STATA
TECHNICAL
BULLETIN

May 1992
STB-7

A publication to promote communication among Stata users

Contents of this issue

an1.1 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	data sets	*qs*	questions and suggestions
gr	graphics	*tt*	teaching
in	instruction	*zz*	not elsewhere classified
Statistical Categories:			
sbe	biostatistics & epidemiology	*srd*	robust methods & statistical diagnostics
sed	exploratory data analysis	*ssa*	survival analysis
sg	general statistics	*ssi*	simulation & random numbers
smv	multivariate analysis	*sss*	social science & psychometrics
snp	nonparametric methods	*sts*	time-series, econometrics
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified

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

an17 Stata seminars announced

Joseph Hilbe, Editor, STB, 602-860-4331, FAX 602-860-1446

Seminars featuring intermediate and advanced use of Stata are scheduled for August 14–15 at the University of New Hampshire and August 19–20 at California State University, Fullerton.

Dr. Lawrence Hamilton, Professor of Sociology at the University of New Hampshire and author of *Statistics with Stata*, *Regression with Graphics*, and other related works, and Dr. Joseph Hilbe, Editor of the STB, are the instructors at the New Hampshire site and Dr. J. Theodore Anagnoson, Professor and Chair of the Department of Political Science at California State University, Los Angeles and author of numerous articles on EDA and NSF EDA Workshop leader, and Hilbe will conduct the California seminar.

The focus of discussions will be Exploratory Data Analysis, regression modeling and diagnostics, including robust and quantile regression, logistic regression including ordinary, grouped, conditional, ordinal, multinomial, and Huber random effects modeling and diagnostics, and Stata 3.0 programming techniques. Dr. Hamilton will also discuss Monte Carlo sampling. Each seminar includes a format of theoretical and applied discussion as well as allowing for on-hands "learn-by-doing." All participants will have a 386 PC on which to work. Numerous handouts and data sets will be provided. Participants in the New Hampshire seminar will receive free copies of Dr. Hamilton's books *Statistics with Stata, Version 3* and *Regression with Graphics.* Participants in the Los Angeles seminar will receive a free copy of the Stata Graphics Editor. The cost is $395 per participant. Nearby accommodations are available at conference rates.

Preliminary Schedule

August 14–15, University of New Hampshire

Aug. 14, Fri.	AM	Hamilton	Introduction
	AM	Hamilton	Exploratory data analysis
	PM	Hilbe	Logistic regression modeling and diagnostics
Aug. 15, Sat.	AM	Hamilton	Regression modeling and diagnostics
	PM	Hilbe	Stata 3.0 programming

August 19–20, California State University, Fullerton

Aug. 19, Wed.	AM	Anagnoson	Introduction / Data management
	AM	Anagnoson	Exploratory data analysis
	PM	Hilbe	Logistic regression modeling and diagnostics
Aug. 20, Thu.	AM	Anagnoson	Regression modeling and diagnostics
	PM	Hilbe	Stata 3.0 programming

Registration forms and more information can be obtained from CRC, telephone 800-782-8272 or fax 310-393-7551 or directly from Joseph Hilbe, 602-860-4331, fax 602-860-1446.

Space is limited at both locations due to individual computer use; early registration is advised. Note that the New Hampshire session starts the day following the American Statistical Association convention in Boston—less than two hours by car. Also, California State University, Fullerton is very near Disneyland. We mention this for those who desire additional and alternative stimulation after the seminar.

an18 STB-1—STB-6 available in bound format

Joseph Hilbe, Editor, STB, FAX 602-860-1446

The first year of the *Stata Technical Bulletin* has been reprinted into a 200+ page, bound book called *The Stata Technical Bulletin Reprints, Volume 1*. The volume is available from CRC for $25—$20 for STB subscribers—plus shipping. Authors of inserts in STB-1—STB-6 will automatically receive the book at no charge and need not order.

Everything that appeared in the first six issues of the original journals appears in this volume, implying (1) there is no reason to purchase this volume if you have saved your original STBs and (2) the bound format is a perfect substitute for the original STBs and more easily stored since it fits on a bookshelf. Our primary reason for reprinting the STB is to make it easier and cheaper for new users to obtain back issues. For those not purchasing the volume, note that *zz1* in this issue provides a cumulative index for the original STBs.

an19 Stand-alone statistical tools

Gerard Dallal, 53 Beltran Street, Malden MA 02148

STATOOLS$^{\text{tm}}$ are stand-alone FORTRAN programs to fill in some gaps left by major statistical packages. They are available on 5.25-inch disks for IBM PC and compatibles running DOS 2.0 or later versions. The disks contain executable files, user guides, and, in some cases, FORTRAN source code.

Program	Source included	Price	Description
PC-SIZE	N	$15	sample size calculations
PC-PLAN	Y	10	generates randomization plans
PC-EMS	N	10	tables of expected mean squares for balanced experiments
PC-AIP	Y	10	fits additive-in-probits models to 2-D contingency tables
STAT-SAK	Y	10	Statistician's Swiss army knife
OCTA	N	10	interactive log-linear analysis
RCJOIN	Y	5	identifies sources of interactions in 2-way tables of counts
MUDIFT	N	5	multivariate distribution-free comparison of growth curves
PITMAT	Y	10	significance levels using recursive relationships
TRACK	Y	10	Foulkes-David tracking index

Other programs are available as well. Please contact Gerard E. Dallal, 53 Beltran Street, Malden, MA 02148 for more information.

crc12.1 Oops!, again

In *crc12*, we reported that [5s] ci incorrectly states that the standard error of the mean s_μ is defined $\sqrt{s^2/(n-1)}$ rather than $\sqrt{s^2/n}$, and that the `ci` and `cii` commands themselves shared this misconception. We also claimed to have fixed the problem.

We did fix the problem for `cii`, but not for `ci`. Installing the `crc` directory from the STB-7 diskette will finally put an end to this problem.

In addition, the `lvr2plot` command described in [5s] fit was implemented incorrectly and is fixed with the installation of these updates. The program incorrectly graphed leverage against the "normalized" predictions squared rather than the normalized residuals squared. The graphs shown on page 301 of volume 2 are also incorrect as they were drawn with the uncorrected `lvr2plot` command.

dm7 Utility to reverse variable coding

Marc Jacobs, Dept. of Sociology, Univ. of Utrecht, The Netherlands, FAX (011)-31-30 53 4405

The syntax for `omscore` is

`omscore` *varname*

`omscore` creates the new variable `rr_`*varname*; e.g., '`omscore x`' creates `rr_x`.

There are times when I have found a variable to be coded in the reverse of how I desired. I submit a utility program that reverses variable coding. It is necessary, however, that the original coding be in numeric order; for example, 1, 2, 3, 4, 5. The algorithm used is

$$\text{score}_{\text{new}} = (\text{score}_{\text{max}} + \text{score}_{\text{min}}) - \text{score}$$

Two conditions must obtain in order to use this utility. First, it is necessary that the original coding be incremented by one and be in joined numeric order, as in 1, 2, 3, 4, 5. Second, the variable name can be no longer than five characters.

dm8 Command to unblock data sets

Joseph Hilbe, Editor, STB, FAX 602-860-1446

The `blogit` and `bprobit` commands ([5s] glogit) attempt to deal with "blocked" data, that is, data in which the number of positive and negative outcomes are contained in the same observations. Think of estimating a logit model of `posresp` in terms of `x1`, `x2`, and `x3`, with frequency-weighted data. One way the data might be recorded is

```
. list in 1/2
      posresp        pop         x1         x2         x3
  1.        3          5          1          5          9
  2.        2          3          2          8          3
```

Observation 1 says that in the first group (the group with `x1` = 1, `x2` = 5, and `x3` = 9), there were 3 positive responses (`posresp` = 3) out of 5 (`pop` = 5). There were 2 positive responses out of 3 in the second group. In this form, we could estimate our model using `blogit` by typing '`blogit posresp pop x1 x2 x3`'.

The problem is that `blogit`, being based on `logit`, does not provide the diagnostic features of `logistic`. We could use `logistic` with this data, but first we would have to unblock it. That is, the first observation would become two observations—one reflecting positive responses and a second reflecting the negative responses—and similarly for the second. The unblocked version of our data would appear as

```
. list in 1/4
      posresp        pop         x1         x2         x3
  1.        1          3          1          5          9
  2.        0          2          1          5          9
  3.        1          2          2          8          3
  4.        0          1          2          8          3
```

In this format, the data can be used with either `logit` or `logistic`; using `logistic`, the command would be '`logistic posresp x1 x2 x3 [freq=pop]`'.

Thus, an alternative to `blogit` could be the more powerful `logistic` command, but only after we unblock our data. The `unblock` command does this:

`unblock` *pos_var pop_var* [`, gen(`*new_grp_var*`)`]

pos_var is the variable recording the number of positive responses and *pop_var* the total population. At the conclusion of this command, *pos_var* will contain a 0/1 variable, with 1 indicating a positive response; *pop_var* will contain the total population for the positive or negative response; and the optionally generated *new_grp_var* will contain a group identification number 1, 2, …, n, where n is the total number of observations in our original data set. The new data set will have between n and $2n$ observations. Each observation in the original data set becomes two observations if there are both positive and negative responses and one observation if there are only positive or negative responses.

new_grp_var, if requested, ties the new observations back to their original structure. If two observations in the new data set have *new_grp_var* equal to k, then both were created from the kth observation in the original data. Unless one is interested in going back to the blocked structure from the unblocked structure—and there is no reason why one should—this variable will be of no use.

Thus, starting with the data shown at the start of this insert, one could estimate using `logistic` by typing

```
. unblock posresp pop
. logistic posresp x1 x2 x3 [freq=pop]
```

The advantage, of course, is that you can now use all the post-`logistic` commands described in [5s] logistic.

dm9 An ANOVA blocking utility

Peter A. Lachenbruch, Dept. of Biostatistics, UCLA

I have created a short ado-file which I find very useful in ANOVA problems. It corresponds to the `%gl(a,b)` command in GLIM. It generates a variable with levels 1 through A in blocks of B. Thus, `gl 2 3 g2` will generate a variable named `g2` with the series 1 1 1 2 2 2 1 1 1 2 2 2, etc., through the end of the data. This can be used to generate the needed levels for several factors without the necessity for entering all the numbers. The data, of course, must be sorted in an order corresponding to the factor levels.

The program for doing this, included on the STB-7 diskette, is

```
program define gl
        version 3.0
        gen `3´=int(mod((_n-1)/`2´,`1´))+1
end
```

gr10.1 Printing graphs and creating WordPerfect graph files

Marc Jacobs, Dept. of Sociology, Univ. of Utrecht, The Netherlands, FAX (011)-31-30- 53 4405

The disadvantage of using the HP-driver (`hp7475ls.pen`) for creating WP graphs is that shading is not printed very smoothly. Besides that, importing a thus created Stata graph into WP is slow. Therefore using the Lotus PIC file driver results in a better picture. The shading is neatly translated into parallel running lines that prints nicely on a HP-printer and copies just as well. The Saving and Montgomery (1992) program `gphwp` can be modified by replacing the next to the last line with:

```
gphpen %1 /d \pic.pen /oc: \%1.wpg /n /t1111 /p111111111
```

The created file can be imported (scaled and rotated) into WordPerfect. It saves considerable data space when the graph is not physically imported into the text file, but is treated as a "file on disk." In WP:

```
Alt-F9,1,1,2,2 (file on disk), 1 (choose filename)
```

References

Saving, T. and J. Montgomery. 1992. gr10: Printing graphs and creating WordPerfect graph files. *Stata Technical Bulletin* 5: 6–7.

os4 Stata icon for Microsoft Windows 3.1

Joseph Hilbe, Editor, STB, FAX 602-860-1446

I have created a Windows icon for users who desire to run Stata under the Windows 3.1 operating system. It was made using the Borland C++ Resource Workshop utility and can be linked to either `stata.exe`, `istata.exe`, or both.

`stata.ico` and `istata.ico`, found on the STB-7 diskette, should be copied to the directory where you have placed `stata.exe`; e.g. `c:\stata`. If you have Intercooled Stata and have named it `stata.exe`, disregard the `istata.ico` file. You must have already created a Stata Program Group and an appropriate `pif` file for `stata.exe` and/or `istata.exe` per manual instructions ([4] win3). To link the icon with `stata.exe` do the following:

1. Access Stata 3.0 Program Group.
2. Click once on or select default `ms dos` icon for Stata. If you have not yet done this, select New from Program Manager and create a Program Item—then go to 5.
3. Select File in Program Manager.
4. Select Properties from menu.
5. Select Change Icon.
6. Select Browse. There may be a screen message prior to selection informing the user that no icon exists. Select OK to accept default and change the directory to `c:\stata\*.ico`.
7. Select `stata.ico`. Press or click OK.
8. Press or click OK from Change Icon Group.
9. Press or click OK from Program Item Properties.
10. `stata.ico` should be linked to `stata.exe` and be visible in the Stata Program Group.

You may follow the same procedure for `istata`.

sbe5 Calculating person-years and incidence rates

Joseph Hilbe, Editor STB, FAX 602-860-1446

"Person-years" provides a means to allocate the amount of time a case contributes to a particular age group or interval. Total person-years represents the sum of the time that individual cases in the study contribute to a group. Epidemiologists frequently use person-years as the denominators for various statistical calculations. For instance, in Poisson regression models, person-years are the time-based denominators used to obtain disease or incidence rates for longitudinal studies. The Stata incidence rate command (`ir` or `iri`; see [5s] epitab) also uses person-years as a time-variable denominator.

`pyears` allows calculation of summary person-year totals for each specified age group as well as numerator failure totals. The syntax for `pyears` is

`pyears` $dob_{\rm var}$ $begin_{\rm var}$ $end_{\rm var}$ $\big[dead_{\rm var}\big]$, `i(`$\#_{\rm interval}$`)` `g(`$\#_{\rm group}$`)` `s(`$\#_{\rm start}$`)` [`list death`]

where $dob_{\rm var}$ is "date of birth," $begin_{\rm var}$ is the date the patient entered the test or trial, and $end_{\rm var}$ is the date on which the patient either withdrew from the study or died, or is the date on which the study ended. The optional $dead_{\rm var}$ indicates whether the patient died. $dead_{\rm var}$ is not used in person-year calculations, but summary statistics appropriate to ascertaining incidence rates are provided with the `death` option, where this information is required. Unadjusted rates are calculated by dividing the number of failures by person-years in each group.

`pyears` requires three additional parameters: `interval()`, the length in whole years of the age group interval; `group()`, the number of age groups; and `start()`, the starting year of the first age group. For example, if a study is performed on patients whose ages range from 30 to 50 during the test period, and if the age groups are stratified into four groups of five years each, then the command line options would read `interval(5) group(4) start(30)`.

Options include `list`—screen output of a casewise listing of person-year contribution per age group—and `death`—summary statistics for the number of cases entering and ending the study and the number of failures or deaths (coded 1 for failure) per age group.

All dates must first be converted to elapsed dates; that is, the number of days from January 1, 1960. Stata's date commands (see [5d] dates) allow the user to easily convert to elapsed dates from a variety of date formats. For instance, if a date is stored as *yymmdd*, convert to an elapsed date using the `ftoe` command. In the example below, I converted `dob` to `sdob`, an elapsed date, by typing '`ftoe dob, gen(sdob)`'.

There is no `by()` option currently; however, it is rather simple to separate groups and perform a `pyears` on each to achieve the same result. In fact, doing this yourself allows calculation of numerous person-year strata.

The example below provides the results of calculating both individual and total person-years on a five-observation data set. Note that all observations having a missing value for any of the three time variables will be deleted prior to calculating person-years.

```
. use pyrs

. describe

Contains data from pyrs.dta
  Obs:      5 (max=126272)
 Vars:      6 (max=    52)
Width:     24 (max=   102)
  1. dob          float  %9.0g          Date of Birth
  2. begin        float  %9.0g          Enter Testing
  3. end          float  %9.0g          End Testing
  4. sdob         long   %10.0g         SAS DOB
  5. bs           long   %10.0g         SAS Enter Testing
  6. es           long   %10.0g         SAS End Testing
  7. dead         float  %9.0g          Dead (1|0)
Sorted by:

. list
          dob     begin       end     sdob       bs        es   dead
  1.   441201    840801    911231    -5509     8979     11687      0
  2.   510801    840301    911231    -3075     8826     11687      0
  3.   540601    840901    890801    -2040     9010     10805      1
  4.   450101    900301    911131    -5478    11017     11657      0
  5.   500706    850906    881006    -3466     9380     10506      1

. * Note: 5 year intervals, 4 groups, starting at 30 years of age
. pyears sdob bs es, i(5) g(4) s(30) l d

        ingrp1     ingrp2     ingrp3     ingrp4
   1.        .   .3312798   4.999999   2.080081
   2. 2.414099   4.999999   .4161532          .
   3. 4.744009   .1676933          .          .
   4.        .          .          .   1.752224
   5.        .   3.082819          .          .

Person-years:           Totals      Enter     End    Failure
-------------------------------------------------------------

 Grp  30 to 34   =>       7.1581       2        0         0
 Grp  35 to 39   =>       8.5818       2        2         2
 Grp  40 to 44   =>       5.4162       0        1         0
 Grp  45 to 49   =>       3.8323       1        2         0
```

I welcome any suggestions from users as well as examples of program use.

References

Kahn, H. A. and C. T. Sempos. 1989. *Statistical Methods in Epidemiology*. New York: Oxford University Press.

sbe6 3x3 matched case–control tests

Joseph Hilbe, Editor, STB, FAX 602-860-4331

`mcc3i` is an immediate command to calculate appropriate χ^2 statistics and significance tests for 3×3 matched case–control tables. Its syntax is

`mcc3i` $\#_{11}\ \#_{12}\ \#_{13}\quad \#_{21}\ \#_{22}\ \#_{23}\quad \#_{31}\ \#_{32}\ \#_{33}$

Three χ^2 statistics are provided on output: Stuart–Maxwell, Extended McNemar, and Fleiss–Everitt. A summary table of the differences between cases and controls is also displayed.

The Stuart–Maxwell test is a modification of Stuart (1955) and Maxwell's (1970) test derived by Fleiss and Everitt (1971) for 3×3 tables. Let such a table be characterized as follows:

	A	B	C	Total
A	n_{11}	n_{12}	n_{13}	n_{1*}
B	n_{21}	n_{22}	n_{23}	n_{2*}
C	n_{31}	n_{32}	n_{33}	n_{3*}
Total	n_{*1}	n_{*2}	n_{*3}	n_{**}

Define $d_i = ni* - n_{*i}$, $i = 1,2,3$, and $\overline{n}_{ij} = (n_{ij} + n_{ji})/2$.

Stuart–Maxwell test

The Stuart–Maxwell χ^2 statistic is

$$\chi^2 = \frac{\overline{n}_{23}d_1^2 + \overline{n}_{13}d_2^2 + \overline{n}_{12}d_3^2}{2(\overline{n}_{12}\overline{n}_{13} + \overline{n}_{12}\overline{n}_{23} + \overline{n}_{13}\overline{n}_{23})}$$

The Fleiss–Everitt test

The Fleiss–Everitt ordered categories χ^2 statistic is

$$\chi^2 = \frac{(d_1 - d_3)^2}{2(\overline{n}_{12} + 4\overline{n}_{13} + \overline{n}_{23})}$$

This test should be used if the three outcome categories are ordered. However, the significance value is treated differently if the comparison was planned prior to the collection of the data. In this case, the χ^2 distribution is given with one degree of freedom. For retrospective studies, two degrees of freedom are appropriate. Both values are provided to the user.

Extended McNemar test

McNemar χ^2 tests whether the corners of the 3×3 table are symmetrical; row and column totals as well as the diagonal are ignored.

$$\chi^2 = \sum_{i=1}^{r}\sum_{j>i}\frac{(n_{ij} - n_{ji})^2}{n_{ij} + n_{ji}}$$

Example

The following table of hypothetical data is found in Fleiss (1981, 121):

	Diagnostician A			
Diagnostician B	Schizophrenia	Affective	Other	Total
Schizophrenia	35	5	0	40
Affective	15	20	5	40
Other	10	5	5	20

The command and output follows:

```
. mcc3i 35 5 0 15 20 5 10 5 5
                               Controls
          Cases  |     A         B         C    |    Total
          -------+------------------------------+---------
             A   |    35         5         0    |      40
             B   |    15        20         5    |      40
             C   |    10         5         5    |      20
          -------+------------------------------+---------
          Total  |    60        30        10    |     100

                        3X3 Matched Case-Control Tests

Stuart-Maxwell Chi2 =     14.00       Pr>chi2(2)      =  0.0009

Extend McNemar Chi2 =     15.00       Pr>chi2(3)      =  0.0018

Fleiss-Everitt Chi2 =     12.86       Pr>chi2(1) Pre  =  0.0003
  (ordered cells)                     Pr>chi2(2) Post =  0.0016

Summary Differences Between Cases and Controls

  Diff 1 =     -20
  Diff 2 =      10
  Diff 3 =      10
```

References

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

Fleiss, J. L. and B. S. Everitt. 1971. Comparing the marginal totals of square contingency tables. *Brit. J. Math. Stat. Psychol.* 24: 117–123.

Maxwell, A. E. 1970. Comparing the classification of subjects by two independent judges. *Brit. J. Psychiatry* 116: 651–655.

Stuart, A. 1955. A test for homogeneity of the marginal distribution in a two-way classification. *Biometrika* 42: 412–416.

Zar, J. 1984. *Biostatistical Analysis.* Englewood Cliffs, NJ: Prentice–Hall.

sed7 Resistant smoothing using Stata

Isaias H. Salgado-Ugarte, Biologia, E.N.E.P. Zaragoza, U.N.A.M., Mexico City, Mexico, FAX (011)-52-5-744-1217
Jaime Curts Garcia, Evaluacion y Proyectos Academicos, U.N.A.M., Mexico City, Mexico

As has been shown in several articles in the STB (Geiger 1991; DeLeon and Anagnoson 1991), one of the main purposes of exploratory data analysis (EDA) techniques is the finding of trends and patterns that are nonlinear. Science and other human activities produce data for which sequential order is important and for which values are defined by the adjacent ones in the series. Though “time series” are the general examples of such bivariate data—e.g., daily temperature values or rainfall recorded at a meteorological station; daily body temperature of milk producer cows; or amount of fish caught at a sea region over several years—it is possible to consider other types of variables to specify order, such as the resistivity of geological materials along stratigraphic sections (distance) or the relative frequency of the size of aquatic organisms (length). Often those patterns are hidden by erratic fluctuations (noise) in the sequence. The smoothers eliminate the noise and make clearer the gradual variable behavior.

Any smoother decomposes the original sequence into two parts: a structured smooth with gradual variation sequence and a noisy, rough, and varied sequence according to the schematic expression:

$$\text{data} = \text{smooth} + \text{rough}$$

The smoothed sequences show patterns that can easily be understood, as seasonal variations and long-term trends. On the other side, the rough values (residuals of smoothers) make possible the discovery of additional patterns or extraordinary values (spikes) that deserve additional attention (Velleman 1982).

Traditionally, moving averages have been employed to smooth sequences. However, this kind of smoother presents some undesirable results because of shifting of peak and trough positions (Davis 1973) and their lack of resistance to spikes. Spikes are isolated extraordinary observations that affect not only the smoothed value at that point, but all other smoothed values in which the average participates (Velleman and Hoaglin 1981). For that reason, a few spikes severely occult the subjacent pattern of a sequence. As a response to this non-resistance, Tukey (1977) suggested the use of moving or running medians that are resistant to spikes. Velleman (1980, 1982) analyzed several properties and performances of this type of smoother and provides us with some guidance for its understanding and application. These kinds of exploratory procedures have been named resistant nonlinear smoothing (RNLS) after their performance and mathematical basis (Velleman 1982).

Resistant smoothers have successfully been used to explore data for patterns and trends which might not be so readily exposed by more commonly used classical techniques (see Himes and Hoaglin 1989, who compared cubic splines with the

4253EH,twice smoother and found that the latter captured the structure of the raw data better than the former). Following Tukey (1977), by exploration one can reduce the impact of the "rough" on methods intending to seek generalization. What is desirable is a "rough" that has no "smooth"; for example, graphical representation of roughs (residual of smooth) should contain no additional pattern or structure. If the graph of roughs does show some additional structure not removed by the smoother, then further or alternate smoothing should take place (Curts 1986; Himes and Hoaglin 1989).

To show the effects of the resistant, nonlinear smoothers, we utilize data coming from fisheries analysis (Salgado-Ugarte 1991). Table 1 (Alejo-Plata et al. 1989) contains the number of fishes (tilapia) by size (standard body length). It is expected that the frequency distribution be composed by several gaussian components. Plotting these values one hardly can distinguish such a thing (Figure 1), so it is desirable to smooth the values. Instead of using moving averages of three or five (as suggested by Laurec and Mesnil 1987), we prefer to employ a resistant smoother.

Table 1: Length-frequency data and 4253EH smoothing values

Standard body length	Frequency (individuals)	Smoothed values
37	6	6.0000
38	10	6.0000
39	3	6.0000
40	7	6.0000
41	5	6.0000
42	9	5.9375
43	3	5.8125
44	5	5.8125
45	11	5.9375
46	4	6.1875
47	6	6.6250
48	10	6.9375
49	6	7.3750
50	6	8.6875
51	12	10.3125
52	13	11.1250
53	13	11.2500
54	6	10.6250
55	12	9.0625
56	8	7.6250
57	7	6.9375
58	5	6.7500
59	5	6.7500
60	12	7.5625
61	5	9.2500
62	12	10.1875
63	11	10.2500
64	10	10.0000
65	10	8.9375
66	3	7.5625
67	7	7.0000

The simplest resistant smoothers are those that use running medians of groups spanning three points and are resistant to isolated spikes in the sequence. Running medians of span three are, however, affected by two extraordinary values. Increasing the span, for example running medians of span 5, attenuates the problem. Even though these uneven group size smoothers are easy to compute by hand, they are less efficient than those using medians of even size group data values, but such even size span smoothers move the position of smoothed values at the center of each group (between the two central points). To recover the original position, it is required to apply a second running median of span four (to provide resistance) followed by a running median of span two to recover phase.

It is possible to combine even and uneven span running median smoothers, a procedure known as re-smoothing. Traditional weighted moving average smoothers, as the one with a span of three and weights of 1/4, 1/2 and 1/4 ("Hanning," after Julius von Hann, an Austrian meteorologist of the 19th Century) can be also used. The general principle is first to apply resistant smoothers of larger span and then to smooth with hanning to provide smoother sequences. To briefly represent these compound smoothers, the span of each is written. In this way, the digits 42 indicate a running median of span four re-smoothed by a running median of span two. The repeated running median of span 3 until no changes occur (simplest compound smoother) is written as '3R'. The hanning is indicated by an 'H' (Tukey 1977; Velleman and Hoaglin 1981).

Additionally, the terminal points of the sequence are estimated by the median of three values: the observed, the nearest smoothed, and the one from linear extrapolation of the last two smoothed values one point after the first (or last) data point. This rule is known as the "endpoint adjustment" and here we indicate its application by an 'E' in brief notation, as suggested by Velleman and Hoaglin, 1981 (some programs compute resistant smoothing but do not provide the endpoint adjustment).

These compound smoothers permit elimination of noise but, at the same time, they suppress other interesting trends that the sequence may contain. To recover these additional trends, the rough values are smoothed and the result is added to the first smoothed values (a procedure called ‘re-roughing”). It is preferable to apply the same re-smoothing combination to the roughs so the re-roughing procedure can be indicated by the brief notation “twice”. Velleman (1980) recommends in the first place the compound smoother 4253EH,twice due to its good performance under several unfavorable conditions (the others are 43R5R2H,twice, 3RSSH, and 53EH,twice).

It is clear that the RNLS is one of the most useful techniques of the EDA procedures. However, the numerous calculations required can discourage even the most enthusiastic analyst. For this reason the use of computerized methods are particularly suited. The programming capabilities of Stata make it possible to construct several resistant smoothers. Hamilton (1990a) discusses some elementary smoothers (moving averages and running medians of three in addition to the hanning moving weighted average). The Stata programs (ado-files) written by Dr. Hamilton for these procedures are contained in the student version of Stata and commented on in his book *Statistics with Stata* (Hamilton 1990b). This contribution contains one ado-file to perform a 4253EH smoother. With additional result editing and repeated smoothing (as indicated below), it can compute the “twice” part of the smoother and will produce 4253EH,twice as recommended by Velleman (1980, 1982) and Velleman and Hoaglin (1981). This program uses programs developed by the authors in combination with Dr. Hamilton’s algorithms.

The `sm4253eh.ado` file computes running medians of span four, relocated by running medians of two, followed by uneven span resistant smoothers (span five and three), endpoints adjustment and finally the weighted moving average “hanning.” The syntax of the program is

`sm4253eh` *datavar smthvar*

where *datavar* is the variable containing the data and *smthvar* is the variable that will hold the smoothed values resulting from the smoothing process.

To use this smoother, the values of the original sequence are entered in a file (it is recommended to include only the response variable values, in this case the frequency in the number of fishes). Once the sequence is in memory, one runs the program to apply the smoother. The result is only the smoothed sequence (the original data are dropped) and an index variable (`timendx`), which can be used to plot results. Applying the above steps to the data used for the example, we obtain the smooth values, which were plotted against `timendx` (Figure 2). In this plot, it is easier to distinguish the gaussian components and to specify them by means of any of the analytical procedures developed to characterize the parameters of multimodal frequency distributions (i.e. Hasselblad 1966; Bhattacharya 1967).

If desired, it is possible to compute the twice part of the smoother: use `log` to log output, `list` the results of the smoothing, and then combine (with a word processor or editor) with the original data sequence by `infil`ing the data and `merg`ing with the original. Thereafter, the rough is generated by subtracting the smooth from the data and the result is smoothed by `sm4253eh`. Finally, the log file containing the list of smoothed rough values is combined with that containing the first smooth (in ASCII format), translated to Stata to add them, and finally compute the 4253EH,twice smoother results. In the next version of this smoother, we plan to produce the twice procedure automatically.

We are grateful to Dr. T. Anagnoson, who kindly sent us a copy of the do-files included in the Stata student version diskette containing Dr. Hamilton’s smoothing programs and to Dr. D. C. Hoaglin for sending a collection of his most recent papers on exploratory procedures (including resistant nonlinear smoothing).

[Editors note: I created a data set based on Table 1 for your use. It is called `fishdata.dta` and is found on the STB diskette. I suggest creating a duplicate variable for the one to be smoothed (i.e., `freq`). The program drops it; however, retaining the original may be useful. For example, create a duplicate of `freq` by `gen freq1=freq`. The program does not produce a graph—it simply produces the variables for graphing. The following command will create an appropriate graph, using the duplicate variable we made to observe both the original values and the smooth:

```
graph smooth freq1 timendx, xlab ylab(0,3,6,9,12,15) c(l) sort
```

You may wish to compare the resultant graph with a cubic spline of the original variable values:

```
graph freq length, xlab ylab c(s) bands(8)
```

Note that they are nearly alike.]

References

Alejo-Plata, Ma. del C., M. E. Laguna-Marin, and P. Ramirez-Tlalpan. 1989. *Estudio de algunos aspectos biologicos de Oreochromis mossambicus (Osteichthyes: Cichlidae) en la laguna “El Rodeo” Estado de Morelos.* Bachelor’s Thesis E.N.E.P. Zaragoza, U.N.A.M., Mexico, p. 130.

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

Curts, J. B. 1986. Teaching college biology students the simple linear regression model using an interactive microcomputer graphics software package. *Dissertation Abstracts International*, Vol. 46 (7 sec A).

Davis, J. C. 1973. *Statistics and Data Analysis in Geology.* New York: John Wiley & Sons, pp. 222–231.

DeLeon, R. and T. Anagnoson. 1991. sed1: Stata and the four R's of EDA. *Stata Technical Bulletin* 1: 13–17.

Geiger, P. 1991. sbe4: Further aspects of RIA analysis. *Stata Technical Bulletin* 5: 7–10.

Hamilton, L. C. 1990a. *Modern Data Analysis.* Pacific Grove, CA: Brooks/Cole, pp. 46–52.

——. 1990b. *Statistics with Stata.* Pacific Grove, CA: Brooks/Cole, pp. 147–149.

Hasselblad, V. 1966. Estimation of parameters for a mixture of normal frequency distributions. *Technometrics* 8(3): 431–444.

Himes, J. H. and D. C. Hoaglin. 1989. Resistant cross-age smoothing of age-specific percentiles for growth reference data. *American Journal of Human Biology* 1: 165–173.

Laurec, A. and B. Mesnil. 1987. Analytical investigations of errors in mortality rates estimated from length distributions of catches. In D. Pauly and G. R. Morgan, (eds.) *Length-Based Methods in Fisheries Research.* ICLARM Conference Proceedings 13: 239–282.

Salgado-Ugarte, I. H. 1991. El analisis exploratorio de datos en las poblaciones de peces. *Fundamentos y aplicaciones*, E.N.E.P. Zaragoza, U.N.A.M., pp. 57–85.

Siegel, A. F. 1988. *Statistics and Data Analysis.* Singapore: John Wiley & Sons, pp. 391–408.

Tukey, J. W. 1977. *Exploratory Data Analysis.* Reading, MA: Addison–Wesley, Ch. 7.

Velleman, P. F. 1980. Definition and comparison of robust nonlinear data smoothing algorithms. *Journal of the American Statistical Association* 75: 609–615.

——. 1982. Applied nonlinear smoothing. In S. Leinhardt, S. (ed.), *Sociological Methodology*, San Francisco: Jossey-Bass, pp. 141–178.

Velleman, P. F. and D. C. Hoaglin. 1981. *Applications, Basics, and Computing of Exploratory Data Analysis.* Duxbury Press, pp. 441-463.

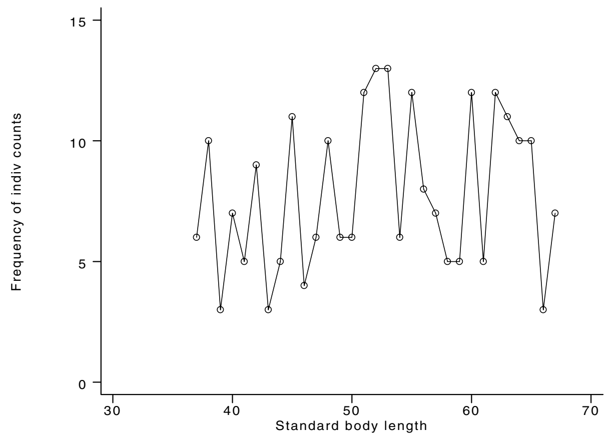

Figure 1

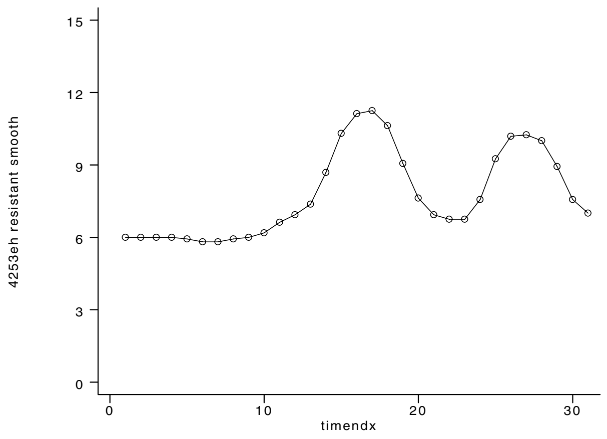

Figure 2

sg1.2 Nonlinear regression command

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011)-44-81-740 3119

Following Danuso's (1991) nonlinear regression program, I provide an enhanced, non-menu-driven command `nl`. Use of the program requires Stata 3.0. The syntax is

```
nl fcn depvar [varlist] [weight] [if exp] [in range] [,
      level(#)
      init(...) lnlsq(#) leave eps(#)
      nolog trace iterate(#)
      fcn_options ]

nlpred newvar [if exp] [in range] [, resid ]

nlinit # parameter_list
```

`aweight`s and `fweight`s are allowed.

Description

`nl` fits an arbitrary nonlinear function to the dependent variable *depvar* by least squares. You provide the function itself in a separate program with a name of your choosing, except that the first two letters of the name must be `nl`. *fcn* refers to the name of the function without the first two letters. For example, you type '`nl nexpgr` ...' to estimate with the function defined in the program `nlnexpgr`.

`nl` typed without arguments redisplays the results of the last estimation.

`nl` shares most of the features of other estimation commands (see [4] estimate). `predict`, however, may not be used after `nl`—use `nlpred` instead. `correlate, _coef` may be used, but you may not use `test`.

`nl` should be viewed as work in progress: the fitting process is iterative (modified Gauss-Newton) and there can be convergence problems. Accurate initial parameter estimates are desirable.

`nlpred` will calculate predicted values and residuals after `nl`.

`nlinit` is useful when writing nlfcns.

Options

`level(`*#*`)` specifies the significance level, in percent, for confidence intervals of the coefficients; see [4] estimate.

`init(`...`)` specifies initial values for parameters that are to be used to override the default initial values. Examples are provided below.

`lnlsq(`*#*`)` fits the model defined by nlfcn using "log least squares," defined as least squares with shifted lognormal errors. In other words, $\ln(\textit{depvar} - \#)$ is assumed normally distributed. Sums of squares and deviance are adjusted to the same scale as *depvar*.

`leave` leaves behind after estimation a set of new variables with the same names as the estimated parameters containing the derivative of $\mathrm{E}(y)$ with respect to the parameter.

`eps(`*#*`)` specifies the convergence criterion for successive parameter estimates and for the residual sum of squares. Default: 1e-5 (.00001).

`nolog` suppresses the iteration log.

`trace` expands the iteration log to provide more details, including values of the parameters at each step of the process.

`iterate(`*#*`)` specifies the maximum number of iterations before giving up and defaults to 100.

fcn_options refer to any options allowed by nlfcn.

`resid` tells `nlpred` to calculate residuals rather than predicted values.

Remarks

`nl` fits an arbitrary nonlinear function to the dependent variable *depvar* by least squares. The specific function is specified by writing an nlfcn, described below. The values to be fitted in the function are called the parameters.

The fitting process is iterative (modifed Gauss-Newton). It starts with a set of initial values for the parameters (guesses as to what the values will be and which you also supply) and finds another set of values that fit the function even better. Those are then used as a starting point and another improvement is found, and the process continues until no further improvement is possible.

nlfcns

`nl` uses the function defined by nlfcn. nlfcn has two purposes: to identify the parameters of the problem and set default initial values, and to evaluate the function for a given set of parameter estimates.

For instance, you have variables y and x in your data and wish to fit a negative-exponential growth curve with parameters B_0 and B_1:

$$y = B_0 \times (1 - e^{-B_1 x})$$

First, you write a program to calculate the predicted values:

```
program define nlnexpgr
        if "`1´" == "?" {                     /* if query call ...            */
                mac def S_1 "B0 B1"           /*    declare parameters        */
                mac def B0=1                  /*    and initialize them       */
                mac def B1=.1
                exit
        }
        replace `1´=$B0*(1-exp(-$B1*x))       /* otherwise, calculate function */
end
```

To estimate the model, you type '**nl nexpgr y**'. **nl**'s first argument specifies the name of the function, although you do not type the **nl** prefix. You type **nexpgr**, meaning the function is **nlnexpgr**. **nl**'s second argument specifies the name of the dependent variable. Replicating the example in the SAS manual (1985, 588–590):

```
. use sasxmpl1
. nl nexpgr y
(obs = 20)
Iteration 0:  residual SS =  .1999027
Iteration 1:  residual SS =  .0026142
Iteration 2:  residual SS =  .0005769
Iteration 3:  residual SS =  .0005768
  Source |       SS       df       MS               Number of obs =        20
---------+------------------------------            F(  2,    18) = 275732.74
   Model |  17.6717234     2  8.83586172            Prob > F      =    0.0000
Residual |   .00057681    18  .000032045            R-square      =    1.0000
---------+------------------------------            Adj R-square  =    1.0000
   Total |  17.6723003    20  .883615013            Root MSE      =  .0056608
                                                    Res. dev.     =  -152.317
(nexpgr)
------------------------------------------------------------------------------
       y |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
------------------------------------------------------------------------------
      B0 |   .9961885   .0016138    617.303   0.000       .9927981    .9995789
      B1 |   .0419539   .0003983    105.346   0.000       .0411172    .0427906
------------------------------------------------------------------------------
 (SE´s, P values, CI´s, and correlations are asymptotic approximations)
```

Notice that the initial values of the parameters were provided in the **nlnexpgr** program. You can, however, override these initial values on the **nl** command line. To estimate the model using .5 for the initial value of **B0** rather than 1, you can type '**nl nexpgr y, init(B0=.5)**'. To also change the initial value of **B1** from .1 to .2, you type '**nl nexpgr y, init(B0=.5, B1=.2)**'.

The outline of all nlfcns is the same:

```
program define nlfcn
        if "`1´" == "?" {
                mac def S_1 "parameter names"
                (initialize parameters)
                exit
        }
        replace `1´ = ...
end
```

On a query call, indicated by **`1´** being "?", the nlfcn is to place the names of the parameters in the global macro **S_1** and initialize the parameters. Parameters are stored as macros, so if nlfcn declares that the parameters are **A**, **B**, and **C** (via '**mac def S_1 "A B C"**'), it must then place initial values in the corresponding parameter macros **A**, **B**, and **C** (via '**mac def A=0**', '**mac def B=1**', etc.). After initializing the parameter macros, it is done.

On a calculation call, **`1´** does not contain "?"; it instead contains the name of a variable that is to be filled in with the predicted values. The current values of the parameters are stored in the macros previously declared on the query call (e.g., **$A**, **$B**, and **$C**).

Example

You wish to fit the CES production functions defined by

$$\ln(q) = B_0 + A\ln\Big((D)l^R + (1-D)k^R\Big)$$

where the parameters to be estimated are B_0, A, D, and R. q, l, and k refer to total output and labour and capital inputs. In your data, you have the variables **lnq**, **labour**, and **capital**. The nlfcn is

```
program define nlces
        if "`1´" == "?" {
                mac def S_1 "B0 A D R"
                mac def B0 = 1
                mac def A = -1
                mac def D = .5
                mac def R = -1
                exit
        }
        replace `1´=$B0 + $A*ln($D*labour^$R + (1-$D)*capital^$R)
end
```

Again using data from the SAS manual (1985, 591–592):

```
. use sasxmpl2

. nl ces lnq
(obs = 30)

Iteration 0:  residual SS =  37.09651
Iteration 1:  residual SS =  35.48615
Iteration 2:  residual SS =  22.69042
Iteration 3:  residual SS =  1.845374
  (output omitted)
Iteration 19:  residual SS =  1.761039
  Source |       SS       df       MS                  Number of obs =       30
---------+------------------------------               F(  3,    26) =   292.96
   Model |  59.5286148     3  19.8428716               Prob > F      =   0.0000
Residual |  1.76103929    26   .06773228               R-square      =   0.9713
---------+------------------------------               Adj R-square  =   0.9680
   Total |  61.2896541    29  2.11343635               Root MSE      = .2602543
                                                       Res. dev.     = .0775148
(ces)
------------------------------------------------------------------------------
     lnq |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
------------------------------------------------------------------------------
      B0*|   .1244882   .0783432      1.589   0.124      -.0365486    .2855251
       A |   -.336291   .2721672     -1.236   0.228      -.8957387    .2231568
       D |   .3366743   .1361148      2.473   0.020       .0568863    .6164623
       R |  -3.011047   2.323489     -1.296   0.206      -7.787048    1.764954
------------------------------------------------------------------------------
* Parameter taken as constant term in model & ANOVA table
 (SE´s, P values, CI´s, and correlations are asymptotic approximations)
```

If the nonlinear model contains a constant term, `nl` will find it and indicate its presence by placing an asterisk next to the parameter name when displaying results. In the output above, `B0` is a constant. (`nl` determines that a parameter `B0` is a constant term because the partial derivative $f = \mathrm{dE}(y)/\mathrm{d}$`B0` has a coefficient of variation (s.d./mean) less than `eps()`. Usually, $f = 1$ for a constant, as it does in this case.)

`nl`'s output closely mimics that of `regress`; see [5s] regress. The model F test, R-square, sums of squares, etc., are calculated as `regress` calculates them, which means in this case that they are corrected for the mean. If no "constant" is present, as was the case in the negative-exponential growth example previously, the usual caveats apply to the interpretation of the F and R-square statistics; see comments and references in Goldstein (1992).

When making its calculations, `nl` creates the partial derivative variables for all the parameters, giving each the same name as the corresponding parameter. Unless you specify `leave`, these are discarded when `nl` completes the estimation. Therefore, your data must not have data variables that have the same names as parameters. I recommend using uppercased names for parameters and lowercased names (as is common) for variables.

After estimating with `nl`, typing '`nl`' by itself will redisplay previous estimates. Typing '`correlate, _coef`' will show the asymptotic correlation matrix of the parameters, and typing '`nlpred myvar`' will create new variable `myvar` containing the predicted values. Typing '`nlpred res, resid`' will create `res` containing the residuals.

nlfcn's have a number of additional features that are described in *More on nlfcns* below.

Log-normal errors

A nonlinear model with identically normally distributed errors may be written

$$y_i = f(x_i, \beta) + u_i, \qquad u_i \sim \mathrm{N}(0, \sigma^2) \tag{1}$$

for $i = 1, \ldots, n$. If the y_i are thought to have a k-shifted lognormal instead of a normal distribution, that is, $\ln(y_i - k) \sim \mathrm{N}(\zeta_i, \tau^2)$, and the systematic part $f(x_i, \beta)$ of the original model is still thought appropriate, the model becomes:

$$\ln(y_i - k) = \zeta_i + v_i = \ln\bigl(f(x_i, \beta) - k\bigr) + v_i, \quad v_i \sim \mathrm{N}(0, \tau^2) \tag{2}$$

This model is estimated if `lnlsq(`*k*`)` is specified.

If model (2) is correct, the variance of $(y_i - k)$ is proportional to $\bigl(f(x_i, \beta) - k\bigr)^2$. Probably the most common case is $k = 0$, sometimes called "proportional errors" since the standard error of y_i is proportional to its expectation, $f(x_i, \beta)$. Assuming the value of k is known, (2) is just another nonlinear model in β and it may be fitted as usual. However, we may wish to compare the fit of (1) with that of (2) using the residual sum of squares or the deviance D, $D = -2 \times$ log-likelihood, from each model. To do so, we must allow for the change in scale introduced by the log transformation.

Assuming, then, the y_i to be normally distributed, Atkinson (1985, 85–87, 184), by considering the Jacobian $\prod |\partial \ln(y_i - k)/\partial y_i|$, showed that multiplying both sides of (2) by the geometric mean of $y_i - k$, $\dot{y}$, gives residuals on the same scale as those of y_i. The geometric mean is given by

$$\dot{y} = e^{n^{-1} \sum \ln(y_i - k)}$$

which is a constant for a given dataset. The residual deviance for (1) and for (2) may be expressed as

$$D(\widehat{\beta}) = \bigl(1 + \ln(2\pi\widehat{\sigma}^2)\bigr)n \tag{3}$$

where $\widehat{\beta}$ is the maximum-likelihood estimate (MLE) of β for each model and $n\widehat{\sigma}^2$ is the RSS from (1), or that from (2) multiplied by $\dot{y}^2$.

Since (1) and (2) are models with different error structures but the same functional form, the arithmetic difference in their RSS or deviances is not easily tested for statistical significance. However, if the deviance difference is "large" (> 4, say), one would naturally prefer the model with the smaller deviance. Of course, the residuals for each model should be examined for departures from assumptions (nonconstant variance, non-normality, serial correlations, etc.) in the usual way.

Example

Consider alternatively modeling

$$\mathrm{E}(y_i) = 1/(C + Ae^{Bx_i}) \tag{4}$$

$$\mathrm{E}(1/y_i) = \mathrm{E}(y_i') = C + Ae^{Bx_i} \tag{5}$$

where C, A, and B are parameters to be estimated. We will use the data $(y, x) = (.04, 5)$, $(.06, 12)$, $(.08, 25)$, $(.1, 35)$, $(.15, 42)$, $(.2, 48)$, $(.25, 60)$, $(.3, 75)$, and $(.5, 120)$ (Danuso 1991).

Model	C	A	B	RSS	Deviance
(4)	1.781	25.74	-.03926	-.001640	-51.95
(4) with `lnlsq(0)`	1.799	25.45	-.04051	-.001431	-53.18
(5)	1.781	25.74	-.03926	8.197	24.70
(5) with `lnlsq(0)`	1.799	27.45	-.04051	3.651	17.42

There is little to choose between the two versions of the logistic model (4), whereas for the exponential model (5) the fit using `lnlsq(0)` is much better (a deviance difference of 7.28). The reciprocal transformation has introduced heteroscedasticity into y_i' which is countered by the proportional errors property of the lognormal distribution implicit in `lnlsq(0)`. The deviances are not comparable between the logistic and exponential models because the change of scale has not been allowed for, although in principle, it could be.

Weights

Weights are specified the usual way—analytic and frequency weights are supported; see [4] weights. Use of analytic weights implies that the y_i have different variances. Model (1) may therefore be rewritten

$$y_i = f(x_i, \beta) + u_i, \qquad u_i \sim \mathrm{N}(0, \sigma^2/w_i) \tag{1a}$$

where w_i are (positive) weights, assumed known and normalized such that their sum equals the number of observations. The residual deviance for (1a) is

$$D(\widehat{\beta}) = \bigl(1 + \ln(2\pi\widehat{\sigma}^2)\bigr)n - \sum \ln(w_i) \tag{3a}$$

(compare with equation 3), where

$$n\widehat{\sigma}^2 = \text{RSS} = \sum w_i\bigl(y_i - f(x_i,\widehat{\beta})\bigr)^2$$

Defining and fitting a model equivalent to (2) when weights have been specified as in (1a) is not straightforward and has not been attempted. Thus, deviances using and not using the `lnlsq()` option may not be strictly comparable when analytic weights (other than 0 and 1) are used.

Errors

`nl` will stop with error 196 if an error occurs in your nlfcn program and it will report the error code raised by nlfcn.

`nl` is reasonably robust to the inability of nlfcn to calculate predicted values for certain parameter values. `nl` assumes that predicted values can be calculated at the initial value of the parameters. If this is not so, an error message is issued with return code 480.

Thereafter, as `nl` changes the parameter values, it monitors nlfcn's returned predictions for unexpected missing values. If detected, `nl` backs up. That is, `nl` finds a linear combination of the previous, known-to-be-good parameter vector and the new, known-to-be-bad vector, a combination where the function can be evaluated, and continues its iterations from that point.

`nl` does require, however, that once a parameter vector is found where the predictions can be calculated, small changes to the parameter vector can be made in order to calculate numeric derivatives. If a boundary is encountered at this point, an error message is issued with return code 481.

When specifying `lnlsq()`, an attempt to take logarithms of $y_i - k$ when $y_i \le k$ results in an error message with return code 482.

If `iterate()` iterations are performed and estimates still have not converged, results are presented with a warning and the return code set to 430.

General comments on fitting nonlinear models

In many cases, achieving convergence is problematic. For example, a unique maximum-likelihood (minimum-RSS) solution may not exist. A large literature exists on different algorithms that have been used, on strategies for obtaining good initial parameter values, and on tricks for parameterizing the model to make its behavior as "linear-like" as possible. Selected references are Kennedy and Gentle (1980, ch. 10) for computational matters, and Ross (1990) and Ratkowsky (1983) for all three aspects. Much of Ross's considerable experience is enshrined in the computer package MLP (Ross 1987), an invaluable resource. Ratkowsky's book is particularly clear and approachable, with useful discussion on the meaning and practical implications of "intrinsic" and "parameter-effects" nonlinearity. An excellent general text, though (in places) not for the mathematically faint-hearted, is Gallant (1987).

The success of `nl` will be enhanced if care is paid to the form of the model fitted, along the lines of Ratkowsky and Ross. For example, Ratkowsky (1983, 49–59) analyses three possible 3-parameter "yield-density" models for plant growth:

$$\text{E}(y_i) = \begin{cases} (\alpha + \beta x_i)^{-1/\theta} \\ (\alpha + \beta x_i + \gamma x_i^2)^{-1} \\ (\alpha + \beta x_i^{\phi})^{-1} \end{cases}$$

All three models give similar fits. However, he shows that the second formulation is dramatically more "linear-like" than the other two and therefore has better convergence properties. In addition, the parameter estimates are virtually unbiased and normally distributed and the asymptotic approximation to the standard errors, correlations and confidence intervals is much more accurate than for the other models. Even within a given model, the way the parameters are expressed (e.g., ϕ^{x_i} or $e^{\theta x_i}$) affects the degree of linear-like behavior.

My advice is that even if you cannot get a particular model to converge, don't give up. Experiment with different ways of writing it or with slightly different alternative models that also fit well.

More on nlfcns

Note that the syntax for `nl` is

`nl` *fcn depvar* [*varlist*] [...] [, ... *fcn_options*]

The syntax for an nlfcn is

nlfcn {*varname* | ?} [*varlist*] [, *fcn_options*]

The *varlist*, if specified with `nl`, will be passed to nlfcn along with any options not intended for `nl`. Thus, it is possible to write nlfcns that are quite general.

When nlfcn is called with a `?`, the *varlist* and *fcn_options*, if any, are still passed. In addition, `$S_E_depv` contains the identity of the dependent variable; `$S_E_if` and `$S_E_in` contain the `if` *exp* and `in` *range* specified on the `nl` command line; and `$S_E_wgt` and `$S_E_exp` contain the weight and expression.

nlfcn is required to post the names of the parameters to `S_1` and to provide default initial values for all the parameters. In addition, it may post up to two titles in `S_2` and `S_3` that will be subsequently used to title the output. The `S_E_` macros provide useful information for filling in titles and generating initial parameters estimates.

When nlfcn is called without a `?`, it is required to calculate the predicted values conditional on the current value of the parameters. Note that nlfcn is not required to process `if` *exp* or `in` *range*. Restriction to the estimation sample will be handled by `nl`.

Thus, at the beginning of this insert, I gave an example for calculating a negative-exponential growth model. A better version of the nlfcn would have been

```
program define nlnexpgr
        if "`1´" == "?" {
                mac def S_1 "B0 B1"
                mac def B0=1
                mac def B1=.1
                mac def S_2 "negative-exp. growth"
                mac def S_3 "$S_E_depv = B0*(1-exp(-B1*`2´))"
                exit
        }
        replace `1´=$B0*(1-exp(-$B1*`2´))
end
```

This version would title the output and allow the independent variable to be specified on the `nl` command line:

```
. nl nexpgr y xval
```

An even more sophisticated version of `nlnexpgr` might use `S_E_depv`, `` `2´ ``, `S_E_if`, and `S_E_in` to generate more reasonable starting values of `B0` and `B1`.

nlinit

`nlinit` is intended for use by nlfcns. Its syntax is

`nlinit` *# parameter_list*

`nlinit` initializes each parameter in *parameter_list* to contain *#*. For example:

```
nlinit 0 A B C
nlinit 1 D E
```

Saved Results

`nl` saves in the system `S_#` macros:

S_1	number of observations	S_7	R-square
S_2	model sum of squares	S_8	adjusted R-square
S_3	model degrees of freedom	S_9	residual root mean square
S_4	residual sum of squares	S_10	residual deviance
S_5	residual degrees of freedom	S_11	geometric mean $(y-k)^2$ if `lnlsq()`, otherwise 1
S_6	model F statistic	S_12	0 if convergence failed, otherwise 1

Note that `S_1` through `S_9` correspond to the successive elements of `_result()` following `regress`; see [5s] regress.

The final parameter estimates are available in the parameter macros defined by nlfcn. The standard errors of the parameters are available through `_se[`*parameter*`]`; see [2] coefficients.

`nlpred` and `nlinit` save nothing in `S_#` macros or `_result()`.

References

Atkinson, A. C. 1985. *Plots, Transformations and Regression.* Oxford: Oxford Science Publications.

Danuso, F. 1991. sg1: Nonlinear regression command. *Stata Technical Bulletin* 1: 17–19.

Gallant, A. R. 1987. *Nonlinear Statistical Models.* New York: John Wiley & Sons.

Goldstein, R. 1992. srd7: Adjusted summary statistics for logarithmic regressions. *Stata Technical Bulletin* 5: 17–21.

Kennedy, W. J., Jr. and J. E. Gentle. 1980. *Statistical Computing.* New York: Marcel Dekker.

Ratkowsky, D. A. 1983. *Nonlinear Regression Modeling.* New York: Marcel Dekker.

Ross, G. J. A. 1987. *MLP User Manual, release 3.08.* Oxford: Numerical Algorithms Group.

——. 1990. *Nonlinear Estimation.* New York: Springer-Verlag.

SAS Institute Inc. 1985. *SAS User's Guide: Statistics, Version 5 Edition.* Cary, NC.

sqv3 Wald and Atkinson's R extensions to logistic

Joseph Hilbe, Editor, STB, FAX 602-860-1446

Wald and Atkinson's partial correlation statistics for logistic regression were provided in the previously published `logiodd2.ado` command (Hilbe 1991). The new 3.0 `logistic` command does not incorporate these statistics; hence, I have provided an extension to `logistic` called `lwald`. Type `lwald` after using `logistic` as you would other extensions, for example `lfit`, `lstat` and so forth. No variable names or options are required. The following results are provided for each coefficient in the model: Wald statistic, χ^2 significance of Wald, and Atkinson's R (partial correlation).

A variable's R value is reported as 0.000 if its Wald statistic is less than 2 (see Atkinson 1980). Moreover, negative coefficients are given negative R values. I have not included any adjustment for categorical variables, although creation of appropriate design or indicator variables using `tab` *var*`, gen(`*var*`)` should solve the problem.

I advise caution when using Wald's test p-values—or t-test p-values, for that matter—when selecting variables for fitting a model. Wald's test proves erratic if there is collinearity between predictors or if there exists extreme values for the coefficients. In general, use of the likelihood-ratio test is preferable and more robust when selecting model predictors.

The basic formula for calculating the Wald statistic is

$$Wald_i = \left(\frac{\beta_i}{\sigma_{\beta_i}}\right)^2$$

where

$$\sigma_{\beta_i} = \frac{|\beta|}{\sqrt{F}}$$

Atkinson's R is calculated as

$$R_a = \sqrt{\frac{W_i - 2}{2|l_0|}}$$

where l_0 is the intercept log likelihood.

`lwald` ado and help files are found on the STB diskette.

References

Atkinson, A. C. 1980. A note on the generalized information criterion for choice of a model. *Biometrika* 67: 413–418.

Hilbe, J. 1991. sqv1.3: An enhanced logistic regression program. *Stata Technical Bulletin* 4: 16–18.

sts2 Using Stata for time series analysis

Sean Becketti, Federal Reserve Bank of Kansas City

[The programs reported here, and included on the STB-7 diskette, require Stata 3.0—Ed.]

1. Introduction

Many Stata users switch to another software package for time series analysis. Some switch because they require specialized tests or procedures not readily available in Stata. Others switch because they believe a software package designed to handle time series will be more convenient to use than Stata.

In fact, Stata is well-adapted to performing time series analyses including fairly sophisticated statistical tests. I routinely use Stata for analyses that others tell me can be performed only by such programs as PC-GIVE, RATS, SAS, or TSP. In my opinion, Stata's data handling capabilities and powerful programming tools make it the equal, if not the better, of these more-specialized programs.

This article presents a set of ado-files for time series analysis. The next section presents six simple utility programs that are generally useful for time series applications. The following sections present three Stata programs for particular time series applications. The sequence moves from the simplest, most frequently used program to programs that perform more advanced analyses. These programs demonstrate Stata's ability to perform time series analysis. However, this suite of programs does not cover all, or even most, of the needs of a practicing time series analyst. Extensions to this suite are discussed, some of which may appear in later STBs.

No program is perfect, and there are enhancements that would make Stata even more useful for time series analysis. Some of the changes are modest and could easily be incorporated in a future upgrade. Other changes are more ambitious, and Stata users may well disagree on their design. The final section of this article discusses some suggested enhancements. It is my hope that other Stata users will consider these suggestions and communicate their comments and criticisms to the STB. Such an exchange may encourage the developers of Stata to incorporate some of the more important of these enhancements.

2. Utility programs for lags, differences, and growth

2.1 lag, lead, and dif

The lag operator, L, and the difference operator, Δ, are used extensively in time series analysis. The lag operator is defined by the relation

$$Lx_t \equiv x_{t-1},$$

that is, the lag of the variable x in period t is the variable x in period $t-1$. Variables can be lagged more than one time period by applying the lag operator multiple times, thus

$$L^2x_t \equiv L(Lx_t) = x_{t-2},$$

and

$$L^kx_t \equiv L^{k-1}(Lx_t) = x_{t-k}.$$

In other words, L^kx_t is the k-th lag of x.

The difference operator can be defined in terms of the lag operator:

$$\Delta x_t \equiv (1-L)x_t = x_t - x_{t-1}.$$

Thus the first difference of x, Δx_t, is the arithmetic difference between x in period t and x in period $t-1$. The difference operator can be composed in the same way as the lag operator:

$$\Delta^2 x_t \equiv \Delta(\Delta x_t) = x_t - 2x_{t-1} + x_{t-2}.$$

Note that

$$\Delta^2 x_t \neq x_t - x_{t-2}.$$

From these definitions, it is clear that Stata can easily generate lags and differences of series. For example, to create the first lag of the Gross National Product (GNP), we could type

```
. generate gnplag = gnp[_n-1]
```

and to create the first difference of GNP, we could type

```
. generate gnpdif = gnp - gnp[_n-1]
```

However it would be tedious to have to type all the commands needed to generate the many lags of variables used in a typical time series analysis. Moreover, it is useful to be able to generate arbitrary lags and differences from within Stata programs. The `lag` and `dif` programs address these needs.

The `lag` and `dif` programs are so simple, it is probably better to demonstrate them than to explain them. Thus:

```
. describe

Contains data from gnp.dta
  Obs:   160 (max=  5068)                 Quarterly data on GNP
 Vars:     4 (max=    99)
Width:    12 (max=   200)
  1. year         int    %8.0g            Year
  2. quarter      int    %8.0g    quarter  Quarter
  3. date         float  %9.0g            Date
  4. gnp          float  %9.0g            GNP
Sorted by:  year  quarter

. lag 4 gnp

. describe

Contains data from gnp.dta
  Obs:   160 (max=  5068)                 Quarterly data on GNP
 Vars:     8 (max=    99)
Width:    28 (max=   200)
  1. year         int    %8.0g            Year
  2. quarter      int    %8.0g    quarter  Quarter
  3. date         float  %9.0g            Date
  4. gnp          float  %9.0g            GNP
  5. L.gnp        float  %9.0g            L.gnp
  6. L2.gnp       float  %9.0g            L2.gnp
  7. L3.gnp       float  %9.0g            L3.gnp
  8. L4.gnp       float  %9.0g            L4.gnp
Sorted by:  year  quarter
Note:  Data has changed since last save

. dif gnp

. describe

Contains data from gnp.dta
  Obs:   160 (max=  5068)                 Quarterly data on GNP
 Vars:     9 (max=    99)
Width:    32 (max=   200)
  1. year         int    %8.0g            Year
  2. quarter      int    %8.0g    quarter  Quarter
  3. date         float  %9.0g            Date
  4. gnp          float  %9.0g            GNP
  5. L.gnp        float  %9.0g            L.gnp
  6. L2.gnp       float  %9.0g            L2.gnp
  7. L3.gnp       float  %9.0g            L3.gnp
  8. L4.gnp       float  %9.0g            L4.gnp
  9. D.gnp        float  %9.0g            D.gnp
Sorted by:  year  quarter
Note:  Data has changed since last save

. lag 2 D.gnp

. describe

Contains data from gnp.dta
  Obs:   160 (max=  5067)                 Quarterly data on GNP
 Vars:    11 (max=    99)
Width:    40 (max=   200)
  1. year         int    %8.0g            Year
  2. quarter      int    %8.0g    quarter  Quarter
  3. date         float  %9.0g            Date
  4. gnp          float  %9.0g            GNP
  5. L.gnp        float  %9.0g            L.gnp
  6. L2.gnp       float  %9.0g            L2.gnp
  7. L3.gnp       float  %9.0g            L3.gnp
  8. L4.gnp       float  %9.0g            L4.gnp
  9. D.gnp        float  %9.0g            D.gnp
 10. LD.gnp       float  %9.0g            LD.gnp
 11. L2D.gnp      float  %9.0g            L2D.gnp
Sorted by:  year  quarter
Note:  Data has changed since last save
```

Note that `lag` can generate many new variables while `dif` always generates exactly one new variable. In addition to `lag`, I have also written `lead`. '`lead 2 gnp`' creates `F.gnp` and `F2.gnp`, but so will '`lag -2 gnp`'. ('`lead -2 gnp`' will do the same thing as '`lag 2 gnp`', too.)

The syntax of these three commands is

`lag` [*#*] *varname* [`, suffix(`*string*`)`]
`lead` [*#*] *varname* [`, suffix(`*string*`)`]
`dif` [*#*] *varname* [`, suffix(`*string*`)`]

In all commands, if # is not specified, 1 is assumed. `lag` and `lead` create variable names like `L.gnp`, `L2.gnp`, `L3.gnp`, ..., `F.gnp`, `F2.gnp`, ..., and `dif` creates variable names like `D.gnp`, `D2.gnp`, and so on.

All commands have a `suffix` option that allows replacing the variable name with an arbitrary suffix. For example, '`lag 2 gnp, suffix(x)`' generates the variables called `L.x` and `L2.x`. This is useful in programs.

Useful conventions

In order for this suite of time series programs to work together smoothly, a number of conventions must be adopted. The conventions I have adopted are

1. The use of capital letters (such as `L` for lag) to denote operators;
2. The exclusive use of a period to separate operators from variable names.

There are a number of different conventions I could have adopted for naming lags and differences of variables. After a bit of experimentation, I settled on the current convention. My use of period allowed me to write another command, `dropoper`, that drops all variables that have an embedded period. Such derived variables have a way of accumulating in a data set, but with `lag`, `lead`, and `dif`, they are easily reconstructed so there is no reason why they shouldn't be easily eliminated. Moreover, the convention allows programs like `lag` and `lead` to drop variables freely. Typing '`lag gnp`' produces `L.gnp`. If, later, I decide I need two lags, '`lag 2 gnp`' knows it can drop `L.gnp` and recreate it because the convention makes clean the variable's derivation. And, it turns out that attaching a prefix rather than a suffix to the original variable name simplifies some of the programs that use `lag` and `dif`.

Another simple time series program that benefits from the adoption of sensible conventions is a program to calculate growth rates. Many of my colleagues have written short Stata programs to generate the growth rate of a variable. Program authors must decide which growth rate formula they prefer, whether to express the growth rates as a fraction or as a percent (fractions are more useful in succeeding calculations while percents are easier to read), and whether to annualize the growth rates. For example, one way to calculate an annual growth rate for quarterly GNP is

```
. generate ggnp = 4 * (log(gnp) - log(gnp[_n-1])
```

Alternatively,

```
. generate ggnp = (gnp/gnp[_n-1])^4 - 1
```

The '`4`' in these formulas is the appropriate factor for converting quarterly to annual growth rates. Since you are unlikely to use only quarterly data, it would be nice to have a program that annualized growth rates for any time series frequency. An obvious solution would be to pass the factor as an option. For example,

```
. growth gnp, period(4)
```

might generate `G.gnp`, the annualized growth rate of quarterly `GNP`. A more useful convention is to define a system macro, say `S_PERIOD`, that indicates the data frequency. Thus

```
macro define S_PERIOD "quarter"
```

indicates the current data set contains quarterly data. This macro can be used by any time series program that needs to know or to set the data frequency.

In fact, this worked out so well that, with more work, I wrote a `period` command to hide the fact that `S_PERIOD` needed to be set. My `period` command has the syntax

`period` { *#* | *word* | *# word* }

where *word* is `annual` (corresponding to $\# = 1$), `semiannual` (corresponding to 2), `quarterly` (4), `monthly` (12), `weekly` (52), and `daily` (325).

Thus, I can type '`period 4`' or '`period quarterly`' to indicate that my data is quarterly. In case I did not anticipate all my needs, I could even type '`period 1 hourly`' to say that each observation corresponds to an hour of the day, or '`period 24 hourly`', which again indicates hourly data, but that, in general, I want rates normalized to daily rates.

Once the period is set, the `growth` command can be used without options. The syntax of `growth` is

`growth` *varname* [`, period(`*#*`) ma(`*#*`) noannual log percent`]

The `period()` option is an alternative to specifying `period` ahead of time; I never use it. `ma()` generates moving averages; so with quarterly data, I can type '`growth gnp`' to obtain quarterly growth in GNP normalized to annualized rates, or '`growth gnp, ma(4)`' to obtain the moving average yearly rates. In either case, the resulting variable is called `G.gnp`. `growth`, like `lag`, `lead`, and `dif`, is quite willing to replace the `G.gnp` variable.

3. A common problem: finding the optimal lag length

A practical problem in time series analysis is determining the appropriate number of lags of a variable to include in an equation. This problem is a special case of the problem of selecting the best set of regressors, and some of the tests developed for this problem are also helpful in determining optimal lag length.

Formally, we want to estimate the equation

$$A(L)y_t = \beta_0 + \beta_1 x_{1t} + \ldots + \beta_k x_{kt} + \epsilon_t$$

where $A(L)$ is a polynomial in the lag operator, that is,

$$\begin{aligned} A(L)y_t &\equiv (1 - \alpha_1 L - \alpha_2 L^2 - \ldots - \alpha_p L^p)y_t \\ &= y_t - \alpha_1 y_{t-1} - \alpha_2 y_{t-2} - \ldots - \alpha_p y_{t-p}. \end{aligned}$$

Expanding $A(L)$ in the original equation and rearranging terms gives

$$y_t = \alpha_1 y_{t-1} + \ldots + \alpha_p y_{t-p} + \beta_0 + \beta_1 x_{1t} + \ldots + \beta_k x_{kt} + \epsilon_t$$

While theory may suggest the set of x's that should enter this equation, it is rare that theory indicates p, the appropriate lag length. If too few lags are included, estimates of the coefficients will be inconsistent. If too many lags are included, the estimates will be inefficient.

`findlag` calculates the optimal values of p suggested by four widely used statistics: the root mean squared error (RMSE) of the estimated regression, Akaike's Information Criterion (AIC), Amemiya's Prediction Criterion (PC), and Schwarz's Criterion (SC). (Judge et. al. 1985, contains an excellent discussion of these criteria.)

For example, the following commands could be used to help determine the appropriate specification for a simple autoregression of the log of GNP.

```
. generate lgnp = log(gnp)
(29 missing values generated)
. findlag lgnp
RMSE   AIC   PC    SC     (obs=127)
---------------------
  3     3     3     2
```

In this example, one to four lags of log GNP are tried in the equation

$$A(L)\log \text{GNP}_t = \epsilon_t.$$

For each of the five estimated equations, the four statistics are calculated and the lag lengths that produced the optimal values of each statistic are reported. In this example, the Schwarz Criterion favors two lags while the other statistics indicate that three lags are required.

`findlag` has other options not shown in this example. Typing `help findlag` will display an explanation of them. It is worth noting that `findlag` uses `lag`.

4. The problem of nonstationarity: unit roots and cointegration

The majority of time series statistical techniques can be applied only to *stationary variables*. The concept of stationarity can be a little abstract. For most purposes, the simpler property of *covariance stationarity* will serve. (See Box and Jenkins 1976 or Granger and Newbold 1977 for a more rigorous explanation of stationarity.) A time series x_t is covariance stationary if

$$Ex_t = \mu$$

and

$$E(x_t-\mu)(x_{t-k}-\mu)=\gamma_k.$$

In plain English, x_t is covariance stationary (or just stationary, for short) if its mean is constant and if the covariance between x and the k-th lag of x (called the k-th autocovariance of x and denoted by γ_k) depends only on k, the distance in time between x_t and x_{t-k}.

A moment's reflection reveals that many interesting time series are nonstationary. For example, GNP does not have a constant mean. Rather it grows more or less steadily over time. This is true of most economic time series and most population measures. Even per capita measures of economic variables tend to grow steadily as societies become more affluent over time. Although many interesting variables are nonstationary in the sense of trending steadily up or down, the growth rates (or log first differences) of many of these variables are often stationary. Thus these stationary growth rates can be analyzed using standard time series techniques.

Stationarity is such a crucial characteristic that almost all time series analyses begin by checking variables for stationarity. A way of characterizing nonstationary variables that is convenient for testing is in terms of unit roots. If a time series can be modeled as

$$A(L)x_t=\epsilon_t$$

and ϵ_t is *white noise* (that is, a serially uncorrelated, stationary disturbance), then x_t is stationary if and only if the roots of the equation

$$0=A(z)$$
$$\Rightarrow 0=1-\alpha_1 z-\alpha_2 z^2-\ldots-\alpha_p z^p$$

are greater than one in modulus. If there are any *unit roots*, that is, if $z=1$ is a root of this equation, then x_t is nonstationary.

There are a number of ways to check for stationarity. You can inspect a timeplot of the variable to see if it "looks" stationary. Alternatively, you can plot the autocorrelations and partial autocorrelations of the variable to see if they decay rapidly enough to indicate stationarity. (The *sts1* programs `ac` and `pac`, Becketti 1992, produce these charts; I have updated them for this submission; see the on-line help.) Finally, you can test the hypothesis that $A(L)$ contains a unit root. (Dickey and Fuller 1979 pioneered formal tests for unit roots. Engle and Granger 1987 develop the related concept of cointegration which will be discussed below. Engle and Yoo 1987 provide useful tables of critical values for tests of both unit roots and cointegration.)

Let's consider the simplest possible case. The null hypothesis is that the time series x_t follows a *random walk*, that is,

$$x_t=x_{t-1}+\epsilon_t.$$

The alternative hypothesis is that x_t follows a stationary first-order autoregression

$$x_t=\rho x_{t-1}+\epsilon_t$$

where $|\rho|<1$. The null hypothesis can be restated as $\rho=1$ which implies that x_t has a unit root. By subtracting x_{t-1} from both sides of this equation, we obtain

$$\Delta x_t=(\rho-1)x_{t-1}+\epsilon_t.$$

The so-called Dickey–Fuller test estimates this equation by ordinary least squares and tests the null hypothesis $\widehat{(\rho-1)}=0$ against the alternative hypothesis $\widehat{(\rho-1)}<0$. The test statistic is the t-statistic on the estimated coefficient $\widehat{(\rho-1)}$. This statistic has a non-standard distribution under the null hypothesis. Critical values for the Dickey–Fuller statistic are tabulated in the articles by Dickey and Fuller and by Engle and Yoo. In some cases, the appropriate test statistic is derived from the regression

$$A(L)\Delta x_t=(\rho-1)x_{t-1}+\epsilon_t.$$

`dickey` calculates the Dickey–Fuller test. As an example, let's test whether the log of GNP has a unit root.

```
. dickey lgnp
(obs=130, constant, no trend)
  Lags   tau
-------------
    0    .087
    1    .074
    2   -.264
    3   -.375
    4    -.26
RMSE   AIC   PC    SC     (obs=126)
--------------------
  2     2     2     0
```

In order to reject the null hypothesis of a unit root at the 5 percent level, the test statistic `tau` should be less than -2.89. Thus, in this example, we cannot reject the hypothesis that the log of GNP is nonstationary. Note that `dickey` calls `findlag` to determine how many lags of Δx_t should be included in the Dickey–Fuller regression.

The final time series concept we will consider in this article is cointegration. Regressions between nonstationary variables are known to give spurious results: this is one of the reasons time series analysts check so carefully for stationarity before proceeding in an analysis. However, theory often suggests that some combinations of nonstationary variables should not drift too far apart. For example, some theories indicate that the money supply and the price level should have a definite relationship on average even though both these variables are nonstationary. Formally, nonstationary variables are *cointegrated* if a linear combination of the variables is stationary. Just as univariate time series analysis begins with tests for stationarity, multivariate time series analysis begins with cointegration tests.

`coint` performs the Engle–Granger test for cointegration. For this test, the null hypothesis is that the variables are *not* cointegrated. To run the test, a regression is run with one of the variables chosen as the left-hand-side variable. Then a Dickey–Fuller test is run on the estimated residuals from this initial regression. (Critical values for the cointegration test are in Engle and Yoo 1987.)

Many asset prices are closely linked by arbitrage even though none of the asset prices in isolation is stationary. The following listing illustrates the use of `coint` and shows that the prime rate and the 3-month Treasury bill yield are cointegrated.

```
. describe

Contains data from interest.dta
  Obs:   877 (max=  5186)                 Monthly data on interest rates
 Vars:     5 (max=    99)
  1. year         int    %8.0g            Year
  2. month        int    %8.0g    month   Month
  3. date         float  %9.0g            Date
  4. rprime       float  %9.0g            Prime rate
  5. rtb3         float  %9.0g            3-month T-bill yield
Sorted by:  year  month

. dickey rprime

(obs=515, constant, no trend)
  Lags   tau
-------------
    0   -1.72
    1   -2.68
    2   -2.12
    3   -2.01
    4   -1.89

RMSE   AIC   PC    SC     (obs=511)
---------------------
  4     4     4     2

. dickey rtb3

(obs=696, constant, no trend)
  Lags   tau
-------------
    0   -1.75
    1   -2.39
    2   -1.97
    3   -1.99
    4   -1.96

RMSE   AIC   PC    SC     (obs=692)
---------------------
  2     2     2     2

. coint rprime rtb3

(obs=515, constant, no trend)
  Lags   tau
-------------
    0   -6.39
    1   -7.84
    2   -6.16
    3   -5.77
    4   -5.12

RMSE   AIC   PC    SC     (obs=511)
---------------------
  4     4     4     4
```

The 5 percent critical value for this test is −3.37; thus, the test clearly rejects the null hypothesis of no cointegration.

5. Some suggested enhancements to Stata

The sections above illustrate just a few of the time series tools that Stata can provide. Other techniques I have incorporated in my library of Stata ado-files include programs for exponential smoothing, Cochrane–Orcutt correction of serial correlation, and estimation under autoregressive conditional heteroscedasticity (ARCH). This list can easily be extended without undue effort.

Nonetheless, aspects of Stata do make it more difficult than I'd like to handle time series. This section lists some of these problems along with some suggested enhancements to overcome these problems.

5.1 Date formats: a simple but amazing useful enhancement

One of the most frustrating gaps in Stata is the lack of date formats. The single most frequently used time series technique is to graph one or more series against time. Currently, Stata comes with a number of useful programs for creating and manipulating date variables. However, Stata contains no way to format these date variables for use in graphs or listings. My rough and ready solution is to create a variable called `date` that serves this purpose. For monthly data, I type

```
. generate date = year-1900 + (month-1)/12
```

and for quarterly data I use

```
. generate date = year-1900 + (quarter-1)/4
```

This is a stopgap, but it is better than nothing. All this rigmarole could be avoided if Stata defined date formats that would display the current date variables in a form readable by humans. Nothing fundamental in Stata would be affected by this change.

5.2 Expanded variable lists: a more ambitious enhancement

An obvious disadvantage of the `lag` program is the number of extra variables it creates. Typically these lags are needed only for one or two calculations. Furthermore these lags are easily calculated from the original variable at any step of the calculations. More subtle but equally important, it is impossible to construct dynamic forecasts with `predict` because the lagged forecasts are not used, that is, the temporal connection between the lagged variables and the current values is obscured.

Both these problems, and several others, could be solved if the notion of a variable list in Stata were expanded. Currently, a variable list must contain either all new variables or all existing variables. A more useful notation for lists of existing variables would allow (non-existent) leads and lags of existing variables to be included.

There are a number of complications that must be considered in designing the syntax for expanded variable lists. Without confronting them now, let me illustrate the idea with a simple example. In an improved Stata, we might be able to type

```
. regress y L.y L2.y x z L.z L2.z
```

or, even better,

```
. regress y-L2.y x z-L2.z
```

For more complicated examples than this one, some notation such as this would both reduce typing and conserve variable storage space. More importantly, Stata commands could recognize and exploit the temporal connections between the coefficients on the `y`'s (and the `z`'s) when constructing forecasts and test statistics.

5.3 Shoot the moon: linking Stata to a matrix language

Stata's multiple equation estimation techniques and matrix calculations are limited. Many modern approaches rely on systems estimation, matrix corrections to covariance matrices, and the like. All these could easily be accommodated without changing Stata if Stata's data sets and calculations could be linked to some other matrix programming language (MPL). If ado-files could offload these calculations to an MPL, then utilize the output from the MPL within Stata, Stata would essentially have no limits at all.

6. Call for comments

This article highlights the perspective of a generally happy, though sometimes frustrated Stata user. While I have a high regard for my own opinion, Stata will develop more productively if others contribute to the debate over useful programming conventions and future enhancements. I look forward to reading such contributions in future issues of the STB.

7. Summary of submitted programs

`ac`	Display correlogram	`growth`	Generate growth rate
`coint`	Test for cointegration	`lag`	Generate lags
`dickey`	Test for unit root	`lead`	Generate leads
`dif`	Generate differences	`pac`	Display partial autocorrelation plot
`dropoper`	Drop operator variables	`period`	Set period of time series data
`findlag`	Find optimal lag length		

8. References

Becketti, S. 1992. sts1: Autocorrelation and partial autocorrelation graphs. *Stata Technical Bulletin* 5: 27–28.

Box, G. E. P. and G. M. Jenkins. 1976. *Time Series Analysis: Forecasting and Control.* revised ed. Oakland, CA: Holden–Day.

Dickey, D. A. and W. A. Fuller. 1979. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association* 74: 427–31.

Engle, R. F. and C. W. J. Granger. 1987. Cointegration and error correction: Representation, estimation, and testing. *Econometrica* 55(2): 251–76.

Engle, R. F. and B. S. Yoo. 1987. Forecasting and testing in co-integrated systems. *Journal of Econometrics* 35(1): 143–159.

Granger, C. W. J. and P. Newbold. 1977. *Forecasting Economic Time Series.* Academic Press.

Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and Tsoung-Chao Lee. 1985. *The Theory and Practice of Econometrics.* 2d ed. New York: John Wiley & Sons.

tt4 Teaching ecology with Stata

James Taylor, Department of Zoology, University of New Hampshire

As noted in *tt1*, (Anagnoson and DeLeon 1991) and *tt2* (Macy 1991), the pedagogical use of computer programs for statistical analysis presents many problems. If program access is a “user friendly” graphical interface or menu, students are prone to proudly submit totally inappropriate analyses (e.g., a fifth-order polynomial fitted to a simple allometric relationship.) With more conventional access, all but the simplest procedures require cookbook recipes of cryptic commands and options all too susceptible to frustrating errors (a ‘1’ for an ‘l’, etc.) To a student, all statistics packages spew out a bewildering array of numbers, many of which are not germane to introductory level interpretation, and some of which are not strictly comparable (see 3 below for an example.) Finally, many discipline-specific analyses may require special tricks to get a general package to perform them.

All of these problems have arisen in our General Ecology laboratory, in which I and my associate attempt to get students to “do science” by recording and interpreting their own observations. As in the standard classic experiments approach, the methodology for each exercise is carefully defined. Our approach differs from the classic approach, in which a known result is to be obtained, in that there is no one correct result; students must interpret their own data in the light of competing hypotheses. In any one laboratory section, results and their correct interpretation may vary widely. We believe that our emphasis on interpretation closely approximates the way real ecologists work. We do have the advantage of having computers in the laboratory, so students can move from observation to analysis with assistance nearby.

We rely heavily on Stata’s graphics to aid the students, as we emphasize graphing as the first step in exploratory data analysis. We also introduce a few rudimentary statistics. However, as a review in the *Bulletin of the Ecological Society of America* recently pointed out, Stata is more of a programming language than a statistical package, and initial efforts with raw Stata, in menu or command line form, were not well received by students. Our solution was to write a series of ado-programs that achieve the following goals:

1. Combine several procedures into a single command. For example, `fitline.ado` computes regression statistics, graphs a linear regression, and displays the regression equation, R-squared, and significance level on the graph.
2. Selectively display the few statistics emphasized in the course. For example, variance is a statistic used throughout the course; it’s very simple to write an ado-file that adds variance to the statistics displayed by the `summarize` command, and avoid the bewildering detail of `summarize, detail`. Also, `fitline.ado` displays a few pertinent regression statistics right on the graph, without the detail produced with the `regress` command.
3. Correctly calculate statistics for comparative purposes. For example, `fitpower.ado` graphs a power curve using antilogs of the predicted values from linear regression of log-transformed data. It also calculates the R-squared for the fit of the nonlinear regression (which is not the same as the fit of the linear regression on log-transformed data.) This R-squared is directly comparable with that displayed by `fitline.ado` for the linear regression on the same data, and fits of the two models to the same data can be compared.

4. Make arcane commands more intuitive. For example,

   ```
   . display exp(n)
   ```

   means nothing to most students; they do understand

   ```
   . antilog n
   ```

   which calls the `antilog.ado` file encapsulating `display exp(`1´)`.

5. Implement discipline-specific analyses. In ecology, species-area and dominance-diversity curves are important tools. `esarea.ado` and `domdiver.ado` produce these graphs.

6. Make arcane options, such as axes maxima and labels, interactive. For example, careful comparison of different species-area or dominance-diversity graphs requires use of common axes maxima. The following ado-file for species-area curves illustrates interactive requests for input of axes maxima, as well as for an informative title. Note that defaults are provided for each choice. In all fairness, note also that the program name is as cryptic as any Stata command, even if one knows that S is a common symbol for species richness.

```
* Draw species-area curve
program define esarea
version 3.0
    ***         CLEAR DUMMY VARIABLES
capture drop V1
capture drop V2
capture drop V3
    ***         DEFINE VARIABLES & STANDARD AXIS LABELS
local varlist "required existing max(1)"
local options "*"
parse "`*´"
parse "`varlist´", parse(" ")
local a "l1(Cumulative Number of Species)"
local b "b2(Number of Samples/ Relative Area Sampled)"
local c "b1(Species-Area Curve)"
generate V2=sum(`1´)
    ***         ENTER AXIS MAXIMA
di
di "To control scale of the X-axis, enter a maximum value (must be >" _N ")"
di "  OR  Press <Enter> to let STATA select X-axis scales."
di "Enter X-axis maximum (or press <Enter>)..." _r(_xmax)
di
di "To control scale of the Y-axis, enter a maximum value (>" V2[_N] ")"
di "  OR  Press <Enter> to let STATA select Y-axis scales."
di "Enter Y-axis maximum (or press <Enter>)..." _r(_ymax)
    ***           USE DEFAULT MAXIMA, if necessary
if "`xmax´"=="" { local xmax=_N  }
if "`ymax´"=="" { local ymax =V2[_N]  }
confirm number `xmax´
confirm number `ymax´
    ***           GET GRAPH TITLE
di
di "Enter a title to label this graph,"
di "    OR press <Enter> to use " in red "   %_1  " in white " as a label."
di "Title..." _r(_label)
di
if "`label´"=="" { local label "`1´" }
local label "t2(`label´)"
    ***           ANCHOR X-AXIS AT 0, by adding a null observation.
local d=_N+1
quietly {
      set obs `d´
      generate V1=_n-1
      generate V3=V2[_n-1]
      replace V3=0 if _n==1
}
   ***     GRAPH WITH INTERACTIVES  `label´, `xmax´, and `ymax´
gr V3 V1, c(l) xla yla xscale(0,`xmax´) yscale(0,`ymax´) `a´ `b´ `c´ `label´ `options´
   drop V1 V2 V3
   capture drop in l
end
```

By making it easier for students to get appropriate results, and leaving less opportunity for inappropriate analysis, more time can be spent actually examining the results and thinking about what they mean. The use of original data makes the exercises more than cookbook procedures. We see no merit in teaching Stata (or any other) syntax to students. It is familiarity with some basic tools of analysis that we wish to teach, and Stata is just a means of providing those tools.

References

Anagnoson, T. and R. DeLeon. 1991. tt1: Teaching beginning students with Stata. *Stata Technical Bulletin* 4: 27–28.

Macy, M. 1991. tt2: Using "front-ends" for Stata. *Stata Technical Bulletin* 4: 28.

tt5 Simple chemical equilibrium

Paul Geiger, USC School of Medicine, pgeiger@vm.usc.edu

Equilibrium considerations in chemical reactivity determine the direction of a chemical reaction. These considerations are important in industrial processes as well as in understanding biochemical processes in cellular metabolism. The smaller the equilibrium constant, K_{eq}, the less the product derived from the reactants.

Consider a simple reaction of the type

$$A + B \rightleftharpoons C + D$$

where reactants are placed on the left of the double arrows and products are to the right in the scheme.

By chemical convention, the equilibrium constant has the form

$$K_{eq} = \frac{[C]_{eq}[D]_{eq}}{[A]_{eq}[B]_{eq}}$$

where $[A]_{eq}$, $[B]_{eq}$, $[C]_{eq}$, and $[D]_{eq}$ are the concentrations of A, B, C, and D at equilibrium.

If the initial concentrations (at time zero) of A, B, C, and D are denoted by $[A]_o$, $[B]_o$, $[C]_o$, and $[D]_o$ and the change in concentration that occurs to reach equilibrium is designated x, the following equations express concentrations at equilibrium:

$$\begin{aligned}[A]_{eq} &= ([A]_o - x)\\ [B]_{eq} &= ([B]_o - x)\\ [C]_{eq} &= ([C]_o + x)\\ [D]_{eq} &= ([D]_o + x)\end{aligned}$$

Substituting these values into the equation for K_{eq}, multiplying and rearranging leads to the quadratic equation:

$$(1 - K_{eq})x^2 + ([C]_o + [D]_o + K_{eq}[A]_o + K_{eq}[B]_o)x + [C]_o[D]_o - K_{eq}[A]_o[B]_o = 0$$

In order to make a do-file to carry out calculations with the quadratic formula let:

$$\begin{aligned}a &= 1 - K_{eq}\\ b &= [C]_o + [D]_o + K_{eq}[A]_o + K_{eq}[B]_o\\ c &= [C]_o[D]_o - K_{eq}[A]_o[B]_o\end{aligned}$$

Solving for x in $x = (-b \pm sqrt(b^2 - 4ac))/2a$ gives a pair of roots, but only the positive solution makes chemical sense. Calculation of a, b, and c need not be done separately but may help in teaching, particularly when a spreadsheet is used as in Atkinson et al. 1987.

The do-file `equil.do` supplied on the disk, generates `a`, `b`, `c`, `x` and `Aeq`, `Beq`, `Ceq`, `Deq` and `x_Ao`, from the variables `Ao`, `Bo`, `Co`, `Do`, and `Keq` provided in the file `equil.dta`. The calculated ratio, x/A_o (Stata variable `x_Ao`), provides the fraction of $[A]_o$ converted when various starting concentrations for reactants have been assumed. Of course, starting values can be typed in using the `input` command or generated using the `range` command. The `infile` command is used if a table of values has already been made using an editor.

Running the do-file after selecting the `equil.dta` file or typing in the K_{eq} and values for the reactants at time zero (`Keq`, `Ao`, `Bo`, `Co`, `Do`) allows the student to explore equilibrium easily and quickly and see how a reaction might be driven to completion or nearly so. The left half of the following table illustrates starting conditions with $K_{eq} = 0.01$ and with $[A]_o$ and $[B]_o$ supplied but no products present, that is, a reaction unfavorable to making products from reactants. The right half shows equilibrium conditions and `x_Ao` (x/A_o), the fraction of $[A]_o$ converted, say, to $[C]_{eq}$.

```
Key   Ao    Bo    Co    Do    Aeq    Beq    Ceq    Deq   x_Ao
-------------------------------------------------------------
.01   1      1    0     0     .91    .91    .09    .09    .09
.01   1     20    0     0     .64   19.6    .36    .36    .36
.01   1     40    0     0     .54   39.5    .46    .46    .46
.01   1     60    0     0     .47   59.5    .53    .53    .53
.01   1     80    0     0     .42   79.4    .58    .58    .58
.01   1    100    0     0     .38   99.4    .62    .62    .62
```

With up to 100 times more starting reactant B supplied, A has been forced to yield 62% product from reactant. A graph of `Ceq` vs. `Bo` illustrates how product tapers off even if 100 times more of reactant B than A is supplied (in Stata, type **`graph using equil`**, graph supplied on disk).

Now suppose that we desire to make C from A and that A is very expensive relative to B. A naive chemist might think, at first blush, to increase the concentration of A. The Stata user can check this method quickly by reversing the values of `Ao` and `Bo` from the table. The fraction of A converted, `x_Ao`, will be only 0.006. True, the absolute amount will be the same, but the chemist will have used 100 times more of an expensive starting material to get the same amount of product.

What about the influence of K_{eq}? This can be explored by changing all `Bo` values, say, to 100 and supplying a range of `Keq` values and then running the do-file. For instance, if $K_{eq} = 0.1$, ten times greater than before, the yield of $[C]_{eq}$ rises to 0.92 if $[A]_o = 1$ and $[B]_o = 100$, and so forth.

Other explorations of changes in the starting mix can also be made. Suppose there were contaminants, $[C]_o$ and/or $[D]_o$, in the starting materials? This situation also easily yields to Stata and the chemist user (exercise left to the reader). For instance, find the effect of the ratio of $[B]_o$ to $[D]_o$ on the conversion of A (valuable, remember) to C.

In biochemical systems, the problem of product build-up and the consequent reaction slowing with approach to equilibrium is often obviated by product removal. For instance, in the important energy yielding reaction system of glycolysis the enzyme aldolase produces glyceraldehyde phosphate and dihydroxyacetone phosphate from fructose-1,6-bis-phosphate, $K_{eq} = 10^{-4}$. The direction of equilibrium favors the fructose bis-phosphate (backward direction) but glyceraldehyde phosphate is removed by oxidation while an isomerase enzyme converts dihydroxyacetone phosphate to more of the glyceraldehyde phosphate. Thus the pathway goes forward to produce pyruvate which is used further in other energy yielding reactions for the cell.

Manipulating chemical equilibrium is highly useful in establishing assays for certain biochemical compounds. In the case of fructose bis-phosphate, Lowry and Passonneau (1974) used three different enzymes as reagents to determine it.

In the University of Southern California basic chemistry course, chemical equilibrium is taught from a standard textbook used for chemistry majors. Programmable calculators are not used but ordinary ones that provide such things as squares and square roots, simple statistics, etc., are permitted. The students, therefore, come closer to understanding principles by actually setting up and solving the quadratic equations that often appear in equilibrium calculations, a situation usually avoided by approximation in the old, slide-rule days. With Stata, however, or Student Stata it would seem that learning the nuances of chemical equilibria could be made much more efficient, particularly with the power to visualize the results of changes in reagent and product concentrations with Stata's graphics.

References that may be of interest in addition to Atkinson et al. and Lowry and Passonneau are included below.

References

Atkinson, D. E., et al. 1987. *Dynamic Models in Biochemistry. A Workbook of Computer Simulations Using Electronic Spreadsheets.* Menlo Park, CA: Benjamin/Cummings.

Benson, S. W. 1971. *Chemical Calculations. An Introduction to the Use of Mathematics in Chemistry.* 3d ed. New York: John Wiley & Sons.

Dickerson, R. E., et al. 1970. *Chemical Principles.* New York: W. A. Benjamin.

Lowry, O. H. and J. V. Passonneau. 1972. *A Flexible System of Enzymatic Analysis.* New York: Academic Press.

Segel, I. H. 1976. *Biochemical Calculations.* 2d ed. New York: John Wiley & Sons.

zz1 Cumulative index for STB-1—STB-6

[an] Announcements

STB-1	2	an1	STB overview *J. Hilbe*
STB-1	3	an2	The editorial board *J. Hilbe*
STB-1	4	an3	STB and STB disk subscriptions
STB-1	4	an4	Submission guidelines
STB-1	5	an5	BeNeLux countries Stata distributor *J. Groot*
STB-1	5	an6	CRC provides 800 service to Canada *C. Fox*
STB-1	5	an7	Physiatric research supplement available *T. Findley*
STB-1	5	an8	Stata-based instructional software for social science *M. Macy*
STB-3	2	an9	Change in associate editors *J. Hilbe*
STB-3	2	an10	Stata available for DECstation *T. Anderson*
STB-3	2	an11	Stata X-Window driver available for SPARCstation *T. Anderson*
STB-4	2	an12	CRC has new area code *C. Fox*
STB-4	2	an13	TCL now marketing Stata in the UK *A. Timberlake*
STB-4	3	an14	SAS's new technical journal *J. Hilbe*
STB-4	3	an15	Regression with Graphics released *J. Hilbe*
STB-5	2	an15.1	Regression with Graphics now available from CRC *L. Brown*
STB-6	2	an16	Stata 3.0 released *T. Anderson*
STB-6	2	an16.1	Implications for the STB *J. Hilbe*

[crc] CRC-Provided Support Materials

STB-1	6	crc1	CRC provided support materials
STB-1	6	crc2	File comparison command
STB-1	7	crc3	Variable comparison command
STB-1	7	crc4	Data set rectangularization command
STB-1	7	crc5	Data set transposition command
STB-1	8	crc6	Likelihood-ratio test command
STB-2	2	crc7	Corrections and changes in ado-files
STB-2	2	crc8	Actuarial or life-table analysis of time-to-event data
STB-2	4	crc9	Replacing coded missing values
STB-3	3	crc10	Correction and updates to roc and poisson commands
STB-4	3	crc11	Drawing random samples
STB-6	3	crc12	Oops!

[dm] Data Management

STB-2	4	dm1	Date calculator *M. Ureta*
STB-3	3	dm2	Data format conversion using DBMS/COPY & STAT/TRANSFER *J. Hilbe*
STB-3	7	dm2.1	Vendor's response to review *S. Dubnoff*
STB-6	3	dm2.2	Stat/Transfer 2.0 review update *J. Hilbe*
STB-4	4	dm3	Automatic command logging for Stata *D. Judson*
STB-5	2	dm3.1	Typesetting correction to automatic command logging for Stata
STB-4	5	dm4	A duplicate value identification program *M. Jacobs*
STB-5	3	dm5	Creating a grouping variable for data sets *M. Jacobs*
STB-5	3	dm6	A utility to document beginning and ending variable dates *S. Becketti*

[gr] Graphics

STB-1	8	gr1	Enhancing visual display using stem-and-leaf *P. Geiger*
STB-2	5	gr2	Importing Stata graphs into MS-Word or WordPerfect *R. DeLeon and J. Anagnoson*
STB-2	6	gr3	Crude 3-dimensional graphics *W. Gould*
STB-2	8	gr4	3-dimensional contour plots using Stata and Stage *G. van de Kuilen*
STB-2	9	gr5	Triangle graphic for soil texture *F. Danuso*
STB-3	7	gr6	Lowess smoothing *P. Royston*
STB-3	9	gr7	Using Stata graphs in the Windows 3.0 environment *J. Hilbe*
STB-4	5	gr8	Printing a series of Stata graphs *J. Anderson*
STB-5	5	gr9	Partial residual graphs for linear regression *J. Hilbe*
STB-5	6	gr10	Printing graphs and creating WordPerfect graph files *T. Saving and J. Montgomery*

[in] Instruction

STB-1	9	in1	Installation of the STB disk	

[ip] Instruction on Programming

STB-5	4	ip1	Customizing a Stata menu system	*M. Jacobs*

[os] Operating System, etc.

STB-1	10	os1	Gphpen and colour PostScript	*R. Reese*
STB-3	10	os1.1	Update on gphpen and colour PostScript use	*R. Reese*
STB-2	10	os2	Questions and answers about Stat/Transfer	*S. Dubnoff*
STB-4	6	os2.1	os2—an addendum	*T. Anagnoson*
STB-3	11	os3	Using Intercooled Stata with DOS 5.0	*J. Hilbe*
STB-4	7	os3.1	Comment: Using Intercooled Stata with DOS 5.0	*M. Jacobs*
STB-4	7	os3.2	A follow-up question to os3	*A. Katz*

[qs] Queries and Suggestions

STB-1	10	qs1	Request for Stata course outline	*P. Geiger*
STB-2	11	qs2	Why is the cubic spline so-called?	*T. Anagnoson*
STB-2	11	qs3	Request for confidence intervals for proportions	*I. Zavala*
STB-3	12	qs4	Request for additional smoothers	*I. Ugarte*

[sbe] Biostatistics and Epidemiology

STB-1	11	sbe1	Poisson regression with rates	*W. Rogers*
STB-2	11	sbe2	Bailey–Makeham survival model	*W. Rogers*
STB-3	12	sbe3	Biomedical analysis: Radioimmunoassay calculations	*P. Geiger*
STB-5	7	sbe4	Further aspects of RIA analysis	*P. Geiger*

[sed] Exploratory Data Analysis

STB-1	13	sed1	Stata and the four R's of EDA	*R. DeLeon and T. Anagnoson*
STB-2	14	sed2	Ladder-of-powers variable transformation	*W. Gould and J. Hilbe*
STB-2	15	sed3	Variable transformation and evaluation	*T. Findley*
STB-3	15	sed4	Resistant normality check and outlier identification	*L. Hamilton*
STB-3	18	sed5	Enhancement of the Stata collapse command	*P. Banens*
STB-6	4	sed6	Quartiles, outliers, and normality: Some Monte Carlo results	*L. Hamilton*

[sg] General Statistics

STB-1	17	sg1	Nonlinear regression command	*F. Danuso*
STB-3	19	sg1.1	Correction to the nonlinear regression program	*J. Hilbe*
STB-1	19	sg2	Exact and cumulative Poisson probabilities	*J. Hilbe*
STB-1	20	sg3	Skewness and kurtosis test of normality	*W. Gould*
STB-2	16	sg3.1	Tests for departure from normality	*P. Royston*
STB-3	19	sg3.2	Shapiro–Wilk and Shapiro–Francia tests	*P. Royston*
STB-3	20	sg3.3	Comment on sg3.1	*R. D'Agostino, A. Belanger, and R. D'Agostino Jr.*
STB-3	20	sg3.4	Summary of tests of normality	*W. Gould and W. Rogers*
STB-3	23	sg3.5	Comment on sg3.4 and an improved D'Agostino test	*P. Royston*
STB-4	8	sg3.6	A response to sg3.3: Comment on tests of normality	*P. Royston*
STB-5	10	sg3.7	Final summary of tests of normality	*W. Gould*
STB-3	25	sg4	Confidence intervals for t-test	*R. Goldstein*
STB-5	11	sg5	Correlation coefficients with significance levels	*S. Becketti*
STB-5	12	sg6	Regression switching models	*D. Benjamin and W. Gould*

[smv] Multivariate Analysis

STB-4	9	smv1	Single-factor repeated measures ANOVA	*J. Hilbe*
STB-5	13	smv1.1	Minor change to single-factor repeated measures ANOVA	*J. Hilbe*
STB-4	10	smv2	Analyzing repeated measurements—some practical alternatives	*W. Rogers*
STB-5	13	smv3	Regression-based dichotomous discriminant analysis	*J. Hilbe*
STB-6	5	smv4	One-way multivariate analysis of variance (MANOVA)	*J. Hilbe*
STB-6	7	smv5	Performing loglinear analysis of cross-classifications	*D. Judson*

[snp] Nonparametric methods

STB-2	17	snp1	Kolmogorov–Smirnov one- and two-variable tests *L. Giventer*
STB-3	26	snp2	Friedman’s ANOVA test and Kendall’s coefficient of concordance *R. Goldstein*
STB-3	28	snp3	Phi coefficient (fourfold correlation) *R. Goldstein*

[sqv] Analysis of Qualitative Variables

STB-1	21	sqv1	Additional logistic regression extensions *J. Hilbe*
STB-2	21	sqv1.1	Correction to logit regression extensions *J. Hilbe*
STB-3	28	sqv1.2	Additional logistic regression diagnostic: Cook’s distance *J. Hilbe*
STB-4	16	sqv1.3	An enhanced Stata logistic regression program *J. Hilbe*
STB-5	17	sqv1.4	Typographical correction to enhanced logistic regression *J. Hilbe*
STB-6	17	sqv2	A graphical method for assessing goodness-of-fit of logit models *D. Judson*

[srd] Robust Methods and Statistical Diagnostics

STB-2	21	srd1	How robust is robust regression? *L. Hamilton*
STB-2	26	srd2	Test for multivariate normality *R. Goldstein*
STB-2	26	srd3	One-step Welsch bounded-influence estimator *R. Goldstein*
STB-2	27	srd4	Test for general specification error in linear regression *R. Goldstein*
STB-2	27	srd5	Ramsey test for heteroscedasticity and omitted variables *R. Goldstein*
STB-2	27	srd6	A randomization test for the equality of two groups *R. Goldstein*
STB-5	17	srd7	Adjusted summary statistics for logarithmic regressions *R. Goldstein*
STB-5	21	srd8	Interpretations of dummies in regressions with log dependent variable *R. Goldstein*
STB-5	22	srd9	Box–Cox statistics for help in choosing transformations *R. Goldstein*
STB-5	25	srd10	Maximum-likelihood estimation for Box–Cox transformation *P. Royston*
STB-6	19	srd11	Generating ordered (“cascading”) dummy variables *R. Goldstein*
STB-6	22	srd12	Some model selection statistics *R. Goldstein*

[ssa] Survival Analysis

STB-1	23	ssa1	Actuarial or life-table analysis of time-to-event data *H. Krakauer and J. Stewart*
STB-2	28	ssa1.1	Menu interface to life-table command *W. Gould*
STB-5	26	ssa2	Tabulating survival statistics *W. van Putten*

[ssi] Simulation and Random Numbers

STB-1	25	ssi1	Monte Carlo simulation *L. Hamilton*
STB-4	18	ssi2	Bootstrap programming *L. Hamilton*

[sts] Time Series and Econometrics

STB-5	27	sts1	Autocorrelation and partial autocorrelation graphs *S. Becketti*

[tt] Teaching

STB-4	27	tt1	Teaching beginning students with Stata *T. Anagnoson and R. DeLeon*
STB-4	28	tt2	Using “front ends” for Stata *M. Macy*
STB-6	26	tt3	Teaching biochemistry and chemistry: Understanding buffer solutions *P. Geiger*