed to overlap. One can think of moving the interval (or *window* in kernel density terminology) along the data range with density estimates being gathered for each of the many consecutive overlapping windows. When plotted, these estimates will appear to be continuous and smooth, and will be independent of the choice of origin. An additional feature of more complex kernel density estimators is that, rather than merely counting the observations in each window, a weight is assigned to each observation that depends on its distance from the center of the window, and it is these weighted values that are summed and gathered. Stata has implemented a number of kernel density estimators and makes these available to users in the `kdensity` procedure; see [R] **kdensity**. These estimators differ only in that they use different weighting functions, and it is the weighting function itself that is called the kernel. Options `biweight`, `cosine`, ..., `triangle` can be invoked to specify the kernel in `violin`, but the default, `epan`, should be sufficient for most applications.

The density trace procedure described by Chambers et al. (and used by Hintze and Nelson in their seminal paper) is likely the simplest of the kernel density estimators: it assigns a weight of 1 to every point in the interval. It should be noted, though, that it is not one of the estimators provided by Stata's `kdensity` procedure. Because the `violin` program described in this insert uses Stata's `kdensity` procedure to generate the density estimates, it does not provide the Chambers procedure.

Nevertheless, it is generally agreed that the choice of kernel is not as important as the choice of window width (also called the *bandwidth*). A small bandwidth (narrow window) will result in a wiggly density curve, while a large bandwidth (wide window) will result in a very smooth density curve. By default, Stata's `kdensity` procedure computes an "optimal" bandwidth from the data. The user may, instead, choose to specify the bandwidth using option `width()`. This behavior is emulated in `violin` and intervention by the user is available via a similarly named option. It should be noted that the optimal bandwidth is the width that minimizes the mean integrated square error of Gaussian data when the `gaussian` kernel is used. It is not optimal in any global sense and it is believed to oversmooth the density for skewed or multimodal distributions. Specification of an appropriate bandwidth is critical in obtaining a density curve that properly describes the underlying character of the data. Oversmoothing of a variable from a small dataset may give the illusion of knowing the shape of the distribution when, in fact, too little data is available to properly describe it. In contrast, undersmoothing of the data may lead to apparent features in the density curve that are artifacts of the sample data and that are not present in the underlying distribution. To date, the experience of this author suggests that the default kernel and bandwidth generally lead to reasonable representations of real data, though I have tended to lean toward bandwidths that are slightly smaller than optimal for some data.

Examples

The ability of `violin` to properly describe the salient features of a variable can best be shown by presenting data with known distributions. Loosely following Hintze and Nelson's lead, the commands below generate 1,000 observations randomly drawn from three known distributions, each with approximately equal location and scale characteristics, as measured by the median and interquartile range.

```
. clear
. set obs 1000
. gen uniform = 20 * uniform() - 10
. gen normal = invnorm(uniform()) * sqrt(55)
. gen bimodal = invnorm(uniform()) * sqrt(3) - 5 if _n <= 500
. gen bimodal1 = invnorm(uniform()) * sqrt(3) + 5 if _n  > 500
. replace bimodal = bimodal1 if _n > 500
```

The first is a *uniform* distribution on the interval -10 to 10. The second is a *normal* distribution with mean 0 and variance 55. The third is a *bimodal* distribution composed of two normal distributions with means -5 and 5 and common variance 3. When these data are displayed using side-by-side box and violin plots (Figure 2), it is evident that the violin plots display significantly more information about the distributions than the box plots.

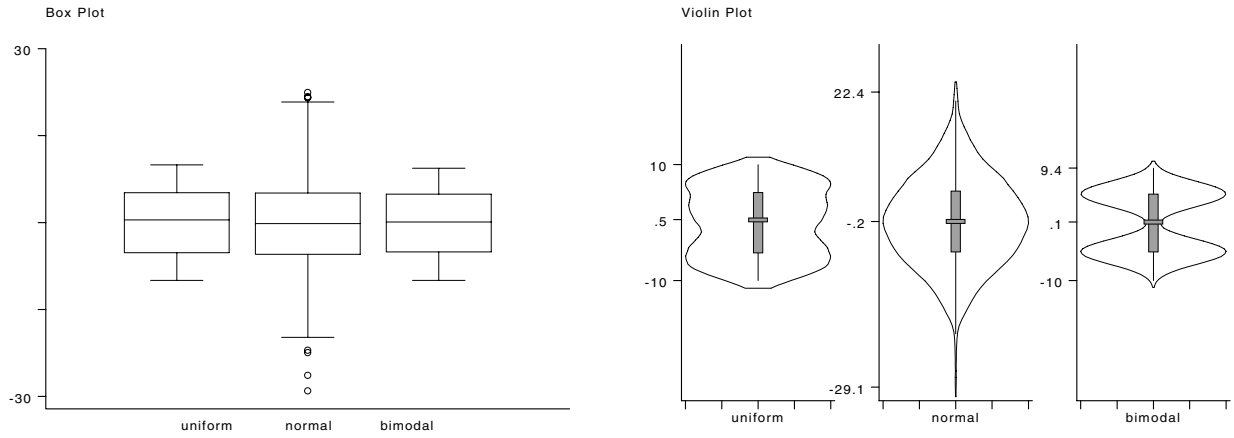

Figure 2. Comparison of known distributions.

Both graphs clearly show that the medians and interquartile range are effectively equal, and both show that the normal distribution has considerably greater range, but the box plots completely fail to distinguish the uniform from the bimodal distribution. This distinction is clearly shown by the violin plots, and the general shapes of all three distributions are readily apparent. The commands needed to create these plots (less the full `b2title` command) are shown below:

```
. graph uniform normal bimodal, box t1title(Box Plot) yscale(-30,30) b2title(...)
. violin uniform normal bimodal, yscale(-30,30) round(.1) gap(3)
```

Both commands allow multiple variables to be displayed side-by-side. While the `violin` command does not limit the number of variables that can be placed in a single graph, esthetic considerations suggest that no more than three or four variables should be displayed together. More than this results in very narrow plots with little density curve detail. With a large number of variables the plots start overwriting each other and become useless.

The example above demonstrates a few of the features of the `violin` command. First, most of the common options of `graph` can be specified with `violin`. Here we see the use of `yscale()` and `gap()` to control the display. `violin`'s default mode for a multi-variable *varlist* is to display each violin using a variable-specific range. The user may, instead, specify the `yscale()` option to set a common range to be used for all variables in *varlist*. When appropriate, this facilitates direct comparison of related variables. In addition, `violin` sets the gap for the y-axis tic mark labels to the number of significant digits needed to fully display the values. Here we use option `round()` to limit the displayed values to one significant digit and we specify `gap(3)` to reduce the amount of space used by the tic mark labels. We note that option `t1title()` was not specified in this `violin` command (although it could have been specified) and this resulted in the default title: `Violin Plot`. Likewise, because the `ylabel()` option was not specified, `violin` labeled each y-axis with relevant statistics, identifying the minimum, median, and maximum data values for each variable.

Two other options of `violin` are demonstrated below in Figure 3, again using the `mpg` variable from the auto data. First, `violin` has a `by(`*byvar*`)` option that, for a single variable, produces separate plots for the subgroups of observations defined by *byvar*. These plots will be displayed side-by-side in a single graph having common vertical scale. Note, though, that `by()` cannot be specified when there is more than one variable in *varlist*.

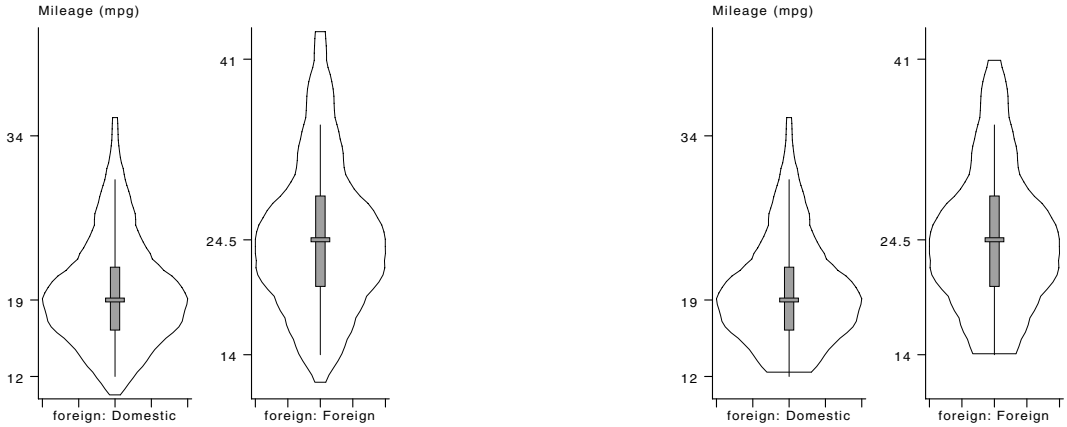

Figure 3. The `by()` and `truncat()` options.

Second, violin has a truncat() option to limit the range of the density curve, either to an interval specified as (#,#) or to the observed data limits, which can be specified as (*). Regardless of the actual (#,#) specification, the maximum range truncation honored is never less than the observed data limits, and the precise truncation points will be the most extreme values within the honored range where the density is actually calculated (the points of density calculation depend on n(), width() and the observed data). This option is useful in those situations where the underlying distribution has real range constraints and some observed points are near a range limit (for example, an age distribution where small ages are observed in the data). Because the first and last windows of the density curve extend beyond the data by half the window width, all such curves have ranges wider than the data range. Therefore, there will be some situations where the density curve suggests positive probability in a region where positive probability is impossible (for example, age less than zero). The truncat() option allows the user to avoid this situation by limiting the range for the density curve.

The graphs in Figure 3 show the results of using the by() and truncat() options. The left panel, which results from using by(foreign), shows the domestic and foreign subsets of the mpg variable in side-by-side violins plotted on a common scale. The right panel shows the changes resulting from addition of truncat(15,40). Please note the inappropriate use of the truncat option in this example (i.e., the request has specified limits, 15 and 40, that if honored would not include all observed data). The request was automatically changed to reset the truncation range to the actual data range, 12 to 41.

In addition to displaying a graph, violin lists basic descriptive statistics about the data and provides information about the density estimation. When by() is specified, this report is displayed for the combined group only. When multiple variables are included in *varlist*, the report is displayed for the last variable only. Shown below is the command and printed report for the graph in the right panel of Figure 3.

```
. violin mpg, by(foreign) truncat(15,40)
Statistics (all groups combined):
  Min: 11.483227  Q25: 18  Median: 20  Q75: 25  Max: 43.516773  n: 74
Densities computed using:
  Kernel: Epanechnikov  N: 50  Ave. Scale:   0.59  Ave. Width:   2.09
```

Because the by() option is specified, the report is for all groups combined. The report first shows the summary statistics, displaying the minimum (Min), the 25th centile (Q25), the median (Median), the 75th centile (Q75), the maximum (Max), and the sample size (n). Had by() not been specified, the report also would have included the lower and upper adjacent values (LAV and UAV, respectively). Second, the report includes information on the density estimation. This information includes the kernel method used (Kernel), the number of points of density estimation (N), and the resulting scale (Scale) and width (Width) factors. Because by() was specified in this example, the reported scale and width values are the averages of the scale and bandwidth factors used in the subgroup density estimations. The most useful of these reported results is the bandwidth factor, as it is often appropriate to try a bandwidth smaller than the optimal value.

Notes and acknowledgments

There are a few issues that a user of violin should consider. First, violin calls Stata's centile command to compute information needed for the box plot portion of the violin plot and, as noted before, uses a subset of Stata's kdensity code to compute the kernel density estimates. The concern is that centile does not respond to a [*weight*] specification, in contrast to the kdensity code, which responds to that specification. The implications of this conflict have not been explored, but violin currently allows a [*weight*] specification to be passed through to the kdensity code.

A second issue concerns a minor limitation imposed by Stata's gph clear command. This command, which is used in violin, exceeds Stata's release 5 gph format. As a result, neither Stage nor the gphdot or gphpen DOS-based graphics output programs can process a saved violin-plot graphics file. This limitation does not affect screen display or output using the Graph Print option of Stata's *File* menu. Some question also remains about how this command behaves on Macintosh computers.

Lastly, I wish to thank the many members of Statalist who kindly provided comments on the early drafts of violin. Your encouragement and thoughtful feedback helped create a better program.

Saved results

violin saves in the S_# macros:

S_1	name of kernel used for density trace	S_7	first quartile
S_2	number of points of density estimation	S_8	median
S_3	bandwidth for density estimation	S_9	third quartile
S_4	scale factor of density plot	S_10	upper adjacent value
S_5	minimum	S_11	maximum
S_6	lower adjacent value	S_12	n

When by() is specified, S_3 and S_4 contain the averages of the bandwidth and scale factors used in the subgroup density estimations; S_5, S_7, S_8, S_9, S_11 and S_12 are statistics for the combined group, and S_6 and S_10 are set missing.

When multiple variables are specified, the saved values contain results for the last variable in *varlist*.

References

Chambers, J. M., W. S. Cleveland, W. S. Kleiner, and P. A. Tukey. 1983. *Graphical Methods for Data Analysis.* Belmont, CA: Wadsworth.

Hintze, J. L. and R. D. Nelson. 1998. Violin plots: a box plot–density trace synergism. *The American Statistician* 52: 181–184.

sg89.2	Correction to the adjust command

Kenneth Higbee, Stata Corporation, khigbee@stata.com

The `adjust` command (Higbee 1998) has been fixed to allow the setting of covariates to negative numbers. Previously it would produce an error message if this was attempted.

References

Higbee, K. T. 1998. sg89: Adjusted predictions and probabilities after estimation. *Stata Technical Bulletin* 44: 30–37.

——. 1998. sg89.1: Correction to the adjust command. *Stata Technical Bulletin* 45: 23.

sg94	Right, left, and uncensored Poisson regression

Joseph Hilbe, Arizona State University, hilbe@asu.edu
Dean H. Judson, University of Nevada, djudson@unr.edu

Syntax

`cenpois` performs censored Poisson regression where the types of censoring can be left, right, or uncensored. Its syntax is

`cenpois` *depvar varlist* [*weight*] [`if` *exp*] [`in` *range*] [`,` `level(#)` `ltolerance(#)` `irr`

`iterate(#)` `censor(`*cvar*`)` `offset(`*ovar*`)` `nolog`]

`fweights` and `aweights` are allowed. Issuing `cenpois` by itself replays the most recent occurrence of the command.

Options

`level(#)` specifies the level, in percent, for confidence intervals. The default is `level(95)` or as set by `set level`.

`ltolerance(#)` specifies the convergence criterion and defaults to 0.01.

`irr` reports estimated coefficients transformed to exponential form; standard errors and confidence intervals are adjusted accordingly.

`iterate(#)` limits the number of iterations to #. This allows you to stop the iterations and view results (with a warning that the procedure has not converged). The default is 1000 (effectively infinite).

`censor(`*cvar*`)` specifies the variable that indicates censoring. 1 indicates not censored, 0 indicates left censored, and −1 indicates right censored.

`offset(`*ovar*`)` specifies the offset variable. Remember to log the raw variable prior to using it as an offset.

`nolog` suppresses display of the iteration log.

Theory

`cenpois` is an extension of the Poisson regression command to handle censored data. It would be appropriate to use `cenpois` when 1) the dependent variable is a nonnegative count, 2) the distribution of counts (conditional on the independent variables) can be reasonably considered Poisson, and 3) some, but not all, of the cases are censored either above or below.

A typical example of censoring occurs in weeks of unemployment insurance (UI) receipt. UI recipients can receive from one to 13 weeks of UI coverage, but their coverage terminates at 13 weeks. Thus, the cases are censored to the right at 13 (anyone who would have received UI coverage for a longer period is terminated at 13), and censored to the left at one (anyone who would have received UI coverage for greater than zero but less than one week is not recorded).

If all cases are censored, the equation cannot be estimated.

Poisson regression is one of the class of generalized linear models with

1. A Poisson distributed dependent variable, $Y \sim \text{Poisson}(u)$,
2. A log-link between dependent and independent variables, i.e., $\ln u = XB$.

This program uses Stata's `ml` commands to implement right, left and uncensored data in the Poisson regression framework; hence the name censored Poisson regression. In this case, we are referring to Type II censored sampling as it is defined in Bain and Engelhardt (1987, 164–167); the number of observations is considered fixed, but the length of the experiment is a random variable.

The likelihood function, ignoring constants, is

$$L(u,X) = \prod_{i=1}^{N} f(x_i,u)^{I(p_i=1)} \left(\sum_{j=0}^{x_i} f(j,u)\right)^{I(p_i=0)} \left(1 - \sum_{j=0}^{x_i} f(j,u)\right)^{I(p_i=-1)}$$

where

- N is the number of cases,
- $p_i = 1$ if the ith observation is not censored, 0 if left censored, -1 if right censored,
- $I(p_i)$ is the indicator function, taking the value one when the statement in parentheses is true, otherwise taking the value 0,
- f is the probability density function of a Poisson random variable with parameter u,
- $u = \exp(X\beta)$,
- $1 - \sum_{j=0}^{x_i} f(j;u)$ is the probability of observing x_i or more events when $E(Y) = u$,
- $\sum_{j=0}^{x_i} f(j;u)$ is the probability of observing x_i or fewer events when $E(Y) = u$.

Example

Censoring can occur in many contexts; the most familiar context is where the experiment ends before an event occurs. We use Stata's cancer data to illustrate:

```
. use /usr/local/stata/cancer,replace
(Patient Survival in Drug Trial)
. tabulate drug, generate(drug)
Drug type   |
(1=placebo) |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         20       41.67       41.67
          2 |         14       29.17       70.83
          3 |         14       29.17      100.00
------------+-----------------------------------
      Total |         48      100.00
```

We have generated the dummy variable `drug` for the three types of treatment; namely the values 1, 2, and 3 for placebo, first drug test, and second drug test, respectively. The variable `age` is the age of the patient in years at the beginning of the study. Standard poisson regression does not take into account that some of the cases are censored (that is, the study ended before they died):

```
. poisson studytim drug2 drug3 age
Iteration 0: Log Likelihood = -197.98743
Iteration 1: Log Likelihood = -188.7749
Iteration 2: Log Likelihood = -188.69067
Poisson regression                                Number of obs    =      48
Goodness-of-fit chi2(44)    =   170.455           Model chi2(3)    = 160.016
Prob > chi2                 =    0.0000           Prob > chi2      =  0.0000
Log Likelihood              =  -188.691           Pseudo R2        =  0.2978
------------------------------------------------------------------------------
studytim |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   drug2 |   .5274627   .1017334      5.185   0.000       .3280688    .7268566
   drug3 |   .9923626    .091891     10.799   0.000       .8122596    1.172466
     age |  -.0325103   .0070028     -4.642   0.000      -.0462356    -.018785
   _cons |    4.00428   .3931972     10.184   0.000       3.233628    4.774932
------------------------------------------------------------------------------
```

However, in many cases the experiment ended before the patient died. These cases are censored on the right, or `died` = 0. Censored poisson regression provides estimates for this type of dataset:

```
. cenpois studytim drug2 drug3 age,censor(died)
Iteration 0:  Log Likelihood = -1610.8409
(unproductive step attempted)
Iteration 1:  Log Likelihood = -259.02719
Iteration 2:  Log Likelihood =  -145.3059
Iteration 3:  Log Likelihood = -143.18667
Iteration 4:  Log Likelihood = -143.18493
Iteration 5:  Log Likelihood = -143.18493
Censored Poisson Estimates                      Number of obs    =      48
                                                Model chi2(3)    =  185.77
                                                Prob > chi2      =  0.0000
Log Likelihood =   -143.1849300                 Pseudo R2        =  0.3935
------------------------------------------------------------------------------
studytim |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   drug2 |   .7798293   .1127562      6.916   0.000       .5588313    1.000827
   drug3 |   1.124823   .0980459     11.472   0.000       .9326566    1.316989
     age |  -.0452013   .0083118     -5.438   0.000       -.061492   -.0289105
   _cons |   4.713316   .4609625     10.225   0.000       3.809846    5.616786
------------------------------------------------------------------------------
```

Censored poisson will also generate incidence rate ratios:

```
. cenpois studytim drug2 drug3 age,censor(died) irr
Iteration 0:  Log Likelihood = -1610.8409
(unproductive step attempted)
Iteration 1:  Log Likelihood = -259.02719
Iteration 2:  Log Likelihood =  -145.3059
Iteration 3:  Log Likelihood = -143.18667
Iteration 4:  Log Likelihood = -143.18493
Iteration 5:  Log Likelihood = -143.18493
Censored Poisson Estimates                      Number of obs    =      48
                                                Model chi2(3)    =  185.77
                                                Prob > chi2      =  0.0000
Log Likelihood =   -143.1849300                 Pseudo R2        =  0.3935
------------------------------------------------------------------------------
studytim |     IRR    Std. Err.      z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
drug2    |   2.1811   .2459325     6.916   0.000       1.748628    2.720532
drug3    |   3.079672 .3019492    11.472   0.000       2.541251    3.732169
age      |   .9558051 .0079444    -5.438   0.000       .9403605    .9715034
------------------------------------------------------------------------------
```

Thus, as expected, patients survive twice as long on the second drug as on the placebo and three times as long on the third. Also, older patients survive shorter times in each case, as expected.

References

Bain, L. J. and M. Engelhardt. 1987. *Introduction to Probability and Mathematical Statistics.* Boston, MA: Duxbury Press.

sg95 | Geographically weighted regression : A method for exploring spatial nonstationarity

Mark S. Pearce, University of Newcastle upon Tyne, UK, m.s.pearce@ncl.ac.uk

The syntax for the gwr commands is

`gwr` *varlist* [`if` *exp*] [`in` *range*] , `east(`*varname*`) north(`*varname*`)` [*options*]

`gwrgrid` *varlist* [`if` *exp*] [`in` *range*] , `east(`*varname*`) north(`*varname*`)` [*options*]

where `east()` and `north()` are required and the allowed *options* are

`test` `bandwidth(#) sample(#) saving(`*filename*`) outfile(`*filename*`)`

`mcsave(`*filename*`) replace double nolog iterate(#) reps(#) dots`

with the additional option `square(#)` for `gwrgrid`.

gwr uses iweights. For this reason, while the default model (i.e., a linear regression model) actually uses regress (for speed), all other models use the glm command. Where gwr uses glm, many of the options used with glm are also available for gwr, enabling the user to define the form of model. These glm options are: eform, family, link, [ln]offset noconstant, scale, disp, and init. Not specifying family, link or offset automatically results in the default linear regression model being used.

Description

The gwr command applies geographically weighted regression to a dataset containing geographical reference points. These points must be defined in the gwr command using the east() and north() "options." The gwrgrid command is a slight variation which fits a grid across the area defined by east() and north(). This method is especially useful for large datasets where the gwr command can become quite time consuming.

The essence of geographically weighted regression is that it allows different relationships between the dependent and independent variables to exist at different points, (x, y), in space. For a full discussion of this method see Brunsdon et al. (1996).

The gwr command produces a set of parameter estimates for a regression at each point, i, at which there is an observation. The gwrgrid command produces a set of parameter estimates at each grid square centroid. Any grid square in which there are no observations is ignored. Using the outfile() and saving() options, these estimates can be explored in more detail, for example using a geographical information system to map risk estimates across space. In each regression the observations are weighted by a function related to their distance from the point at which the regression is being carried out. The actual function used depends upon a bandwidth estimation which is carried out using a cross-validation approach. See, for example, Cleveland (1979) and Bowman (1984).

The command is designed to test two hypotheses, both using a Monte Carlo simulation where the spatial points are randomly distributed amongst the data:

1. Does the geographically weighted regression model describe the data significantly better than a global regression model, the results of which are also shown in the output?

2. Does the set of parameter estimates exhibit significant spatial variation? This compares the standard deviations of the observed parameter estimates (S_i) with those from the Monte Carlo simulation.

Options

test requests that the first hypothesis be tested, i.e., that the model produced by gwr describes the data significantly better than a global model. Not testing this hypothesis reduces the need to calibrate the bandwidth for each run of the Monte Carlo simulation and so reduces the time the command will take to run. The second hypothesis, concerning spatial variation of the parameter estimates, is always carried out, using either the user-defined bandwidth or the bandwidth estimated from the observed data. If test is specified and convergence of the bandwidth not achieved during the Monte Carlo simulation, a note is made in the results of how many times convergence was not achieved. The significance level is adjusted to ignore those runs where convergence was not achieved.

bandwidth(#) allows the bandwidth to be declared, eliminating the need to calibrate the bandwidth and so saving time.

sample(#) specifies the percentage of observations to be used in the bandwidth calibration process, the default being 100%. If this option is specified, #% of the observations will be randomly sampled and used in the calibration process. The same number of observations will be used when recalibrating the bandwidth for the simulation test of the geographically weighted regression model.

saving(*filename*) specifies the name of the file to contain the parameter estimates and grid reference for each point at which the regression is carried out.

outfile(*filename*) creates a text file filename.raw containing the parameter estimates from each point at which gwr is calculated. The comma and wide options for outfile() are also allowed. The file is set out as *east*, *north*, *dep_vars*, *constant*. outfile() and saving() can be specified simultaneously.

mcsave(*filename*) requests that the results of the Monte Carlo simulation be saved as filename.dta rather than using a temporary file. This file will contain the standard errors of the parameter estimates for each run, as well as the simulated bandwidths if the test option is specified. A simulated bandwidth of -99.99 indicates that the bandwidth calibration failed to converge.

replace indicates that the filenames specified by saving(), outfile() or mcsave() may be overwritten.

double specifies that the results stored in the file specified by saving(), outfile() or mcsave() are stored as doubles (8-byte reals). By default they are stored as floats (4-byte reals).

`nolog` suppresses the display of the bandwidth calibration process.

`iterate(#)` specifies the maximum number of iterations allowed in estimating the bandwidth. The default is 50.

`reps(#)` specifies the number of Monte Carlo simulations to be performed. The default is 1000.

`dots` requests a dot be placed on the screen at the beginning of each run of the Monte Carlo simulation, showing how far the simulation has gone.

`square(#)` is only used with `gwrgrid` and allows the size of the grid squares to be defined. The default is to set the width of the grid squares to be half the bandwidth.

Example

This example uses the ward-level 1991 census data for the county of Tyne and Wear in the United Kingdom, and explores the relationship between the number of cars per 100 household (`cars1`) and 2 independent variables, the proportion of unemployed males (out of the total economically active male population) (`unemp`) and social class (the proportion of households with the head of the household in social class I) (`class`). The spatial points are given by the variables `east` and `north`, in this case they are the eastings and northings of the ward centroids in kilometers.

Using this dataset, both hypotheses can be tested, in this example using a Monte Carlo simulation of 1000 repetitions.

```
. gwr cars1 class unemp, east(east) north(north) reps(1000) dots test saving(gwrout) replace nolog
Global Model
  Source |       SS       df       MS                  Number of obs =     120
---------+------------------------------                F(  2,   117) =  287.17
   Model |  45196.5848     2    22598.2924              Prob > F      =  0.0000
Residual |  9207.00732   117    78.6923702              R-squared     =  0.8308
---------+------------------------------                Adj R-squared =  0.8279
   Total |  54403.5921   119    457.173043              Root MSE      =  8.8709

------------------------------------------------------------------------------
   cars1 |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   class |   1.880726   .3344889      5.623   0.000       1.218289    2.543164
   unemp |  -1.827983   .1123766    -16.267   0.000      -2.050539   -1.605427
   _cons |   88.47704   2.885689     30.661   0.000       82.76208      94.192
------------------------------------------------------------------------------

Convergence : Bandwidth = 4829.2874

Running Monte Carlo simulation
  (output omitted)
Geographically Weighted Regression

Significance Test for Bandwidth

----------------------------------------
Observed           P-Value
----------------------------------------
4829.2874          0.010
----------------------------------------

Significance Tests for Non-Stationarity

------------------------------------------------------------
Variable         Si                 P-Value
------------------------------------------------------------
Constant         6.0421             0.568
class            1.6079             0.023
unemp            0.1573             0.930
------------------------------------------------------------
```

The global model shows that the number of cars per household is significantly related to social class and male unemployment in the study region. The test of the bandwidth suggests that the geographically weighted regression model is a significantly better model for these data than the global linear regression model. The significance tests for nonstationarity of the parameter estimates show that the relationship between the number of cars per household and social class varies significantly over the study area. Mapping the estimates in the file `gwrout.dta` in ARC-VIEW gives Figure 1, which suggests that the relationship between car ownership and social class was stronger in areas not particularly well served by the region's light railway mass transit system which has its focus on the city of Newcastle upon Tyne. Mapping the male unemployment parameter suggested that the relationship was less marked in certain areas, the cause of which wasn't immediately obvious, thus suggesting an area of further investigation.

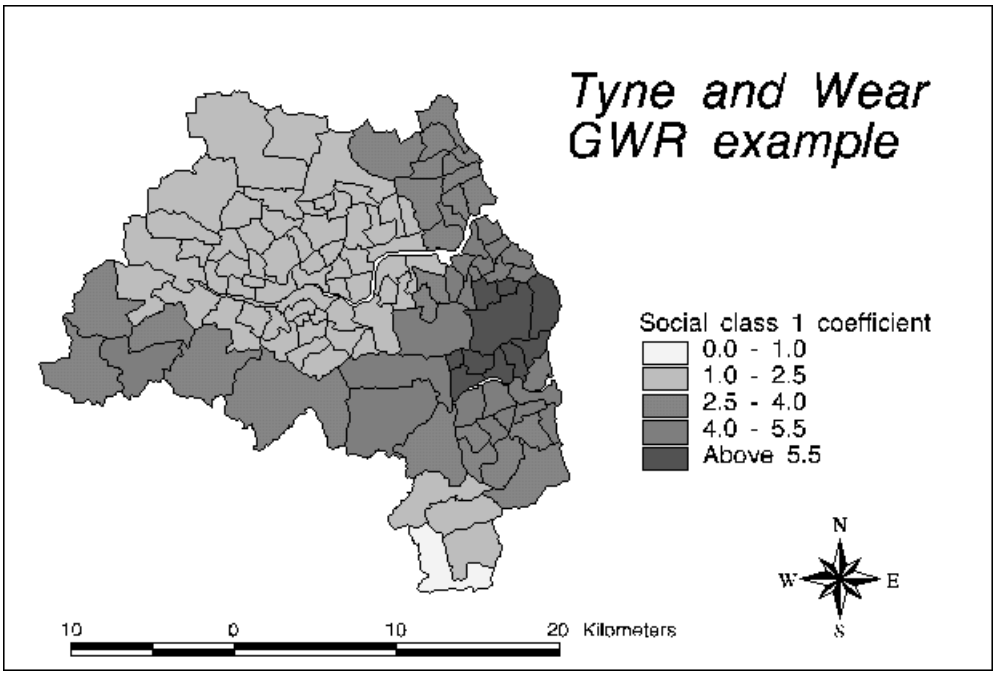

Figure 1. A map of the values produced by `gwr`.

Methods and formulas

The basic idea of geographically weighted regression is that a regression model is fitted at each point, i, weighting all observations, j, by a function of distance from that point. Hence observations sampled near to the observation where the regression is centered have more influence on the resulting regression parameters at that point than observations further away. This then produces a set of parameter estimates for a regression at each point in space.

The weighting function used by `gwr` takes the form

$$W_j = \exp(-d_j/\beta^2)$$

where d_j is the distance from the point i at which the regression model is being fitted, and β is the bandwidth. This was the weighting function used by Brunsdon et al. If a different weighting function is desired, the `gwr` ado-file can easily be altered.

The bandwidth is calibrated by a cross-validation technique which aims to minimize the score

$$\sum_{i=1}^{n} \left(y_i - \hat{y}_{\neq i}(\beta)\right)^2$$

where $\widehat{y}_{\neq i}(\beta)$ is the fitted value of y_i using the bandwidth β and the weighted regression model centered at the point i, with the observation for point i excluded from the calibration process.

Once the bandwidth calibration process is complete, `gwr` uses the optimal bandwidth to fit a weighted regression model at each point, the parameter estimates being output to either a temporary or permanent datafile depending on the options used. Outputting the parameter estimates to a permanent data file (in Stata and/or text format) allows further investigation, for example using a geographical information system.

The two hypotheses previously described are tested by a Monte Carlo simulation, although the significance of the `gwr` approach as compared to a global regression model is only tested if specifically requested by the `test` option. Each run of the Monte Carlo simulation randomly distributes the spatial points across the observations, and the `gwr` process is repeated. If `test` is specified, the simulated bandwidths are compared with that calibrated using the observed data. The second hypothesis is tested by comparing the standard error of the parameter estimates from the observed data with those from each run of the Monte Carlo simulation.

Acknowledgments

I thank Brunsdon et al. for the use of their census data and original Fortran program which can be obtained from their website `http://www.ncl.ac.uk/~ngeog/GWR/`. I also thank Dr. Heather Dickinson, Dr. Chris Brunsdon, Mr. Martin Charlton,

and Mr. Trevor Dummer, University of Newcastle for their useful comments in the formulation of these commands. This work was also helped by comments made at the 4th Stata UK User Group Meeting in May of 1998.

References

Bowman A. W. 1984. An alternative method of cross-validation for the smoothing of density estimates. *Biometrika* 71: 353–60.

Brunsdon C., A. S. Fotheringham, and M. E. Charlton. 1996. Geographically weighted regression: a method for exploring spatial nonstationarity. *Geographical Analysis* 28: 281–98.

Cleveland W. S. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74: 829–36.

sg96	Zero-inflated Poisson and negative binomial regression models

Jesper B. Sørensen, University of Chicago, jesper.sorensen@gsbpop.uchicago.edu

Background

Perhaps the most popular way to estimate models for count data involves the use of Poisson regression and negative binomial regression models. This insert describes programs for estimating "zero-inflated" Poisson and negative binomial regression models (Lambert 1992, Long 1997, Greene 1997). These models can be motivated both empirically, in terms of the failure of the Poisson and negative binomial models to adequately fit the data, or theoretically, in terms of one's understanding of the processes generating the observed counts.

The negative binomial model is typically motivated as a modification of the Poisson regression model that relaxes the assumption that the variance of the observed random variable is equal to its mean. It is often observed, for example, that empirical distributions have substantially more zero counts than predicted by the Poisson distribution (see, for example, Long 1997). These "excess zeros" are a source of overdispersion. The negative binomial model takes this overdispersion into account by allowing the conditional variance to increase, without changing the conditional mean.

Zero-inflated models, by contrast, allow for excess zeros by assuming that zero counts are generated by a process different from that generating positive counts. Consider, for example, a hypothetical question asking Stata users how often they visit the Stata web site. For users without internet access, the count will always be zero. For those with internet access, the number may be positive, but may also be zero. A Poisson or negative binomial model applied to such data incorrectly assumes that all individuals are at risk of visiting the site.

In the Zero-Inflated Poisson (ZIP) and Zero-Inflated Negative Binomial (ZINB) models, the population is assumed to consist of two groups. A person belongs to group A with probability η and is in group B with probability $1-\eta$. People in group A always have zero counts (in our example, because they do not have internet access). In the second group, counts are generated either by a Poisson process or a negative binomial process.

The probability η can be modeled as $\eta_i = F(z_i\gamma)$, where z is a set of covariates predicting group membership. F can be either the normal or the logistic cumulative distribution function. In the current insert, η_i is modeled using logistic regression.

The ZIP model is

$$\Pr(y_i|x_i) = \begin{cases} \eta_i + (1-\eta_i)\exp(-\mu_i) & \text{for } y_i = 0 \\ (1-\eta_i)\dfrac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!} & \text{for } y_i > 0 \end{cases}$$

where $\mu = \exp(x\beta)$ with x a matrix of covariates. It is apparent that zero counts can come about either through membership in group A (no internet access) or through the Poisson process operating among members of Group B. (Note that $\exp(-\mu_i)$ is the probability of a zero count in the Poisson model.)

Similarly, the ZINB model is

$$\Pr(y_i|x_i) = \begin{cases} \eta_i + (1-\eta_i)\left(\dfrac{1}{1+\alpha\mu}\right)^{1/\alpha} & \text{for } y_i = 0 \\ (1-\eta_i)\dfrac{\Gamma(y_i+1/\alpha)}{y_i!\Gamma(1/\alpha)}\left(\dfrac{1/\alpha}{1/\alpha+\mu_i}\right)^{1/\alpha}\left(\dfrac{\mu_i}{1/\alpha+\mu_i}\right)^{y_i} & \text{for } y_i > 0 \end{cases}$$

where $\mu = \exp(x\beta)$ with x a matrix of covariates. Here, α is a dispersion parameter to be estimated.

These models are fitted using maximum likelihood. For the ZIP model, the maximum likelihood is written as

$$L(\beta,\gamma|y,X,Z) = \sum_{i=1}^{N}\Pr(y_i|x_i,z_i)$$

By substitution, $\Pr(y_i|x_i, z_i)$ for the ZIP model is defined as

$$\Pr(y_i|x_i,z_i) = \begin{cases} F(z_i\gamma) + [1 - F(z_i\gamma)]\exp(-\exp(x_i\beta)) & \text{for } y_i = 0 \\ [1 - F(z_i\gamma)]\,\dfrac{\exp(-\exp(x_i\beta))\exp(x_i\beta)^{y_i}}{y_i!} & \text{for } y_i > 0 \end{cases}$$

Likelihoods for the ZINB are constructed by substituting the appropriate elements. Both the ZIP and ZINB are estimated using Stata's maximum likelihood routines (`ml` method `lf`).

Greene (1994) shows that for both the ZIP and the ZINB models, the conditional expected values are given by

$$E(y_i|x_i,z_i) = [0 \times \eta_i] + [\mu_i \times (1 - \eta_i)] = \mu_i - \mu_i\eta_i$$

Under the ZIP and ZINB models, the predicted probability of different counts are given by substituting the appropriate predicted values into the model definitions above.

Syntax

`zipois` *depvar* [*indepvars*] [`if` *exp*] [`in` *range*] [, `logit(`*varlist*`)` `level(#)` `trace` `nolog` `nobase`]

`zinbreg` *depvar* [*indepvars*] [`if` *exp*] [`in` *range*] [, `logit(`*varlist*`)` `level(#)` `trace` `nolog` `nobase`]

`zippred` *varlist* [, { `xb` | `prob` }]

`zipois` and `zinbreg` share most features of estimation commands; see [U] **26 Estimation and post-estimation commands**. However, to obtain predictions from the models, use `zippred` instead of `predict`.

Options for zipois and zinbreg

`logit(`*varlist*`)` specifies the logit model for predicting membership in the zero-count regime. If *varlist* is not specified, the same model is used as for the count model.

`level(#)` specifies the confidence level, in percent, for confidence intervals. The default is `level(95)` or as set by `set level`.

`trace` requests that coefficients be listed at each iteration so that one can study their convergence.

`nolog` suppresses printing of the maximum likelihood iteration details.

`nobase` suppresses estimation of the constant-only baseline model. This is useful if convergence problems are encountered in estimating the baseline model.

Options for zippred

`xb` requests the predicted count as defined above. One variable should be specified in *varlist*. This is the default.

`prob` requests the predicted probabilities of different counts. The user should specify $J + 1$ variables to obtain predicted counts for the integers from 0 to J. The first variable will contain the predicted probability, for each observation, of a zero count; the second will contain the probability of a count of one; and so on.

Example: Derogatory credit reports

To illustrate the use of the ZIP and ZINB models, we use data reported in Greene (1997, 470–471). The dependent variable of interest is the number of major derogatory credit reports recorded in the credit history of a sample of applicants for a credit card. A major derogatory report is defined as a delinquency of sixty days or more on a credit account. As predictors, we include reported income, their average monthly credit card expenditures, age, and a dummy variable indicating whether they own their home. There are 100 cases; these are a sample of a larger dataset. The observed counts look like this:

```
. tab derog
      derog |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         82       82.00       82.00
          1 |         10       10.00       92.00
          2 |          3        3.00       95.00
          3 |          3        3.00       98.00
          4 |          1        1.00       99.00
          7 |          1        1.00      100.00
------------+-----------------------------------
```

First, we fit a Poisson model to the data.

```
. poisson derog income avgexp ownhome age
Iteration 0: Log Likelihood = -127.42374
Iteration 1: Log Likelihood = -89.294613
Iteration 2: Log Likelihood = -81.146847
Iteration 3: Log Likelihood = -79.661957
Iteration 4: Log Likelihood = -79.509777
Iteration 5: Log Likelihood = -79.50668
Poisson regression                                Number of obs    =     100
Goodness-of-fit chi2(95)    =   115.123           Model chi2(4)    =  24.861
Prob > chi2                 =    0.0785           Prob > chi2      =  0.0001
Log Likelihood              =   -79.507           Pseudo R2        =  0.1352
------------------------------------------------------------------------------
   derog |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  income |   .2195479   .1026526      2.139   0.032       .0183526    .4207432
  avgexp |  -.0068032   .0019248     -3.534   0.000      -.0105758   -.0030306
 ownhome |   .2288366   .3580642      0.639   0.523      -.4729564    .9306296
     age |   .0094613   .0203021      0.466   0.641      -.0303301    .0492528
   _cons |  -1.525245   .7214558     -2.114   0.035      -2.939272   -.1112173
------------------------------------------------------------------------------
```

We see that income and average expenditures are significant predictors of the number of derogatory reports. However, we are concerned about the presence of overdispersion, particularly given the large number of zero counts observed. We can compute the mean predicted probability of a zero count from the Poisson model.

```
. predict yh
. gen pzero=exp(-exp(yh))
. sum pzero
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+--------------------------------------------------------
   pzero |     100    .7267382    .186909   .1445756   .9999979
```

The average probability of a zero count predicted by the Poisson model is 73%, which does not compare favorably to the observed 82% zero counts. This suggests overdispersion. We can test for this by estimating a negative binomial model.

```
. nbreg derog income avgexp ownhome age
  (output omitted)
Negative Binomial Regression                      Number of obs    =     100
                                                  Model chi2(4)    =    9.46
                                                  Prob > chi2      =  0.0505
Log Likelihood =    -67.2326207                   Pseudo R2        =  0.0658
------------------------------------------------------------------------------
   derog |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
_lnmean  |
  income |    .238453   .2090309      1.141   0.254      -.1712401    .6481461
  avgexp |  -.0056734   .0022229     -2.552   0.011      -.0100302   -.0013165
 ownhome |   .1025945   .5817071      0.176   0.860      -1.037531     1.24272
     age |   .0151784   .0347696      0.437   0.662      -.0529687    .0833256
   _cons |  -1.824648   1.239168     -1.472   0.141      -4.253373    .6040775
---------+--------------------------------------------------------------------
_lnalpha |
   _cons |   1.195215   .4665744      2.562   0.010       .2807457    2.109684
------------------------------------------------------------------------------
   alpha     3.304267   [_lnalpha]_cons = ln(alpha)
                  (LR test against Poisson, chi2(1) =  24.54812 P =  0.0000)
```

Our suspicion is correct, as the likelihood ratio contrast with the nested Poisson model indicates. There is overdispersion in the data. However, this overdispersion may be largely due to excess zeros, since the majority of individuals do not enter into credit delinquency. Instead of the negative binomial model, a more appealing way of modeling this outcome may be to approach it as a split population outcome using the ZIP model. Here, we use income and the home ownership dummy variable to predict to which regime a person belongs.

```
. zipois derog income avgexp ownhome age, logit(income ownhome)
  (output omitted)
Iteration 5:  Log Likelihood = -64.660637
Iteration 6:  Log Likelihood = -64.660631
```

```
Iteration 7:  Log Likelihood = -64.660631
Zero Inflated Poisson                             Number of obs    =     100
                                                  Model chi2(6)    =   18.91
                                                  Prob > chi2      =  0.0043
Log Likelihood =    -64.6606309
------------------------------------------------------------------------------
   derog |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
poisson  |
  income |   -.071043   .1335774     -0.532   0.595        -.33285    .1907639
  avgexp |  -.0059627   .0019165     -3.111   0.002       -.009719   -.0022064
 ownhome |   .4315482   .4336339      0.995   0.320      -.4183585    1.281455
     age |  -.0004888   .0200497     -0.024   0.981      -.0397854    .0388078
   _cons |    .859137   .9102744      0.944   0.345      -.9249681    2.643242
---------+--------------------------------------------------------------------
logit    |
  income |  -.7318381   .3703119     -1.976   0.048      -1.457636   -.0060401
 ownhome |   .6870151   .8164723      0.841   0.400      -.9132412    2.287272
   _cons |   2.943077   1.152284      2.554   0.011       .6846424    5.201512
------------------------------------------------------------------------------
```

The parameters in the logit equation are predictors of the probability that an individual will belong to the zero count regime. Thus it appears that income lowers the odds that an individual will belong to the zero count regime.

We can use `zippred` to calculate predicted probabilities of different counts, given the model estimated. We generate predicted probabilities of each of the integers between 0 and 7, and then use summarize to compute the mean probability for each count.

```
. zippred v0-v7, prob
. summarize v0-v7
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+--------------------------------------------------------
      v0 |     100     .823347   .1273081   .3069071   .9999872
      v1 |     100    .0813874   .0558434   .0000128   .3422058
      v2 |     100    .0494619   .0425542   1.29e-10   .2185565
      v3 |     100    .0263504   .0267785   8.62e-16   .1051895
      v4 |     100    .0120506   .0142212   4.33e-21   .0666094
      v5 |     100    .0048396   .0067138   1.74e-26   .0337434
      v6 |     100    .0017462   .0028507   5.81e-32    .014245
      v7 |     100    .0005755   .0010876          0   .0051545
```

This model appears to do a good job of fitting the data, particularly the zero counts.

We may still be concerned about overdispersion. The ZINB model and the ZIP model are nested, so we can test for overdispersion using the chi-square contrast between the two models. First we estimate the ZINB model.

```
. zinb derog income avgexp ownhome age, logit(income ownhome)
  (output omitted)
Iteration 37:  Log Likelihood = -64.389963
Iteration 38:  Log Likelihood = -64.389963
Zero Inflated Negative Binomial                   Number of obs    =     100
                                                  Model chi2(6)    =   15.14
                                                  Prob > chi2      =  0.0192
Log Likelihood =    -64.3899627
------------------------------------------------------------------------------
   derog |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
nb       |
  income |  -.0639252   .1506118     -0.424   0.671      -.3591189    .2312686
  avgexp |  -.0058264    .001965     -2.965   0.003      -.0096778    -.001975
 ownhome |   .3963029   .5399383      0.734   0.463      -.6619568    1.454563
     age |   .0005707   .0253571      0.023   0.982      -.0491283    .0502696
   _cons |   .6891987   1.173421      0.587   0.557      -1.610664    2.989062
---------+--------------------------------------------------------------------
logit    |
  income |  -.7860776   .4395701     -1.788   0.074      -1.647619    .0754639
 ownhome |   .7126939   .9118111      0.782   0.434      -1.074423    2.499811
   _cons |   2.926294   1.302699      2.246   0.025        .373052    5.479537
---------+--------------------------------------------------------------------
lnalpha  |
   _cons |  -1.522253   2.040672     -0.746   0.456      -5.521895     2.47739
------------------------------------------------------------------------------
```

As we can see, the log likelihoods for the ZIP and ZINB models are virtually identical in this case. Thus, after taking into account the split population character of the data, there does not seem to be overdispersion in the data.

Acknowledgment

I am grateful to Scott Long for spotting a problem in an early version of this program.

References

Greene, W. H. 1994. Accounting for excess zeros and sample selection in Poisson and negative binomial regression models. Working Paper EC-94-10, Stern School of Business, New York University.

——. 1997. *Econometric Analysis.* 3d ed. Saddle River, NJ: Prentice–Hall.

Lambert, D. 1992. Zero-inflated Poisson regression, with an application to defects in manufacturing. *Technometrics* 34: 1–14.

Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables.* Thousand Oaks, CA: Sage.

sg97	Formatting regression output for published tables

John Luke Gallup, Harvard University, jgallup@hiid.harvard.edu

I have always wanted my statistical package to turn my regressions into neat tables suitable for publication: t statistics or standard errors under coefficients, asterisks for significance levels, variable descriptions, and R-squared at the bottom. Stata does not, so I have worked it out for myself. This command saves time when I need it most—when I have left out a regressor the night before the paper is due, and must update all the regression results.

The `outreg` command described in this insert takes output from any estimation command in Stata and formats it as in journal articles. One of the most useful features of `outreg` is that successive estimation results, possibly with different variables, can be automatically appended with the variable coefficients lined up properly. Multivariate regression commands like `sureg`, `mvreg`, and `reg3` are formatted equation-by-equation, and can be appended. There are plenty of options, such as a choice between t statistics or standard errors, the number of decimals displayed for the coefficients, and the choice of excluding the number of observations or explanatory notes at the bottom of the table. By default, the output file uses the variable labels rather than the eight character (or fewer) variable names, so the presentation tables are in intelligible English if appropriate variable labels have been assigned.

`outreg` creates a text (ASCII) file with columns separated with tab characters. If you print out the text itself, it is not pretty, but this file format can be converted automatically to a table in word processors and spreadsheets like Microsoft Word and Excel. The sequence in Microsoft Word 97 is simple. First, open or insert the file created by `outreg`. Then select the estimation output text that is in columns (not the notes at the bottom of the table), and choose the "table" and "convert text to table" items in the menus. With some adjustment of the column widths, fonts, etc. the final table is ready. In Excel 97, one simply opens the file created by `outreg` and follows the default choices in the Text Import Wizard. Other software should be similar.

Syntax

The command `outreg` writes formatted regression output to a text file. It may be used after any estimation command.

`outreg` [*varlist*] `using` *filename* [`, se nolabel bdec(#) tdec(#) noparen noaster nocons`

`nonobs noni nor2 adjr2 nonotes comma append replace`]

Description

`outreg` formats regression output as it is presented in most documents: t statistics or standard errors in parentheses under each coefficient, asterisks indicating coefficients statistically different from zero, and summary statistics like R-squared at the bottom. The formatted output is written to a tab- or comma-separated ASCII file, which can then be loaded into word processing or spreadsheet programs to be converted to a table.

`outreg` works after any Stata estimation command. In addition to coefficient estimates, it will report number of observations, true R-squareds, the number of groups in panel estimation, and indicate whether robust heteroscedasticity-consistent errors or t statistics are being reported.

If *filename* is specified without an extension, `.out` is assumed. Several regressions with differing independent variables can be combined into a single table with the `append` option.

If a *varlist* is specified, only the regression coefficients corresponding to the variables in *varlist* will be included in the table, except that the intercept coefficient is included unless the `nocons` option is chosen.

Options

`se` specifies that standard errors rather than t statistics are reported.

`nolabel` specifies that variable names rather than variable labels (where they exist) be used to identify coefficients. Note that this differs from the `nolabel` option in many other Stata commands.

`bdec(#)` specifies the number of decimal places reported for coefficient estimates. It also specifies the decimal places reported for standard errors if `se` is chosen. The default value for `bdec` is 3.

`tdec(#)` specifies the number of decimal places reported for t statistics. It also specifies the decimal places reported for R-squared or adjusted R-squared. The default value for `tdec` is 2.

`noparen` specifies that no parentheses be placed around t statistics or standard errors.

`noaster` specifies that no asterisks denoting 1% and 5% confidence intervals be reported.

`nocons` specifies that the intercept (constant) coefficient estimate not be reported.

`nonobs` specifies that the number of observations in the regression not be reported.

`noni` specifies that the number of groups in a panel data regression not be reported (e.g., the number of groups specified by the `i()` variable in `xtreg`).

`nor2` specifies that no R-squared (or adjusted R-squared) be reported. This option is only meaningful when Stata calculates a true R-squared.

`adjr2` specifies that the adjusted R-squared be reported rather than the unadjusted R-squared.

`nonotes` specifies that notes explaining the t statistics (or standard errors) and asterisks not be included.

`comma` specifies that the ASCII file output be separated by commas rather than by tabs.

`append` specifies that new estimation output be appended to an existing output file. The notes at the bottom of the table explaining the t statistics or standard errors and asterisks are appropriate for the first estimation in the output file. If subsequently appended estimation results use different options (such as `noaster`, or change the estimations' `robust` option), the notes will not be appropriate for all the columns.

`replace` specifies that it is okay to replace *filename* if it already exists.

Examples

We illustrate `outreg` using simulated data:

```
. set obs 100
. gen u = invnorm(uniform())
. gen x1 = _n
. gen x2 = 1/x1
. gen x3 = 0.01*x1^2
. gen x4 = 1/x3
. gen y = x1 - 4*x2 + 3*x3 - 2*x4 + u
. label var x1 "First x"
. label var x2 "Second x"
. label var x3 "Third x"
. label var x4 "Fourth x"
. label var y "Output"
. regress y x1-x4
  (output omitted)
. outreg using model1
```

This produces the default format for the results of `regress` in the file `model1.out`. Next,

```
. outreg x1-x3 using model1, se bdec(4) tdec(3) nocons nonotes replace
```

This restricts the reported output to coefficients for `x1`, `x2`, and `x3`, with no constant reported, displayed with 4 digits to the right of the decimal point. Standard errors rather than t statistics are reported in parentheses below the coefficients, with 3 digits to the right of the decimal point. Explanatory notes at the bottom of the table are omitted.

Finally, we do

```
. regress y x1-x2
  (output omitted)
. outreg using modl1_3, nolabel replace
. regress y x1-x3
  (output omitted)
. outreg using modl1_3, nolabel append
. regress y x1-x2 x4
  (output omitted)
. outreg using modl1_3, nolabel append
. type modl1_3.out
```

which gives

```
                          (1)         (2)         (3)
                            y           y           y
x1                        4.585       2.543       4.556
                       (72.08)**   (14.14)**   (58.28)**
x2                      480.112     410.828     448.646
                       (30.56)**   (34.91)**    (8.82)**
x3                                    1.888
                                   (11.66)**
x4                                                0.342
                                                 (0.65)
Constant                -97.793     -54.978     -95.236
                       (24.38)**   (12.23)**   (16.94)**
Number of observations  100         100         100
R-squared                 0.98        0.99        0.98
    Absolute value of t-statistics in parentheses
* significant at 5% level; ** significant at 1% level
```

sg98 | Poisson regression with a random effect

David Clayton, MRC Biostatistical Unit, Cambridge, david.clayton@mrc-bsu.cam.ac.uk

In its simplest use, this program duplicates the function of `nbreg` and fits the negative binomial regression model for overdispersed count data. The response for subject i is an event count, d_i, which has greater variance than that predicted by the Poisson model. A natural way to generate such a model is to assume that, conditional on random subject effects, u_i, a Poisson regression model holds for the rates λ_i (where $E(d_i) = \lambda_i y_i$, y_i being known rate denominators). Formally, with covariate vectors x_i,

$$
\begin{aligned}
[d_i|u_i] &\sim \text{Poisson}(\lambda_i y_i) \\
\log \lambda_i &= x_i^T \beta + u_i \\
\lambda_i &= f_i \exp x_i^T \beta
\end{aligned}
$$

where $f_i = \exp(u_i)$ are multiplicative subject effects, sometimes called "frailties."

Fitting such a model by maximum likelihood requires the further specification of the distribution of frailties in the population of subjects. Several models give tractable likelihoods (Hougaard 1984), but the most widely used model assumes f_i to be drawn from a Gamma distribution. Since the regression model will usually include a `_cons` term, the frailty distribution is constrained to have unit mean, but has unknown variance, κ. These assumptions define the negative binomial regression model (as fitted by `nbreg`). A convenient method of fitting is a Gauss–Seidel scheme in which we alternate between maximizing the likelihood with respect to β for given κ, and maximizing with respect to κ for given β. This behavior of the program is selected by

`. rpoisson` *depvar indepvars*`, e(`*denomvar*`) method(ml)`

Note that the estimation of β in the negative binomial case is one instance of a generalized linear model (see [R] **glm**), and we use the iteratively reweighted least squares algorithm of Nelder and Wedderburn (1972).

In practice, this model may be criticized as depending on a distributional assumption for frailties which is difficult to check in practice. The estimates of β remain consistent and efficient for any distribution of frailties; they are then maximum *quasi-likelihood* estimates (Wedderburn 1974). However, the estimates of their standard errors are only consistent if the estimate of κ consistently estimates the frailty variance. This will not be the case for maximum likelihood estimates if the frailty distribution is not a Gamma distribution. This can easily be seen from the fact that, if the f_i were directly observed, the maximum likelihood estimate of κ under the Gamma assumption would involve a contrast between arithmetic and geometric means, and the expected value of this is only equal to the variance in the case of the Gamma distribution. An alternative approach, which yields consistent

estimates of the frailty variance for any frailty distribution, is to maximize a *pseudo-likelihood* (Carroll and Ruppert 1982) with respect to κ. The pseudo-likelihood is obtained by assuming the residuals, $r_i = d_i - \mu_i$ (where $\mu_i = \mathrm{E}(d_i) = y_i \exp x_i^T\beta$), to be Gaussian with zero mean and variance $\mu_i + \kappa\mu_i^2$. This behavior of the program is obtained by

. rpoisson *depvar indepvars*, e(*denomvar*) method(ps)

This is the recommended (and default) method for fitting κ.

In certain cases, we might wish to constrain κ to take a fixed value. For example, if we wish to carry out a likelihood-ratio test for some regression coefficients, we drop some variables from the model, refit, and compare the log likelihoods. But we would not wish κ to increase in trying to "explain" the effects of the omitted covariates. The use of the program with fixed frailty variance is

. rpoisson *depvar indepvars*, e(*denomvar*) fvar(*value*) method(fi)

The default frailty variance is its fitted value in the previous fit.

For clustered data, $(d_{ij}, y_{ij}, x_{ij}; i = 1, \ldots, N, j = 1, \ldots, n_i)$, where i represents clusters and j represents records within clusters, the model may be extended such that all records within the same cluster share the same frailty. Formally, the extended model is defined by

$$
\begin{aligned}
[d_{ij}|u_i] &\sim \text{Poisson}(\lambda_{ij}y_{ij}) \\
\log\lambda_{ij} &= x_{ij}^T\beta + u_i \\
\lambda_{ij} &= f_i \exp x_{ij}^T\beta
\end{aligned}
$$

together with a model for the distribution of frailties over clusters. With a Gamma frailty distribution, maximum likelihood estimation of β involves solution of a set of estimating equations of the same general form as the generalized estimating equations of Liang and Zeger (1986). The same estimating equations deliver maximum quasi-likelihood estimates when the frailty distribution takes some other form (details are given by Clayton 1994). As before, the frailty variance may be estimated by maximum likelihood under the Gamma assumption or by pseudo-likelihood, treating the residuals $r_{ij} = d_{ij} - \mu_{ij}$ as Gaussian with

$$
\begin{aligned}
\mathrm{Var}(d_{ij}) &= \mu_{ij} + \kappa\mu_{ij}^2 \\
\mathrm{Cov}(d_{ij}, d_{ik}) &= \kappa\mu_{ij}\mu_{ik}
\end{aligned}
$$

(As before, μ_{ij} represent the "fitted values," $y_{ij} \exp x_{ij}^T\beta$). This use of the program is obtained by

. rpoisson *depvar indepvars*, e(*denomvar*) cluster(*clustid*) ...

where *clustid* is a variable containing cluster identifiers; all records sharing the same value for *clustid* are assumed to share the same frailty. (More than one variable may be specified here, in which case records must share the same value for all these variables in order to share the same frailty.)

An important use of rpoisson is in longitudinal studies of event counts within subjects. In such circumstances, the covariances between counts in different periods may not be as predicted by the simple random effects model. Specifically, covariances between counts in two periods which are close together in time may be greater than if the two periods are widely spaced. To allow for the possibility of misspecification of the variances and/or covariances, "robust" (Huber–White) estimates of the variance–covariance matrix of the estimate of β may optionally be computed.

The program can also be used to fit exponential (or piecewise-exponential) survival time models with random frailty (Clayton 1988, Clayton 1994) However, in this case the pseudo-likelihood method for estimating κ may not be applicable, and the maximum likelihood method should be used.

Syntax

rpoisson [*varlist*] [if *exp*] [in *range*] [, method({ ml | pe | ps | fi }) fvar(#)

cluster(*varlist*) robust exposure(*varname*) level(#) nolog irr]

Weights are not permitted. If the program is invoked with no arguments, the results of the previous fit are "replayed." This is usually done in conjunction with irr or robust options.

Options

method(ml|pe|ps|fi) chooses the estimation method for the frailty variance. Only the first two characters of the selection is significant: ml chooses maximum likelihood, pe chooses posterior expectation method, ps chooses maximum pseudo-likelihood, and fi specifies that this parameter should remain at a fixed value. ps is the default.

`fvar(#)` allows a value to be supplied for the frailty variance, either as a starting value when difficulties are encountered with convergence, or when the frailty variance is to be held at a fixed value for carrying out likelihood ratio tests (see above).

`cluster(`*varlist*`)` indicates that records are clustered, and that records within clusters should share the same frailty. One or more variables must be specified and these should uniquely identify clusters.

`robust` indicates that the Huber–White "information sandwich" should be used to calculate variances and covariances of the regression coefficients, in addition to the model-based estimates. This can be invoked when "replaying" the results of a previous fit. Once `robust` has been used, the robust standard errors will remain in force for all subsequent replays and for carrying out Wald tests using `testparm`.

The remaining options behave in the same way as for the `poisson` command. If no `exposure` option is specified, the rate denominators y_{ij} are treated as 1.0.

Example

The following example is an analysis of the data on counts of epileptic seizures in a clinical trial and are provided in full by Thall and Vail (1990). Subjects are in two treatment groups (31 receiving active treatment and 28 receiving placebo) and each subject is observed over four periods post-treatment. A baseline count of seizures in a pre-treatment run-in period is also available.

The variables used are as follows:

`id`	subject identifier
`trt`	treatment (1 = Active, 0 = Placebo)
`period`	period following treatment (1 – 4)
`d`	seizure count
`lbase`	log of pre–treatment seizure count

In the analysis below, frailties are shared by all four records for each subject, and fixed effects are fitted for treatment, period, and the log of the pre-treatment seizure count. This last variable is a subject-level covariate; without it the estimate of κ is much larger.

```
. xi:rpoisson d trt i.period lbase, cluster(id) method(ps) robust
i.period              Iperio_1-4   (naturally coded; Iperio_1 omitted)

Poisson regression with frailty
  frailty variance estimation: maximum pseudolikelihood

Cycle 1; Estimation of fixed effects
   Iteration 1; deviance = 1181.940
   Iteration 2; deviance =  991.137
   Iteration 3; deviance =  972.112
   Iteration 4; deviance =  972.031
   Iteration 5; deviance =  972.031
Cycle 1; Estimation of frailty variance
   Iteration 1; frailty variance =  0.257
   Iteration 2; frailty variance =  0.349
   Iteration 3; frailty variance =  0.358
   Iteration 4; frailty variance =  0.358
Cycle 2; Estimation of fixed effects
   Iteration 1; deviance =  444.723
   Iteration 2; deviance =  440.860
   Iteration 3; deviance =  440.646
   Iteration 4; deviance =  440.640
   Iteration 5; deviance =  440.640
Cycle 2; Estimation of frailty variance
   Iteration 1; frailty variance =  0.309
   Iteration 2; frailty variance =  0.309
Cycle 3; Estimation of fixed effects
   Iteration 1; deviance =  447.106
   Iteration 2; deviance =  447.106
Cycle 3; Estimation of frailty variance
   Iteration 1; frailty variance =  0.309

Number of observations = 236
Frailty varies by id (59 clusters)
Estimated frailty variance =   0.3094
Deviance =  447.106 ( 230 df)
```

```
------------------------------------------------------------------------------
       d |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     trt |  -.2812423   .1568355     -1.793   0.073      -.5886342    .0261496
Iperio_2 |  -.0685871   .0637958     -1.075   0.282      -.1936245    .0564503
Iperio_3 |  -.0625204   .0636959     -0.982   0.326      -.1873621    .0623213
Iperio_4 |  -.2029882   .0661261     -3.070   0.002       -.332593   -.0733834
   lbase |    1.02347   .1058225      9.672   0.000       .8160619    1.230879
   _cons |    .184813   .2245542      0.823   0.410      -.2553051     .624931
------------------------------------------------------------------------------

Estimates with robust standard errors:

------------------------------------------------------------------------------
       d |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     trt |  -.2812423   .1535621     -1.831   0.067      -.5822184    .0197339
Iperio_2 |  -.0685871   .1131875     -0.606   0.545      -.2904305    .1532563
Iperio_3 |  -.0625204   .1628448     -0.384   0.701      -.3816903    .2566496
Iperio_4 |  -.2029882   .1023568     -1.983   0.047      -.4036039   -.0023725
   lbase |    1.02347   .1144658      8.941   0.000       .7991214    1.247819
   _cons |    .184813   .2342376      0.789   0.430      -.2742843    .6439102
------------------------------------------------------------------------------
```

For the coefficient of interest (`trt`), the robust standard error differs little from the model-based standard error. With a sample as small as this, the model-based standard error is more reliable and the robust estimate is best regarded as a diagnostic for model misspecification.

A known problem

When using `method(ml)`, Newton's method is used for maximizing the likelihood with respect to κ, without any check that the likelihood is increased at each step. Occasionally, the algorithm steps too far and fails to converge. More careful coding could detect this and reduce the step length in such circumstances. However, the problem occurs infrequently, and a simple remedy is to use the `fvar` option to provide a better starting point for the iteration.

Acknowledgments

I must thank Michael Hills for helpful comments and suggestions and, together with many students in various short courses, for extensive testing of the code. This work was partially supported by NIH/NCI grant R01/CA61042.

References

Carroll, R. J. and D. Ruppert. 1982. Robust estimation in heteroscedastic linear models. *Annals of Statistics* 10: 429–441.

Clayton, D. 1988. The analysis of event history data: a review of progress and outstanding problems. *Statistics in Medicine* 7: 819–841.

——. 1994. Some approaches to the analysis of recurrent event data. *Statistical Methods in Medical Research* 3: 227–262.

Hougaard, P. 1984. Life table methods for heterogeneous populations: distributions describing the heterogeneity. *Biometrika* 71: 75–83.

Liang, K. and S. Zeger. 1986. Longitudinal data analysis using generalized linear models. *Biometrika* 73: 13–22.

Nelder, J. and R. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society,* Series A 135: 370–384.

Thall, P. and S. Vail. 1990. Some covariance models for longitudinal count data with overdispersion. *Biometrics* 46: 657–671.

Wedderburn, R. 1974. Quasi-likelihood functions, generalized linear models, and the Gauss–Newton method. *Biometrika* 61: 439–447.

sts13 | Time series regression for counts allowing for autocorrelation

Aurelio Tobias, Institut Municipal d'Investigacio Medica (IMIM), Spain, atobias@imim.es
Michael J. Campbell, University of Sheffield, m.j.campbell@sheffield.ac.uk

Introduction

Usually an epidemiological time series dataset consisting of counts is better modeled by a log-linear model using a Poisson distribution than linear regression. For models of the short-term effects of air pollution on health, the dependent variable (daily mortality) is a nonnegative count, and so a Poisson regression is used. This model assumes $\ln E(Y_t) = \beta X_t$ where X_t is the matrix of predictor variables on day t with regression coefficients β, Y_t is the number of deaths on day t, and E denotes the expected value operator. Time series data usually contain autocorrelation between observations. The presence of autocorrelation is often an indication of incomplete or inadequate model specification since the principal reason for autocorrelation of the deaths is that they are conditional on autocorrelated predictor variables. If the model were correct, the residual autocorrelation should

be minimal since we assume we are not dealing with an infectious disease and that one death does not cause another. Thus residual autocorrelation may imply confounding of air pollution associations due to unmeasured or mismodeled variables.

The APHEA Model

A main objective of the APHEA project (Katsouyanni et al. 1995, Schwartz et al. 1996) was to develop and standardize a methodology for the detection of short-term effects of air pollution on health using epidemiological time series. The solution proposed in the APHEA project was to include a specification of the autocorrelation in the model, and from this, the standard Poisson regression needs to be modified. Following Schwartz et al., Poisson regression with autocorrelated residuals was used. In this model the Y_t are assumed to be overdispersed Poisson $E(Y_t) = \exp(\sum \beta X_{t-\tau}) = \mu_t$, with $\text{Var}(Y_t) = \alpha\mu_t$. The covariance of Y_t is assumed to be of the form $\alpha A^{1/2} R A^{1/2}$, where A is a diagonal matrix with $A_{tt} = E(Y_t)$, α is a scalar to account for overdispersion, and R is a symmetric autocorrelation matrix generated by an autoregressive model. In practice, autocorrelation is modeled by turning autocorrelated residuals into the model as predictor variables. Suppose an autocorrelation of lag τ was observed in the residuals, in which $\widehat{\mu}_{t-\tau}$ was the fitted value for one iteration of the fitting process. Then $r_{t-\tau} = Y_{t-\tau} - \widehat{\mu}_{t-\tau}$ is included as a predictor variable in the next iteration. In contrast, Zeger (1988) uses an approach in which the autocorrelations are removed using an autoregressive filter. The Schwartz model is less explicit in the model description and the removal of serial correlation is empirical.

Marginal or conditional models

A number of authors have distinguished Marginal and Conditional models (Fitzmaurice 1998). For a marginal model $E(Y_t) = f(X_t, X_{t-1}, \ldots, X_{t-\tau})$ where the X_t's are external time-varying covariates. This is in contrast to a conditional model in which $E(Y_t) = f(X_t, X_{t-1}, \ldots, X_{t-\tau}, Y_{t-1}, \ldots, Y_{t-v}), \tau \geq 0, v \geq 1$, and the past values of the dependent variable are included as new predictor variables. It has been argued that marginal models are rather artificial, and give unlikely correlation structures. However, they are useful for modeling mean rates in populations. On the other hand, conditional models are useful for modeling changes in individuals but are poor at determining relationships between the Y and X variables because the parameters are not readily interpretable (Stanek et al. 1989).

Although both the Zeger and Schwartz models are ostensibly marginal models, the method of fitting the Schwartz model means that a residual $r_{t-\tau} = Y_{t-\tau} - \mu_{t-\tau}$ is included as a predictor variable, and so implicitly a lagged value of the Y variable is included as a predictor. Even the Zeger model filters the X variables by autoregressive parameters derived from the Y variables, and so neither model is purely marginal. However, the size of the residuals is such that they are unlikely to cause problems of bias.

Syntax

The code in `arpois` fits a log-linear model allowing for autocorrelation and overdispersion using iterative weighted least squares (IWLS). The command is based on Schwartz's `auto.sas` macro for the SAS system used in the APHEA project. `arpois` uses Stata's `nl` command to fit the log-linear model by IWLS, getting the initial values from a previous standard Poisson regression. Note that the nonlinear function is stored in another Stata program called `nlaphea`.

The syntax for the `arpois` command is as follows:

`arpois` *varlist* [*weight*] [`if` *exp*] [`in` *range*] [`, ar(#) delete prev`]

`aweights`, `fweights`, `iweights`, and `pweights` are allowed.

Options

`ar(#)` specifies the number of autoregressive terms to be included in the model. A non-autoregressive model is fit by default.

`delete` drops the working dependent variable, named `_z`, and the autoregressive terms, named `_rho1`, `_rho2`, and so on. By default these variables are stored in the dataset, because without them predicted values can not be computed, if they are requested using the `nlpred` command.

`prev` shows the previous, standard, Poisson model which neither allows for autocorrelation nor for overdispersion.

Example

As an example we reanalyzed the relationship between total mortality and nitrogen dioxide (NO2), in Barcelona, Spain, for the period 1986-1988 (Sunyer et al. 1996).

```
. use sunyer96
. describe
```

```
Contains data from sunyer96.dta
  obs:         1,096
 vars:            17                          25 Sep 1998 13:54
 size:        78,912 (92.3% of memory free)
-------------------------------------------------------------------------------
   1. date        float  %9.0g                 date yy/mm/dd
   2. death       float  %9.0g                 total mortality
   3. ltrend      float  %9.0g                 Linear trend
   4. ltrend2     float  %9.0g                 Quadratic trend
   5. year87      float  %9.0g                 dummy for year 1987
   6. year88      float  %9.0g                 dummy for year 1988
   7. sin1        float  %9.0g                 Sine term order 1
   8. cos1        float  %9.0g                 Cosine term order 1
   9. sin2        float  %9.0g                 Sine term order 2
  10. cos2        float  %9.0g                 Cosine term order 2
  11. sin3        float  %9.0g                 Sine term order 3
  12. cos3        float  %9.0g                 Cosine term order 3
  13. weekend     float  %9.0g                 dummy for weekend days
  14. temp        float  %9.0g                 temperature 24h average
  15. temp2       float  %9.0g                 temperature^2
  16. influ       float  %9.0g                 influenza epidemics
  17. no2         float  %9.0g                 no2 24h average
-------------------------------------------------------------------------------
Sorted by:  date
```

We first fit a standard Poisson model adjusted by trend, year, sinusoidal terms, weekdays, temperature, and influenza epidemics.

```
. poisson death ltrend ltrend2 year87 year88 sin1 cos1 sin2 cos2 sin3 cos3
> weekend temp temp2 influ no2

Iteration 0: Log Likelihood = -3868.6406
Iteration 1: Log Likelihood = -3850.8906

Poisson regression                                Number of obs    =    1096
Goodness-of-fit chi2(1080)  =  1531.188           Model chi2(15)   = 589.500
Prob > chi2                 =    0.0000           Prob > chi2      =  0.0000
Log Likelihood              = -3850.891           Pseudo R2        =  0.0711

------------------------------------------------------------------------------
       death |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
      ltrend |  -.0007054   .0001252     -5.634   0.000      -.0009508     -.00046
     ltrend2 |   3.76e-07   7.76e-08      4.849   0.000       2.24e-07    5.28e-07
      year87 |   .1315048   .0390977      3.363   0.001       .0548748    .2081348
      year88 |   .2626887   .0744463      3.529   0.000       .1167767    .4086008
        sin1 |   .0159753   .0169886      0.940   0.347      -.0173218    .0492724
        cos1 |   .1103722   .0130309      8.470   0.000        .084832    .1359124
        sin2 |  -.0013212   .0107732     -0.123   0.902      -.0224364    .0197939
        cos2 |   .0243337   .0070674      3.443   0.001       .0104818    .0381855
        sin3 |  -.0155174   .0076217     -2.036   0.042      -.0304556   -.0005792
        cos3 |   .0171352   .0065377      2.621   0.009       .0043215    .0299489
     weekend |  -.0180327   .0101981     -1.768   0.077      -.0380206    .0019552
        temp |  -.0114618   .0038899     -2.947   0.003      -.0190858   -.0038377
       temp2 |   .0005208   .0001559      3.341   0.001       .0002153    .0008263
       influ |   .0805044   .0173967      4.628   0.000       .0464076    .1146012
         no2 |   .0006437   .0002317      2.778   0.005       .0001896    .0010977
       _cons |   3.898009   .0346435    112.518   0.000       3.830109    3.965909
------------------------------------------------------------------------------
```

We obtained the Pearson residuals to check for autocorrelation, using the `acplot` command written by N. J. Cox available from `http://ideas.uqam.ca/ideas/data/Softwares/bocbocodeS320302.html` and included on the diskette with this STB insert. As we can see from Figure 1, there is a residual autocorrelation of order four.

```
. predict lpred
. gen pred=exp(lpred)
. gen resid=(death-pred)/sqrt(pred)
. acplot resid, spike se lags(6) xlabel(0,1,2,3,4,5,6)
```

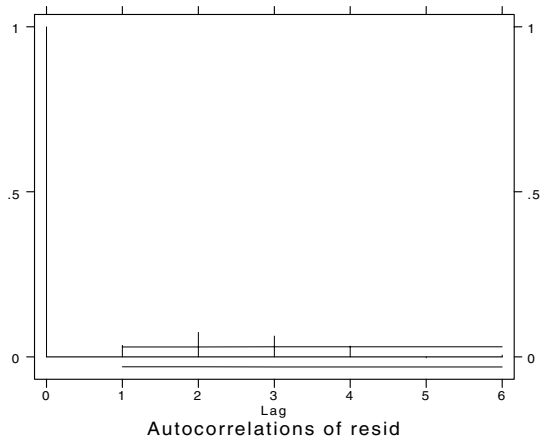

Figure 1. Autocorrelation plot of Pearson residuals from the previous standard Poisson model.

The next step is to fit a Poisson regression model allowing for fourth-order autocorrelation and overdispersion, using the `arpois` command.

```
. generate weekday = ~weekend
. arpois death ltrend ltrend2 year87 year88 sin1 cos1 sin2 cos2 sin3 cos3
> weekday temp temp2 influ no2, ar(4)

(sum of wgt is 49268.137)

Iteration 0:  residual SS =    33.862
Iteration 1:  residual SS =  33.51188
```

Source	SS	df	MS		
Model	13.4079132	19	.705679645	Number of obs	1092
Residual	33.5118806	1072	.031261083	F(19, 1072)	22.57
Total	46.9197939	1091	.043006227	Prob > F	0.0000
				R-squared	0.2858
				Adj R-squared	0.2731
				Root MSE	.176808
				Res. dev.	-699.136

(aphea)

_z	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
X0	3.880196	.042628	91.025	0.000	3.796553	3.96384
X1	-.0006823	.0001551	-4.399	0.000	-.0009866	-.000378
X2	3.69e-07	9.37e-08	3.941	0.000	1.85e-07	5.53e-07
X3	.1245167	.0475332	2.620	0.009	.0312481	.2177853
X4	.2519833	.0901474	2.795	0.005	.0750979	.4288687
X5	.0168415	.0203381	0.828	0.408	-.0230656	.0567486
X6	.1084906	.0155081	6.996	0.000	.0780609	.1389202
X7	.0004821	.0128359	0.038	0.970	-.0247042	.0256685
X8	.0233345	.0084154	2.773	0.006	.0068221	.039847
X9	-.0151304	.0091108	-1.661	0.097	-.0330075	.0027466
X10	.0166063	.0077786	2.135	0.033	.0013434	.0318693
X11	.0186632	.0121826	1.532	0.126	-.0052412	.0425677
X12	-.0119054	.0046277	-2.573	0.010	-.0209858	-.002825
X13	.0005252	.0001853	2.834	0.005	.0001615	.0008888
X14	.0782194	.0208612	3.750	0.000	.0372861	.1191527
X15	.0006635	.0002761	2.403	0.016	.0001218	.0012052
R1	.0273345	.0306481	0.892	0.373	-.0328025	.0874715
R2	.0707834	.0304327	2.326	0.020	.0110689	.1304979
R3	.0578235	.0304567	1.899	0.058	-.001938	.1175851
R4	.0232551	.03056	0.761	0.447	-.036709	.0832193

```
 Parameter X0 taken as constant term in model & ANOVA table
 (SE's, P values, CI's, and correlations are asymptotic approximations)
```

We again obtain the Pearson residuals to check for autocorrelation, but note that we need to use `nlpred` instead of `predict` to get the predicted values. From Figure 2, we can see that autocorrelation has been removed.

```
. nlpred lpred2
. gen pred2=exp(lpred2)
. gen resid2=(death-pred2)/sqrt(pred2)
. acplot resid2, spike se lags(6) xlabel(0,1,2,3,4,5,6)
```

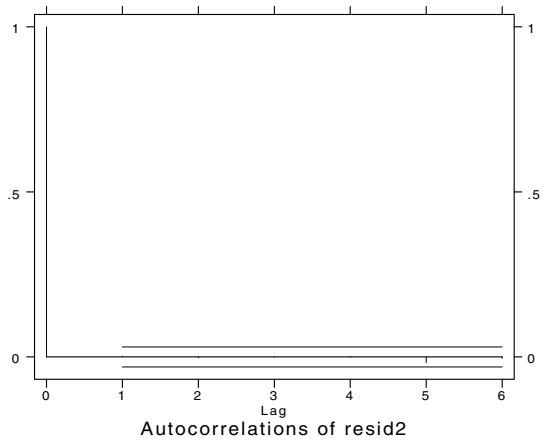

Figure 2. Autocorrelation plot of the Pearson residuals from the Poisson autoregressive model.

Comparing results

We compared results from our `arpois` command to those obtained with Schwartz's SAS macro, for a set of air pollutants; NO2, black smoke, and sulphur dioxide (SO2). The results are as follows:

	Non-autoregressive model				Order 4 autoregressive model			
	`arpois`		Schwartz's SAS		`arpois`		Schwartz's SAS	
Pollutant	β	SE	β	SE	β	SE	β	SE
NO_2	6.437	2.761	6.437	2.761	6.635	2.761	6.898	2.757
Black smoke	7.351	2.716	7.351	2.716	7.148	2.727	7.107	2.726
SO_2	−0.851	2.393	−0.851	2.393	−1.437	2.454	−1.371	2.442

When a non-autoregressive model is fit, the `arpois` command in Stata and the SAS macro give the same result, but when an autoregressive model (for example fourth order) is fit, we found slight differences in the estimates. We have two possible explanations for these minor differences. First, they could come from the degrees of freedom of the models. As we said previously, `arpois` is based on the `nl` command to fit a log-linear model by IWLS, while Schwartz's SAS macro uses the NLIN procedure. For NO2, the `nl` command uses 19 degrees of freedom instead of 20 used by `NLIN`. In the same way, the `nl` command has 1,072 residual degrees of freedom, while NLIN has 1076. This difference of 4 degrees of freedom is due to the inclusion of lagged autoregressive terms in the model. When we lag a variable, the first row becomes a missing value, when we do another lag, the first two rows become missing, and so on. In this case the `nl` command does not consider the first four observations because they are missing, but it seems that NLIN includes them for the analysis. Second, the auto SAS macro gets the residual values in each iteration of NLIN, while this doesn't seem to happen using the `nl` command in `arpois`. However, this problem needs to be studied further.

Acknowledgments

We are grateful to Nick Cox (University of Durham) and Vince Wiggins (Stata Corporation) for their help with the code, and Joel Schwartz (Harvard School of Public Health) for supplying the SAS macro. This work was done while Aurelio Tobias was visiting the Institute of Primary Care, University of Sheffield, and presented in the 4th Stata UK User Meeting. Aurelio Tobias was funded by the British Council and by the Research Stimulation Fund by SCHARR (University of Sheffield).

References

Fitzmaurice G. M. 1998. Regression models for discrete longitudinal data. In *Statistical Analysis of Medical Data: New Developments*, ed. B. S. Everitt and G. Dunn. London: Arnold.

Katsouyanni K., D. Zmirou, C. Spix, J. Sunyer, J. P. Schouten, and A. Ponka. 1995. Short-term effects of air pollution on health: a European approach using epidemiological time series data. *Eur. Respir. J.* 8: 1030–1038.

Schwartz J., C. Spix, G. Touloumi, L. Bacharova, T. Barumamdzadeh, and A. le Tertre. 1996. Methodological issues in air pollution studies and daily counts of deaths or hospital admissions. *J. Epidemiol. Community Health* 50: S3–S11.

Stanek E. J., S. S. Shetterley, L. H. Allen, G. H. Pelto, and A. Chavez. 1989. A cautionary note on the use of autoregressive models in the analysis of longitudinal data. *Statistics in Medicine* 8: 1523–1528.

Sunyer J., J. Castellsagu, M. Saez, A. Tobias, and J. M. Anto. 1996. Air pollution and mortality in Barcelona. *J. Epidemiol. Community Health* 50: S76–S80.

Zeger, S. L. 1988. A regression model for time series of counts. *Biometrika* 75: 621–629.

STB categories and insert codes

Inserts in the STB are presently categorized as follows:

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

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

Guidelines for authors

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

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

A submission to the STB consists of

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

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

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

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

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

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

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

International Stata Distributors

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

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

Company:	Dittrich & Partner Consulting
Address:	Prinzenstrasse 2 D-42697 Solingen Germany
Phone:	+49 2 12 / 33 90 - 200
Fax:	+49 2 12 / 33 90 - 295
Email:	sales@dpc.de
URL:	http://www.dpc.de
Countries served:	Germany, Austria, Italy

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

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

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

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

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

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

(*List continued on next page*)

International Stata Distributors

(*Continued from previous page*)

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

Company:	Timberlake Consultants
Address:	47 Hartfield Crescent West Wickham Kent BR4 9DW United Kingdom
Phone:	+44 (0)181 462 0495
Fax:	+44 (0)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:	Calle Monte Carmelo 36, Bajo 41011 Sevilla Spain
Phone:	+34 (9) 5 428 4094
Fax:	+34 (9) 5 428 4094
Email:	timberlake@zoom.es
Countries served:	Spain

Company:	Timberlake Consultores, Lda.
Address:	Praceta Raúl Brandao, n° 1, 1° E 2720 Alfragide Portugal
Phone:	+351 (0)1 471 73 47
Fax:	+351 (0)1 471 73 47
Telemóvel:	+351 (0)931 6272 55
Email:	timberlake.co@mail.telepac.pt
Countries served:	Portugal

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

For the most up-to-date list of Stata distributors, visit http://www.stata.com