STATA TECHNICAL BULLETIN

May 1991
STB-1

A publication to promote communication among Stata users

Editor

Joseph Hilbe
Stata Technical Bulletin
10952 North 128th Place
Scottsdale, Arizona 85259-4464
602-860-1446 FAX
stb@stata.com EMAIL

Associate Editors

J. Theodore Anagnoson, Cal. State Univ., LA
Richard DeLeon, San Francisco State Univ.
Paul Geiger, USC School of Medicine
Richard Goldstein, Qualitas, Inc.
Stewart West, Baylor College of Medicine

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

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

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

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

Contents of this issue

an1	STB overview

The *Stata Technical Bulletin* is a publication for and by Stata users. It is a forum for users of all disciplines and levels of sophistication to share ado-files, methods of analysis, programming strategies, data sets, software, and instruction on the more esoteric aspects of Stata, as well as questions, suggestions, and ideas. In short, the STB is aimed at providing the Stata user with a comprehensive Stata resource center.

As editor, it will be my job, with the assistance of the Editorial Board, to provide the STB with overall direction and primary technical support. My day-to-day mission is to seek your involvement via submissions, questions, and comments. The Editorial Board will set and periodically review STB policy and progress. Our intention is to be as responsive as possible to the needs of the Stata user community. Brief biographies of myself and the Editorial Board can be found in *an2*, below.

The term '*an2*' refers to articles, to be called inserts, found in the STB. Each insert is coded by a prefix and number. '*An*' is the announcement category, and '*2*' indicates that it is the second announcement. The initial set of categories are

General Categories:

an	announcements	*ip*	instruction on programming
cc	communications & letters	*os*	operating system, hardware, & interprogram communication
dm	data management	*qs*	questions and suggestions
dt	data sets	*tt*	teaching
gr	graphics	*zz*	not elsewhere classified
in	instruction		

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. See *crc1*, below.

Associated with every issue of the STB will be an STB disk. The STB disk will contain the programs and data sets described in the STB and will also contain material mentioned, but not fully published, in the Bulletin. This will include help files, bibliographies, and longer statistical output.

Material on the STB disk is indexed to correspond to the coding explained above. For example, the first insert relating to a biostatistical topic will be the printed insert *sbe1*. Any related software would be found in the `\sbe1` (`/sbe1`, Unix) directory on the disk. If a user or the original author expands or comments on *sbe1* in a later issue, this would be printed as *sbe1.1* and software, if any, would appear in the `\sbe1.1` directory on the disk. One can thus follow the development of a particular program or issue from its initial insert to current time by accessing the *sbe1.** inserts and directories. A complete description of the STB disk, along with installation instructions, can be found in *in1*, below. Ordering of the STB disks is covered in *an3*.

Obviously, if the STB is to be a success, we will need your submissions. Submissions are expected to range from the very detailed to the rather brief and straightforward. Among possible submissions are

- Ado-files that provide the user with routines or capabilities not presently in Stata.
- Enhancements, improvements, or adaptations of current ado-files, either CRC-supplied or previously published in the STB.
- Programs and macros written in Stata that improve on or expand the capabilities of the programming, graphical, printing, or user interface environment.
- Programs written in other languages that are designed to be used with Stata.
- Technical discussions, both short and lengthy.
- Data sets.
- Communications and letters regarding all manner of Stata related material, including questions, comments, confirmations, suggestions, and desires.
- Announcements regarding interesting related publications, achievements, future inserts, and so forth.

All types of submissions are welcome. Statements such as "Results are the same as those produced by BMDP" or questions such as "Why are these results different than those produced by SPSS?" are actively sought. In addition, the announcement category is extremely broad. If you have a publication that you feel would be of interest to the Stata community, or know of a relevant meeting or new product, please announce it.

For those of you who desire assistance in the preparation of your submissions, it can usually be obtained by writing or faxing the editor at the STB office. However, it must be emphasized that it is not necessary to submit elaborately professional programs; many less-sophisticated ones actually serve as more appropriate learning tools than do ones that few can interpret. Programs, comments, questions, etc., which you feel are important only to you are likely to be important to many others as well.

Instructions for preparing submissions can be found in *an4*.

You, the user, are the STB's greatest beneficiary—but at the same time, the success of the enterprise is dependent on your active participation. Stata, of all statistical packages, seems to lend itself most readily to this manner of user interaction. I and the Editorial Board invite you to become part of this innovative way of "doing" statistics.

—Joseph Hilbe, Editor.

an2 The editorial board

Dr. Joseph Hilbe, Editor, is currently a statistical consultant living in Scottsdale, Arizona. He was a faculty member in the University of Hawaii system for sixteen years prior to moving to Seattle, Washington in 1986. During this time Dr. Hilbe wrote five textbooks, authored many articles, spoke at academic conferences both in the United States and Europe, and served as chairman for several national committees. His foremost teaching interests during the last ten years were in quantitative reasoning and statistics. While in Seattle he was Director of Biostatistics at Northwest Hospital and statistical consultant to the Professional Review Organization–Washington. He later assumed the CEO position at Accusoft, Inc. where he was involved in both statistical programming and expert system design. Since moving to Arizona in 1990, Dr. Hilbe has been Senior Biostatistician at Health Services Advisory Group in Phoenix, wrote a discriminant analysis module for another statistical package, and is currently a statistical subcontractor for the Health Care Financing Administration analyzing the Medicare Infrastructure review system.

Dr. Ted Anagnoson, Associate Editor, is Professor and Chair of the Political Science Department at California State University, Los Angeles. He teaches courses in computers in the public sector, statistics and data analysis, and microcomputer management techniques. His academic publications include some twenty articles, book chapters, and monographs. He has received three NSF grants for his research, equipment, and for two workshops for faculty on exploratory data analysis and Stata. Dr. Anagnoson's primary areas of statistical specialization are exploratory data analysis, structure analysis, econometrics, and statistical graphing.

Dr. Richard Deleon, Associate Editor, is Professor of Political Science and Urban Studies at San Francisco State University. He is also Director of San Francisco State's Public Research Institute and Survey Research Center. He teaches courses in public administration, urban research methods, statistics, and data analysis. He has published fifteen articles, book chapters, and monographs in political science and is completing a book on San Francisco politics. He also authored more than twenty research reports on Bay Area Urban Problems and Policy Issues. He has conducted numerous workshops on Stata including a state-funded workshop for faculty on Stata as a tool for exploratory data analysis. His foremost areas of statistical expertise are exploratory data analysis, regression, multivariate analysis and survey methods.

Dr. Paul Geiger, Associate Editor, is Associate Professor in the Department of Pharmacology and Nutrition at the USC School of Medicine. His primary interests are in methods development in the chromatography of metabolic phosphate compounds, mechanisms of insulin action, and enzymology. Dr. Geiger has published widely in biochemistry and authored a text in chromatography. His statistical expertise rests in the area of clinical trials, EDA, categorical data analysis, and robust regression techniques.

Dr. Richard Goldstein is a self-employed statistical consultant actively involved in evaluating statistical software, including for the Boston Computer Society and in papers published in the *American Statistician* and the *Journal of Applied Econometrics*. He is currently editor of the *Software Computing Reviews* section of the *American Statistician*. His areas of statistical specialization include statistical computing, econometrics, linear models, and areas within the domain of biostatistics and epidemiology.

Dr. Stewart West is Assistant Professor of Medicine at Baylor College of Medicine and also Adjunct Assistant Professor of Biometry at the University of Texas School of Public Health. He is currently involved both as a researcher and investigator on a number of NIH grants in cardiovascular and other research areas. Dr. West has been a beta tester of a variety of packages including Stata, where he is a recognized expert in areas related to biostatistics and epidemiology as well as multivariate analysis.

You may request that the editors review your submission and, if necessary, the editors will even go to persons outside the Editorial Board. In general, however, the editors will simply verify that the program performs as claimed. The editors may also confirm, comment on, or compare inserts with other STB inserts, software packages, or articles from other publications. Our aim is to provide both you and subscribers with a solid system of support.

an3 STB and STB disk subscriptions

The STB and associated disk will be published every other month—six issues per year. Current Stata users will receive the first two issues of the STB without charge but also without disks. Rates are as follows:

1. A one-year subscription to the STB alone is $30 in North America and $45 elsewhere.
2. Each STB disk purchased separately is $30 in North America and $45 elsewhere, including shipping.
3. A one-year subscription to both the Bulletin and disks (shipped together) is $100 in North America and $135 elsewhere. Prices include shipping. This combined rate represents a $110 savings in America and a $180 savings elsewhere. Sites may order multiple subscriptions to the STB and only one subscription with diskettes and then freely copy the diskettes.

STB disks will be available on 5.25 and 3.5-inch diskettes, both DOS and Unix formats, and, for Unix, on cartridge and 1600-bpi tape. For tapes, there is an additional charge of $25 in North America and $35 elsewhere to cover the cost of media and shipping. Networked Unix users with PCs running PC/NFS are reminded that they can order the DOS version since files are identical for both DOS and Unix.

Orders are placed with Computing Resource Center, 1640 Fifth Street, Santa Monica, California 90401. Orders may be placed by fax using 213-393-7551 or by telephone using 800-782-8272 or 213-393-9893.

an4 Submission guidelines

Submissions to the STB should be sent to the Editor at the address that appears on the first page. Short inserts with no accompanying STB diskette materials may be mailed or faxed. Longer inserts, or inserts with accompanying STB diskette materials, should be sent on either 5.25- or 3.5-inch DOS-formatted diskettes.

The root directory of the diskette should contain the text of the insert in standard ASCII format in a file named `INSERT`. Note that the text must not be in the format of some word-processor such as WordPerfect. It is not important that the lines be of equal length or that the submission look pretty when printed. The text-processing software used by the STB will handle that. You are warned that inserts will be only minimally edited, to correct only the grossest grammar violations and spelling errors. Please include a printed copy of the `INSERT` file as well as the diskette. If the insertion includes mathematical formulas, indicate them the best you can and they will be converted to the appropriate format. User's familiar with TEX are asked to use TEX format.

Materials for inclusion on the STB diskette should appear in the `\STB` directory on the diskette. Included in `\STB` must be a file named `README`. The `README` file should include the installation instructions and, if you wish, may include your name and affiliation. If the standard *in1* installation instructions are appropriate, the `README` file should simply say that installation instructions can be found in *in1*. The `\STB` directory may contain files and subdirectories in whatever layout you wish. The contents of the `\STB` directory will be moved, in its entirety, to the STB diskette. The directory will simply be renamed to the category code–sequence number assigned to your insert.

an5 BeNeLux countries Stata distributor

Jaap Groot, Oasis Decision Support Systems, The Netherlands

European Stata users in the BeNeLux countries may wish to contact

Oasis Decision Support Systems
Lekstraat 4
3433 ZB Nieuwegein
Nederland
tel. 03402-66336
fax 03402-65844
Contact Persons: Frank Clement, Gerard v.d. Kuilen, or Jaap Groot

In addition to being the Stata distributor in the BeNeLux countries, Oasis provides introductory and advanced training on Stata and provides a technical hotline. Users who register with Oasis will be sent a local newsletter, in Dutch, containing both technical and product information on Stata.

an6 CRC provides 800 service to Canada

Charles Fox, CRC, 800-782-8272 USA, 800-248-8272 Canada

CRC now provides an 800 telephone number for Canadians. Our existing 800 number, 800-STATAPC, could not be dialed from Canada, so we added another 800 number: 800-248-8272. Sharp-eyed readers will notice that only the prefix is changed; that's because no prefix ending in a 2 is available for anyone other than the Canadian government. We cannot give you a nice STATAPC number to dial from Canada, but our Marketing Director, Ted Anderson of Vancouver, assures us that BHUUBRB holds great meaning for all Canadian statisticians.

an7 Physiatric research supplement available

Thomas Findley, MD, PhD, Kessler Institute for Rehabilitation

The *American Journal of Physical Medicine & Rehabilitation* has published a 171-page supplement entitled "Physiatric Research: A Hands-On Approach." The Supplement is designed to provide practical advice to both beginning and advanced researchers in medical rehabilitation.

Stata is used throughout the twelve articles I authored or co-authored. Three articles that may be of particular interest to Stata users are *Data Entry and Early Exploratory Data Analysis*; *Preliminary Data Analysis*; and *Primary Data Analysis*. The other articles are entitled: *How to Ask the Question*; *The Conceptual Review of the Literature or How to Read More Articles Than You Ever Want to See in Your Entire Life*; *The Chart Review or How to Use Clinical Data for Exploratory Retrospective Studies*; *Some Practical Designs in Applied Research*; *Research Project Management*; *The Role of the Principal Investigator*; *Information Resources*; *Research Training: Setting the Stage for Lifelong Learning*; and *Measurement Tools with Application to Brain Injury*.

Individual reprints from the series are available on request from my office. Stata users are asked to send $1 for each article requested to cover postage and handling. Please write to Thomas Findley, MD, PhD, Director of Research, Kessler Institute for Rehabilitation, 1199 Pleasant Valley Way, West Orange, NJ 07052.

A copy of the entire Supplement may be obtained for a $15 contribution to the Association of Academic Physiatrists. Please write: Carolyn Braddom, Association of Academic Physiatrists, American Journal of Physical Medicine and Rehabilitation, 290 Norristown Road, P.O. Box 977, Blue Bell, PA 19422.

an8 Stata-based instructional software for the social science classroom

Michael Macy, Midas Instructional Software, Jamaica Plain, MA, 617-736-2646

The Microcomputer Integrated Data Analysis System (MIDAS) is a highly-intuitive, menu-driven "front-end" to Stata designed to broaden undergraduate instruction in quantitative reasoning and empirical research. MIDAS is predicated on the belief that the social sciences can be learned through laboratory exercises in the same way as the natural and behavioral sciences. A quantitative research component in the undergraduate social-science curriculum serves two vital functions for students:

1. Deepening students' facility with course material through exploration of concrete problems. Learning improves with open-ended inquiry.
2. Developing an appreciation of the basic principles of causal logic, inferential reasoning, real-world application of theory, and empirical falsification.

On the other hand, students with little or no training in statistics, research methods, and computers simply cannot be turned loose with a dataset and a professional research package. Students would be faced with a bewildering array of choices leading to certain disaster. (How many times have students produced meaningless correlations between nominal variables or cross-tabulations of continuous measures?) To alleviate that problem, instructors are forced to develop lab exercises that walk the student through a demonstration of the intended result, with the result that there is no opportunity for the student to engage in real inquiry.

The MIDAS interface to Stata, on the other hand, automatically keeps track of data, variables, cases, skip patterns, measurement level, number of values, and causal order, and based on the student's prior choices, automatically precludes inappropriate options from the current menu. This allows the student to chart their own course without falling off the edge of the Earth. What is analyzed is also left up to the student. MIDAS includes a massive social-science data archive, including U.S. Census data and

seven years of the General Social Survey. MIDAS interprets menu choices—so there are no commands for the student to learn and forget—retrieves the necessary data from the archive, and writes a do-file that tells Stata how to analyze the data. MIDAS then reads and interprets Stata's output and presents the results (including significance tests) in nontechnical, "common sense" language. When appropriate, the results conclude with a "Technical Report" that exposes students to untranslated Stata output.

An elaborate help system and exhaustive error checking ensure that students get results, not error messages. Any time students wonder what to do, they can press the *Help Key* for context-sensitive advice. The help system even explains statistical concepts such as how control variables are used and gives examples that illustrate the logic. MIDAS provides its extensive explanations through its on-line statistical glossary as well as a detailed codebook on all variables.

MIDAS permits a full range of univariate, bivariate, and multivariate procedures, including Stata's high-resolution graphic displays. Students can focus their analyses on subpopulations and can also look at relations between variables while controlling for confounding factors. Several important limitations were built in. MIDAS will not recode or transform variables, nor will it perform N-way contingency tables, ANOVA, time-series analysis, and other advanced procedures. However, MIDAS will let ambitious students download variables from the archive and then analyze them directly using Stata.

Field testing has confirmed the power of MIDAS to capture the research imagination of even avowedly "math-phobic" liberal arts students. MIDAS provides a structured research process that is flexible, manageable, and responsive yet also remarkably bulletproof.

How to Order

MIDAS consists of two components, an Integrated Data Archive that is purchased and installed by the school, and a Program Disk, purchased by the students. The archive costs $300 for the first computer (or $350 for a 386 version using Intercooled Stata) and $50 for each additional installation in a lab. In order to minimize the unit cost for using MIDAS, the Program Disk is sold through college bookstores on a "subscription" basis for $15. Each disk permits 300 statistical procedures to be performed before the "subscription" expires, which is ample for a one-semester course with intensive use. The Program Disk can also be purchased bundled with *Social Association: An Inquiry Approach*, a workbook of innovative, open-ended sociology exercises that take advantage of the power of MIDAS to facilitate self-directed inquiry ($30).

MIDAS may be ordered from CRC or directly from Midas Instructional Software, 12 Enfield Street, Jamaica Plain, MA 02138, telephone 617-763-2646. Additional literature, with a demonstration diskette, is available without charge from either company.

crc1 CRC-Provided support materials

The materials that used to be described in the *Stata News* and provided on the Stata Support Disk are now moved to the STB. The materials provided in the `\crc` directory of the STB disk are the same, cumulative materials that were previously provided on the Stata Support Disk.

Inserts published in the *crc* category are included in the `\crc` directory and supported on our help line.

crc2 File comparison command

The syntax of `cf` is

`cf` *varlist* `using` *filename* [`, verbose`]

Have you ever had data in memory based on a file on disk, thought you might have changed it, but were not sure? `cf` compares *varlist* of the data in memory to the corresponding variables in *filename*.

For instance, the data in memory you think is identical to `mydata.dta`, but you are unsure. If they are different, you want to understand the change before continuing:

```
. cf _all using mydata
. -
```

Unless `verbose` is specified, only differences are mentioned—silence indicates matches. In this case, the files are identical (but see *Warning 1*, below). Alternatively, you might see

```
. cf _all using mydata
mpg:  2 mismatches
mpgsq:  does not exist in using
r(9);
```

Two changes were made to the `mpg` variable and you added the `mpgsq` variable.

Warning 1: If you type '`cf _all ...`', the variables in memory are compared to the file on disk. The file on disk *might* also contain additional variables, variables not in the file in memory. Thus, silence does not really mean that the files are identical since the variables in memory could be a subset of the variables on disk. Think of using a data set, dropping one of the variables, and then comparing. `cf` would respond with silence.

Warning 2: Changes in the sort order of the data confuse `cf` into thinking there are mismatches when there may not be. In this case, check the sort order of the data on disk and make sure the data in memory is sorted in the same way.

Warning 3: `cf` is an ado-file, which should not matter, but does. In particular, ado-files impose limits on the number of variables in a *varlist*. You can try `_all`, but if there are too many variables in the data, you may get the error "`too many variables`". In that case, you will have to perform the comparison in groups of variables, for instance, '`cf make-weight using mydata`' followed by '`cf mpg-displ using mydata`'.

crc3 Variable comparison command

The syntax of `compare` is

`compare` *varname1 varname2* [`if` *exp*] [`in` *range*]

`compare` performs an accounting of the differences and similarities between *varname1* and *varname2*:

```
. compare rep77 rep78
                                        ---------- difference ----------
                             count      minimum      average      maximum
-------------------------------------------------------------------------
rep77<rep78                     16           -3      -1.3125           -1
rep77=rep78                     43
rep77>rep78                      7            1            1            1
                           ----------
jointly defined                 66           -3    -.2121212            1
rep77 missing only               3
jointly missing                  5
                           ----------
total                           74
```

Most useful is the table at the end of jointly defined, singly defined, and jointly missing.

crc4 Data set rectangularization command

The syntax of `fillin` is

`fillin` *varlist*

The technical and uninterpretable description of `fillin` is as follows: `fillin` adds observations with missing data to your data so that all interactions of *varlist* exist, thus making a complete rectangularization of *varlist*. *Varlist* refers to the identifying variables. `fillin` also adds the variable `_fillin` to your data. `_fillin` is 1 for created observations and 0 for previously existing variables.

This sounds more complicated than it is. Assume you have data on something by sex-race-and-age group. You suspect that some of the combinations of sex-race-and-age do not exist, but if so, you want them to exist with whatever remaining variables there are in the data set to missing. That is, rather than having a missing observation for female blacks aged 20-24, you want an observation created containing missing values. You type

```
. fillin sex race age
```

You can find out if any observations were added by typing '`tabulate _fillin`'. You could undo `fillin` by typing '`drop if _fillin`'.

crc5 Data set transposition command

The syntax of `xpose` is

`xpose, clear` [`varname`]

`xpose` transposes the data, changing variables into observations and observations into variables. The `clear` "option" is not optional. This is supposed to remind you that the untransposed data will be lost (unless you have saved it previously).

The `varname` option adds the new variable `_varname` to the transposed data containing the original variable names. In addition, with or without the `varname` option, if the variable `_varname` exists in the data before transposition, those names will be used to name the variables after transposition. Thus, transposing the data twice will (almost) yield the original data.

All new variables—that is, after transposition—are made `float`. Thus, any original variables that were strings will result in observations containing missing values. (Transposing the data twice, therefore, will result in loss of the contents of string variables.)

crc6 | Likelihood-Ratio test command

The syntax of `lrtest` is

`lrtest` [`, saving(`*name*`) using(`*name*`) model(`*name*`) df(`*#*`)`]

`lrtest` performs a likelihood-ratio test between pairs of `cox`, `logit`, or `probit` models. `lrtest` is used both to save information on models and to perform tests of `model()` against `using()`. Name may be alphanumeric or strictly numeric, but may not exceed four characters. If `saving()` is not specified, the current model is not saved. If `using()` is not specified, `using(0)` is assumed. If `model()` is not specified, the most recently estimated model is assumed. `df()` need not be specified unless one wishes to override the automatic degree-of-freedom calculation.

To use this command, one first estimates the saturated model (say, '`logit foreign rep77 displ length weight`') and then types

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

This saves information on the model under the name `0`. The name `0` is special; it is the assumed name for subsequent `lrtest`s if `using()` is not specified. Thus, one can subsequently estimate alternative models (say, '`logit foreign rep77 displ`') and simply type

```
. lrtest
  chi2(2) = .694902  sig. = 0.7065
```

to perform the likelihood-ratio test of the current with the saturated model. One could then save this model under a different name, for instance,

```
. lrtest, saving(1)
```

and now estimate a different model. Typing '`lrtest`' would compare the now-current model to the fully saturated model. Typing '`lrtest, using(1)`' would compare the current model to the model previously estimated. Typing '`lrtest, model(1) using(0)`' would compare the previous model to the fully saturated model. In this case the `using(0)` is unnecessary.

As a better example:

```
. logit chd age age2 sex              /* estimate saturated model     */
. lrtest, saving(0)                   /* save results                 */
. logit chd age sex                   /* estimate simpler model       */
. lrtest                              /* obtain test                  */
. lrtest, saving(1)                   /* save new results as 1        */
. lrtest chd sex                      /* estimate simplest model      */
. lrtest, using(1)                    /* compare to model 1           */
. lrtest, model(1)                    /* repeat earlier test          */
```

gr1 | Enhancing visual display using stem and leaf

Paul Geiger, USC

Preliminary examination of a data set is necessary to establish some idea of the distribution of each of the variables. One method used for moderate size sets, perhaps between 20 and 300 values, is the stem-and-leaf display. The familiar histogram is another. The stem-and-leaf plot has the advantage of displaying actual numbers. Both enable one to see (1) how symmetric the batch is; (2) how spread out the numbers are; (3) whether there are outliers; (4) whether there are concentrations of data; (5) whether there are gaps in the data.

In organizing a stem-and-leaf plot, the number of lines from top to bottom must be decided upon in order to give an effective visual display. Emerson and Hoaglin (1983) treat the subject at length and recommend, $L = 10 \cdot \log_{10} n$ which is easily computed with Stata's command

```
. display 10*log(n)/log(10)
```

where n is the number of data values for the variable under scrutiny. They also discuss other rules that have been proposed for deciding on the upper limit of L, and $L = \sqrt{n}$ is probably most satisfactory for sets with $n < 50$. This formula is due to Velleman (1976). However, the $10\log_{10} n$ equation is probably the best on balance and was actually found in a paper by Dixon and Kronwal (1965) who used it for histograms.

Stata makes both stem-and-leaf plots and histograms. As of this writing the histograms are more malleable and the graphics are excellent. One of the graphical options for histograms is the choice `bin(#)`, where # can be up to 50. Rather than guessing and making a few trials, the `bin(#)` is readily computed by the previous expressions.

One can be convinced quickly of the visual quality of the histograms thus produced by using data sets of one's own or by generating a data set with the command '`gen newvar = invnorm(uniform())`', after first setting the number of observations by using '`set obs #`', say for values between 20 and 300. The results are graphed and inspected. Practice is enhanced by making bimodal or skewed distributions using the `generate` and `stack` commands. It is useful to examine these new variables with both stem-and-leaf and histograms. It is also instructive to make gaps and outliers in the synthetic data sets and reexamine them with stem-and-leaf and histograms.

References

Dixon, W. J. and R. A. Kronwal. 1965. *Journal of the Association for Computing Machinery* 12: 259–261.

Emerson, J. D. and D. C. Hoaglin. 1983. In *Understanding Robust and Exploratory Data Analysis*, edited by D. C. Hoaglin, F. Mosteller and J. W. Tukey. New York: John Wiley & Sons, chapter 1.

Velleman, P. F. 1976. 1975. *Proceedings of the Statistical Computing Section.* Washington, DC: American Statistical Association.

in1	Installation of the STB disk

The STB "disk" contains directories named with the category code–sequence number of the corresponding insert printed in the STB. For instance, materials corresponding to the insert *sqv1* appear in the `\sqv1` (`/sqv1`, Unix) directory. The materials corresponding to *sqv1.1* appear in `\sqv1.1`.

Unix users working from tape or `cpio`(1)-format diskette should copy the tape to a temporary directory. Printed instructions are provided with the media. DOS users can simply insert the diskette into their diskette drive and examine it directly. Here are detailed instructions for the DOS user:

Put the diskette in drive A:. These instructions are also on the diskette, so you can type

```
> TYPE README
```

to review them. If you wish to install the standard CRC support materials (what used to be provided on the CRC Support Disk and what is now documented in the *crc* inserts), type

```
> COPY A:\CRC\*.* C:\STATA\ADO
```

If you also wish to install the materials for, say, *sqv1*, type

```
> TYPE A:\SQV1\README
```

The `\SQV1\README` file will instruct you on how to install the materials for *sqv1*. In many cases, the `README` file will simply say to follow the standard instructions. For instance:

```
> TYPE A:\SQV1\README
SUBJECT:  extlogit:  extensions to the logit command
 AUTHOR:  Joseph Hilbe, Editor, STB
SUPPORT:  Fax 602-860-1446 or write STB; no telephone calls
COMMENTS:
Use standard installation instructions.
From inside Stata, on-line help is available.  Type ´help extlogit´
```

The standard instructions are to copy the contents of the directory to `C:\ADO`, *not* `C:\STATA\ADO`. If you do not have such a directory, first create it:

```
> MD C:\ADO
```

Once the directory exists, you can copy the materials:

```
> COPY A:\SQV1\*.* C:\ADO
```

os1 | Gphpen and colour Postscript

R. Allan Reese, University of Hull, UK. Fax (011)-44-482-466441

The file `ps.plf` is the header file used by `gphpen` for PostScript devices. One needs simply to modify this file to add colour support—only `/Gp` needs new code. The following simple-minded code works although it is clearly inefficient:

```
%!PS-Adobe-2.0 ESPF-1.2
%%BoundingBox:  0 0 396 504
%%Creator:  STATA/GPHPEN 2.0
%% Modified by R. A. Reese, Hull Univ, UK, for Colour PS.
%%EndComments
% The following segment of code defines the STATA/PostScript environment.
% Begin of STATA/PostScript direct interface calls:
% Gp:  Define the current pen number.
/Gp {                                          % pnum Gp -
        /cpno exch def                            %
        .5 1 1 sethsbcolor                        % default pen
        cpno 0 eq {0 0 1 sethsbcolor} if          % pen 0 invisible
        cpno 1 eq {.5 1 .75 sethsbcolor} if       % pen 1 dim cyan
        cpno 2 eq {.166 1 1 sethsbcolor} if       % pen 2 yellow
        cpno 3 eq {0 0 0 sethsbcolor} if          % pen 3 black
        cpno 4 eq {0 1 1 sethsbcolor} if          % pen 4 red
        cpno 5 eq {.33 1 1 sethsbcolor} if        % pen 5 green
        cpno 6 eq {.8 1 1 sethsbcolor} if         % pen 6 purple
        cpno 7 eq {.5 .8 .1 sethsbcolor} if       % pen 7 cyan
        cpno 8 eq {.667 1 1 sethsbcolor} if       % pen 8 dark blue
        cpno 9 eq {0 0 .5 sethsbcolor} if         % pen 9 dim white
        } def
```

The rest of the `ps.plf` file is as before. In my experience, the above changes result in a QMS ColorPS printer outputting graphs that are a close match for the screen.

However, I then wasted some time trying to get version 2 (printed below) to work; it still does not (no output at all) but I cannot see the problem:

```
% Begin of STATA/PostScript direct interface calls:
% Define array Gpens length Npens to hold attributes and save values.
/Npens 10 def
/Gpens Npens array def
Gpens  0 [0 0 1]         put     % pen 0 invisible
Gpens  1 [.5 1 .75]      put     % pen 1 dim cyan
Gpens  2 [.166 1 1]      put     % pen 2 yellow
Gpens  3 [0 0 0]         put     % pen 3 black
Gpens  4 [0 1 1]         put     % pen 4 red
Gpens  5 [.33 1 1]       put     % pen 5 green
Gpens  6 [.8 1 1]        put     % pen 6 purple
Gpens  7 [.5 .8 1]       put     % pen 7 cyan
Gpens  8 [.667 1 1]      put     % pen 8 dark blue
Gpens  9 [0 0 .5]        put     % pen 9 dim white
% Gp:   Define current pen number.
/Gp {                                  % pnum Gp -
        /cpno exch def
        .5 1 1 sethsbcolor      % default pen
        Gpens cpno get aload pop sethsbcolor
        } def
```

Perhaps someone can explain what is wrong? I am also looking into putting greyscale options on the Mono PS driver; it would look nice on a graph to have the axes, etc., fainter.

qs1 | Request for Stata course outline

I would greatly appreciate receiving a course outline, perhaps containing hours and subject breakdown, for a 2 or 3 unit (semester course) in introductory data analysis using Stata. If someone among the users of Stata is willing to share such information, it would be an important saving to us in preparation time. Please address Dr. Paul Geiger, Ph.D., Associate Professor, Department of Pharmacology and Nutrition, USC School of Medicine, 2025 Zonal Ave., Los Angeles, CA 90033. I can also be reached by email: pgeiger@uscvm (bitnet) or pgeiger@vm.usc.edu (internet).

sbe1	Poisson regression with rates

William Rogers, CRC, FAX 213-393-7551

Several users have written to us about using the `poisson` regression command with rate rather than count variables. Rates, for instance, arise if some cases have more exposure than other cases. In the spirit of the example in the November 1990 *Stata News*, we might attempt to measure insect counts on differently-sized plots of land. We would expect higher counts on bigger plots.

Following the comments of one particular user, the `poisson.ado` file was modified so that it no longer required that the dependent variable be non-negative integers—that is, so that it would at least allow a rate rather than a count variable to be specified. However, the user should be warned that if the dependent variable is *not* a count, the inference statistics may be nonsensical. Consider the example originally presented in the November 1990 *Stata News* using data from Beale, "The transformation of data for entomological field experiments", *Biometrika* 32:243-262. The data contain insect counts on different areas of land, with dummy variables specifying which of four insecticides were used and on which of six plots they were applied.

In the *Stata News*, we presented the analysis in terms of counts:

```
. poisson insects dtreat* darea*
Iteration 0: Log Likelihood =  -2481.2263
Iteration 1: Log Likelihood =  -1155.1853
Iteration 2: Log Likelihood =  -545.83963
Iteration 3: Log Likelihood =  -508.94509
Iteration 4: Log Likelihood =  -508.42752
Poisson Regression (Log link function)          Number of obs =      24
Goodness of fit chi2(15) =   862.379            Model chi2(8) =3945.598
Prob > chi2              =    0.0000            Prob > chi2   =  0.0000
Log Likelihood           =  -508.428
Variable |  Coefficient    Std. Error       t    Prob > |t|        Mean
---------+------------------------------------------------------------
 insects |                                                      221.625
---------+------------------------------------------------------------
 dtreat2 |     -1.49265       .042613   -35.028     0.000           .25
 dtreat3 |     -.748955       .032218   -23.246     0.000           .25
 dtreat4 |     -2.60921       .069726   -37.421     0.000           .25
  darea2 |      .113597       .053369     2.129     0.043     .16666667
  darea3 |      .451027       .049634     9.087     0.000     .16666667
  darea4 |      .748717       .047082    15.902     0.000     .16666667
  darea5 |      .366104       .050483     7.252     0.000     .16666667
  darea6 |     -.285179       .059216    -4.816     0.000     .16666667
   _cons |      5.92811       .040621   145.939     0.000             1
---------+------------------------------------------------------------
```

Now suppose we analyze these same data as rates, presuming they come from 100 square-foot plots. Note that 100 is a meaningless constant in the sense that "square-foot" is a concept that appears nowhere else in the analysis. Nothing would be different if we said these were 11.11 square-yard plots, but the numbers would change. That violates a fundamental principle of statistical methods which says that the results should be essentially unchanged, regardless of the unit of measure chosen. Let's try:

```
. gen rinsect = insects/100
. poisson rinsect dtreat* darea*
Note:  You are responsible for interpretation of non-count dependent variable
Iteration 0: Log Likelihood =  -50.534727
Iteration 1: Log Likelihood =  -37.274336
Iteration 2: Log Likelihood =  -31.180879
Iteration 3: Log Likelihood =  -30.811934
Iteration 4: Log Likelihood =  -30.806747
Iteration 5: Log Likelihood =  -30.806743
Poisson Regression (Log link function)          Number of obs =      24
Goodness of fit chi2(15) =     8.624            Model chi2(8) =  39.456
Prob > chi2              =    0.9990            Prob > chi2   =  0.0000
Log Likelihood           =   -30.807
Variable |  Coefficient    Std. Error       t    Prob > |t|        Mean
---------+------------------------------------------------------------
 rinsect |                                                     2.21625
---------+------------------------------------------------------------
 dtreat2 |     -1.49265       .425968    -3.504     0.002           .25
 dtreat3 |     -.748955        .32206    -2.326     0.028           .25
 dtreat4 |      -2.6092       .696986    -3.744     0.001           .25
  darea2 |      .113597       .533483     0.213     0.833     .16666667
  darea3 |      .451028       .496147     0.909     0.372     .16666667
```

```
  darea4 |      .748717       .470638     1.591     0.124     .16666667
  darea5 |      .366104        .50463     0.725     0.475     .16666667
  darea6 |     -.285179       .591926    -0.482     0.634     .16666667
   _cons |      1.32294       .406049     3.258     0.003             1
---------+------------------------------------------------------------
```

Except for the constant term, the coefficients and means are identical, but the standard errors, t-values, chi-square tests, and significance levels have all changed dramatically! Do you see the relationships to dividing by 100? They are very mechanical. Both these and our previous results cannot be right. In fact, if we changed our measurements from feet to yards and re-estimated the model again, we would get another set of answers. These answers are wrong.

There is a right way to "control for" the size of the plot and, as you might expect, the only time the size of the plot makes a difference is when there is variation in the size of the plots. If all plots are the same size, the "size" adjustment effectively drops out of the equations (except for the constant term). The right way is to bring the plot size out of the rate and make it into a right-hand-side variable. Let rate = count/plotsize. If

$$\text{rate} = e^{\beta_0+\beta_1 X_1+\ldots}$$

then

$$\text{count} = e^{\log(\text{plotsize})+\beta_0+\beta_1 X_1+\ldots}$$

Thus, log(plotsize) is a right-hand-side variable with a coefficient fixed at 1. The problem is that the previously released `poisson` command provided no way to constrain a coefficient. Included on the STB disk is `poisson2.ado`, which does. The option is `offset(`*varname*`)`.

The example would now work as follows:

```
. gen lplotsiz = log(100)
. poisson2 insects dtreat* darea*, offset(lplotsiz)
Iteration 0: Log Likelihood =  -2481.2263
Iteration 1: Log Likelihood =  -1155.1853
Iteration 2: Log Likelihood =  -545.83963
Iteration 3: Log Likelihood =  -508.94509
Iteration 4: Log Likelihood =  -508.42752
Poisson Regression (Log link function)          Number of obs =      24
Goodness of fit chi2(15) =   862.379            Model chi2(8) =3945.598
Prob > chi2              =    0.0000            Prob > chi2   =  0.0000
Log Likelihood           =  -508.428
Variable |  Coefficient    Std. Error       t    Prob > |t|        Mean
---------+------------------------------------------------------------
 insects |                                                      221.625
---------+------------------------------------------------------------
 dtreat2 |     -1.49265       .042613   -35.028     0.000           .25
 dtreat3 |     -.748955       .032218   -23.246     0.000           .25
 dtreat4 |     -2.60921       .069726   -37.421     0.000           .25
  darea2 |      .113597       .053369     2.129     0.043     .16666667
  darea3 |      .451028       .049634     9.087     0.000     .16666667
  darea4 |      .748717       .047082    15.902     0.000     .16666667
  darea5 |      .366104       .050483     7.252     0.000     .16666667
  darea6 |     -.285179       .059216    -4.816     0.000     .16666667
   _cons |      1.32294       .040621    32.568     0.000             1
---------+------------------------------------------------------------
```

Note that the likelihood and inference statistics match the first, correct run using counts, but the coefficient of the constant matches the example with rates. (The `lplotsiz` variable did not have to be defined as a constant, but could have been a function of other available data on each case.)

One other facet of this example is noteworthy, namely the degree of significance assigned to the coefficients. Notice that the goodness-of-fit chi-square is very significant, which means that the individual cells are far from their Poisson-predicted values. In other words, there is an area × treatment interaction in these data. Area is a random effect, so we are probably not entitled to make strong statements about the overall effect of the treatments. The significance of the coefficients is based on the assumption that the values in the cells are Poisson.

One more comment: What do you do if you are analyzing published data that is presented as rates? Answer: You convert it back to a count. It is better to "guess" the population sizes than to analyze it as a rate. And then, at least, you will be aware that the significance levels are a function of your guess. Guess larger and you will get smaller standard errors. You were guessing anyway, you just didn't know it.

sed1	Stata and the four R's of EDA

Richard E. Deleon, San Francisco State Univ., and J. Theodore Anagnoson, Cal. State Univ., LA

Introduction

The four R's are resistance, residuals, re-expression, and revelation. They represent the four main emphases that distinguish exploratory data analysis (EDA) from confirmatory data analysis (CDA) and traditional statistical inference.[1] This article will briefly introduce EDA concepts, discuss the four R's, and illustrate Stata's power as an EDA tool.

The motivating spirit of EDA is (1) to investigate data directly making few assumptions rather than poking at the data indirectly through a veil of statistics and assumptions, and (2) to detect patterns *in* the data rather than to impose patterns *on* the data. The specialized vocabulary of modern EDA (median polish, stem-and-leaf plots, box and whiskers, midspreads, hinges, froots, etc.) obscures the fact that in concept and spirit EDA is quite old.

All good analysts look at their data in the spirit of EDA during the early stages of data analysis, prior to statistical model building and confirmatory tests of model reproducibility. One first checks the data, for example, to verify that variables are indeed normally distributed and linearly related before using methods like regression that assume normality and linearity. The well-known "Anscombe quartet" (see *Figure 1* at the end of this insert) illustrates the importance of inspecting the data before fitting a model to it. As the reader can check by running `regress y x` on each of the Anscombe data sets (`ans1.dta`, `ans2.dta`, `ans3.dta`, and `ans4.dta` on the STB disk), the estimates for the F value, R^2, the size of the coefficient and its standard error are nearly identical in every detail for all four plots shown in Figure 1. Yet a glance at the four plots shows clearly that the assumptions of the linear regression model are satisfied only in data set #1.

Resistance

In EDA terms, data = smooth + rough, i.e., data = model fit + residuals. The idea is to remove the smooth from the rough leaving it truly structureless, much in the same fashion as an econometrician plots a best fit to a time series. The result is a set of residuals, or the "rough" in EDA parlance, which is structureless. A common problem analysts face in building and estimating models, however, is that estimates of model parameters are often severely distorted by the presence of extreme values in the data. One of the advantages of EDA techniques is the use of resistant estimators that are less sensitive to the biases caused by outliers than are conventional estimators.

For example, the median is a resistant measure of the center of a distribution. To illustrate, use the `homeless.dta` file on the fifty largest U.S. cities. Type '`summarize pop, detail`' to get descriptive statistics on the distribution of population size (in thousands) of these cities. Note the mean is 689 and the median is 368. Now type '`summarize pop if pop<7000, detail`'. (This eliminates one extreme outlier, New York City.) The result is that through deleting two percent of the data (one case), the mean dropped 132 points while the median dropped only 2. The median is resistant to the distorting effects of this one case but the mean is not.

Residuals

Another reason for preferring resistant measures is that it's easier to isolate a wild value and assess its true magnitude. Non-resistant measures can appear to camouflage the culprit case by making it look as if all the cases are some distance from the mean. Here is a simple example comparing mean and median in some imaginary data:

Case #	Data	X_i–mean	X_i–median
X_1	5	-195	-10
X_2	10	-190	-5
X_3	10	-190	-5
X_4	20	-180	+5
X_5	25	-175	+10
X_6	1130	+930	+1115
Mean	200		
Median	15		

Notice when the mean is subtracted from the data how the wild case X_6 spreads its "error" around to the other five cases and almost looks unobtrusive. When the median is subtracted, on the other hand, case X_6 looks quarantined from the rest, and its unique wildness is difficult to miss. A theme of EDA is to make outliers stand out through the use of resistant measures of distribution parameters.

Stata has an excellent toolkit for analyzing residuals and for detecting and dealing with outliers including an array of residual measures in options to the regression `predict` command, smoothing programs for time series, robust regression, and `t2way` for median polishing of tabular data. Stem-and-leaf/box plots help to show graphically just how extreme a given point is. Stata's influence statistics in regression diagnostics (e.g, `cooksd`, `dbeta`) help the analyst decide whether to remove a particular case

and re-estimate the model. (One should always stop and think carefully, of course, before clipping any data from analysis. See Lawrence Hamilton's dramatic case study of the shoddy analysis that contributed to the Challenger shuttle disaster.[2])

Re-expression

This is an EDA term for transformation of variables to a different scale of measurement (e.g., logs, square roots) to promote symmetry, constant variance, linearity and additivity in the data.

Here's a quick illustration. Use the `world.dta` file on 86 countries. Type '`graph gnpcap, oneway`', then '`graph gnpcap, norm`'. Notice how most data points crowd together at the lower end of the `gnpcap` scale, while the rest straggle out toward the higher end. This pattern illustrates severe positive skew, thus violating the normality assumptions underlying many statistical methods, including OLS regression. Now "re-express" `gnpcap` by transforming it to a logarithmic scale with the command '`generate loggnp = log(gnpcap)/log(10)`'. Now type '`graph loggnp, oneway`' and then '`graph loggnp, norm`' to see the difference. Notice how the log transform of `gnpcap` pulls the lower-end data points apart and squishes the upper-end data points together. To get a better feel for the re-expression taking place, type '`graph loggnp gnpcap, oneway twoway`'. See the pulling and the squishing?

Some analysts feel uncomfortable abandoning a variable's original and more familiar measurement scale in this way. After all, we don't buy bread with "log dollars" but with real ones. Others think "re-expression" is just a fancy word for cheating. But often the re-expressed scale makes practical sense. For example, the reciprocal of miles per gallon (mpg) is $1/\mathrm{mpg}$, which is a standard measure of gas mileage performance on a one-mile test track. (`generate` this reciprocal using Stata's `auto.dta` file and see the results.) And often the re-expressed scale makes good intuitive sense. For example, a strong case can be made that raising John's income from $1,000 to $10,000 yields John the same utility that Fred receives when Fred's income is raised from $100,000 to $1,000,000. The log transform captures this psychic if not monetary equivalence. In log dollars, the net increases for both people are the same—John's net increase is $4 - 3 = 1$, and Fred's net increase is $6 - 5 = 1$. Even if re-expression makes no intuitive sense, however, statistically it is often necessary if we aim to model the structure that actually exists in our data. It is fanciful to expect the world to adapt to arbitrary human yardsticks; the yardsticks must adapt to it.

Revelation

Stata's powers of revelation make it a superb EDA tool. Many of Stata's analytical graphics were inspired by the work of EDA pioneers, including John Tukey and William S. Cleveland. The toolkit includes box plots, one-way scatters, leverage plots, scatter matrices, hilite scatters, qnorm plots and qqplots, stem-and-leaf plots, t2way plots, and many others—all with display and printing options galore. If we must *see* the data, Stata is an excellent choice and a top value. Stata is particularly suited to the iterative and interactive style of EDA because Stata is very fast. There are other statistical packages that produce equally pretty pictures, but one can go out for a shave and a haircut while waiting for the images to appear on the screen.

Most EDA advocates would agree that the box plot (sometimes known as the box-and-whisker plot) is the single most useful analytical graphical tool for univariate analysis. The standard box plot, in one image, shows information about a distribution's center, spread, shape and outliers. Box plots are most useful for comparing distributions of multiple batches of data on a single scale. See *Figure 2*, which shows separate box plots of campaign spending by Democrats and Republicans in the California Congressional delegation. (Use `caldelg.dta`, type '`sort party`', then '`graph exp88w, by(party) s([name]) box`'.[2a])

The upper and lower horizontal lines of each box show the 75th and 25th percentile values, respectively. The height of each box measures the interquartile range or "midspread" containing the middle 50% of the cases. The horizontal line inside each box shows the median. Its location within the box gives a clue as to the symmetry or skewness, i.e., the "shape" of the distribution. If it's low in the box, that suggests positive skew; if it's high in the box, that suggests negative skew. The "whiskers" stretching out from the top and bottom of each box touch what are called "adjacent values." These are the values of cases nearest to but not beyond the "fences" that serve as dividing lines for identifying positive and negative outliers. The outliers themselves, in this case, are identified by name using string variables as plotting symbols.

From the information shown in Figure 2, we conclude that Republicans as a group spend only a bit more than Democrats on campaigns, that spending is more variable among the Republicans, that spending in both parties appears approximately normally distributed (emphasis here on the statistical meaning of "normal"), and that Republicans in 1988 had more "high rollers" (i.e., positive outliers) than the Democrats. (Note: the `caldelg.dta` file contains many variables one might explore in building models to explain these patterns, including how much the rival candidates spent in that election.)

Box plots have limitations, however, as illustrated in *Figure 3* and the output below. (Use the `xyzbox.dta` file, type '`graph x y z, box`' to get the box plots, then '`stem x`', '`stem y`', and '`stem z`' to get the stem-and-leaf plots.) In Figure 3, one sees that the box plots for the `x`, `y`, and `z` variables are identical. Yet the distributions they summarize are dramatically different, as shown by the stem-and-leaf plots:

Stem-and-leaf plot for x	Stem-and-leaf plot for y	Stem-and-leaf plot for z
0 | 05	0 | 0	0 | 059
1 | 05	1 |	1 | 5
2 | 28	2 | 555555555	2 | 0
3 | 15	3 |	3 | 0
4 | 5	4 |	4 | 0
5 | 005	5 |	5 | 000000
6 | 59	6 |	6 | 0
7 | 28	7 | 555555555	7 | 0
8 | 5	8 |	8 | 05
9 | 05	9 |	9 | 05
1 || 00	0 || 00	1 || 00

`x` is a rectangular distribution, `y` is bimodal, and `z` is approximately normal. As always, it's important to use several different tools on the same data to be sure one isn't missing something.

The graphs shown in *Figures 4a* and *4b* illustrate Stata's powers of revelation and also the insights to be gained from re-expression. Figure 4a is a scatterplot of the relationship between life expectancy and GNP per capita in 86 countries. (Use `world.dta`, then type '`graph life gnpcap`'.) This is a good picture of what a nonlinear relationship looks like. If we now recklessly pretend this relationship is linear, we can run a regression (type '`regress life gnpcap`') and get an adjusted R^2 of .44.

Enough pretending. Following the EDA guide of the "ladder of powers"[3], we decide to log transform the independent variable, `gnpcap`. (Type '`generate loggnp = log(gnpcap)/log(10)`'.) We now graph the same relationship, this time using `loggnp` rather than `gnpcap` as our predictor. The results are shown in *Figure 4b*. (Type '`graph life loggnp`' to see an unedited version of Figure 4b.) As can be seen, the relationship is much more linear, and we now have the green light to run a regression, which yields an adjusted R^2 of .71—not a bad improvement using this measure of model fit.

The EDA analyst would not stop here, of course, because there is obvious structure in the rough around the linear fit. (See the S-shaped pattern in the scatter around the regression line?) To see this structure more clearly, type '`regress life loggnp`', then '`predict liferes, residual`', then '`graph liferes loggnp, yline(0)`'. A third iteration of analysis might try a logistic model. That would improve the fit still further and have substantive implications for our theory. And so on to a fourth iteration.

Notice one other thing. Look back at Figure 4a. See those two data points just to the right of the "70" tick on the life scale? Those happen to be China and Sri Lanka, which are identified as such in Figure 4b. Clearly, the populations in those two countries have a much higher average life expectancy than what is predicted given their level of economic development. How did we miss that in Figure 4a? According to William S. Cleveland's studies of graphical perception, people have a very hard time gauging *vertical* distances between points and a curve that moves from left to right with a rapidly changing slope.[4]

Stata's leverage plot is another valuable EDA graphical tool. Here's an abbreviated example using the `homeless.dta` file, which is taken from William Tucker's recent book on the causes of homelessness in large U.S. cities.[5] Figure 5 shows a leverage plot for the relationship between the *residuals* of the log of the city's homeless per thousand population and the *residuals* of average city temperature *controlling for* rent control (dummy), median rent, and population growth. The slope of the line is .021, which exactly equals the estimated partial regression coefficient for temperature. (Use the `homeless.dta` file, type '`regress loghomel temp rentcont medrent growth`'. Then type '`leverage loghomel temp rentcont medrent growth`'.)

In Figure 5, notice the data point for Lincoln, Nebraska in the lower left corner, far from the others. Lincoln represents only two percent of the data, yet eyeballing the chart suggests it exerts excessive "leverage" on the partial regression slope. If we removed that one case from analysis, the slope might flatten out to the point of statistical insignificance. We can use Stata's DF-beta influence statistic to measure each city's influence in determining the slope. (Type '`dbeta loghomel temp rentcont medrent growth, generate(tempbeta)`', then '`sort tempbeta`', then '`list city tempbeta`'.) Lincoln's DF-beta is 1.54, a very large value. After using `predict` with the `cooksd` option to check Lincoln's overall influence in determining the regression results, we decide to drop Lincoln from the analysis and re-estimate the model. When we do so, the adjusted R^2 drops from .33 to .26, and the slope estimate for temperature become statistically insignificant ($p > .05$). That's a lot of clout for *one* case. Thanks to the leverage plot, we discovered it and took proper steps to deal with it.

These examples illustrate only a fraction of Stata's analytical graphical power as an EDA tool.[6]

Notes

1. See David C. Hoaglin, Frederick Mosteller, and John W. Tukey, eds., *Understanding Robust and Exploratory Data Analysis* (John Wiley & Sons, 1983). The introduction discusses the "4 R's." Also see John W. Tukey, *Exploratory Data Analysis* (Addison–Wesley, 1977); Frederick Mosteller and John W. Tukey, *Data Analysis and Regression: A Second Course in Statistics* (Addison–Wesley, 1977); William S. Cleveland, *The Elements of Graphing Data* (Wadsworth, 1985); Paul Velleman and David C. Hoaglin, *Applications, Basics, and Computing of Exploratory Data Analysis* (Duxbury Press, 1981); Catherine Marsh, *Exploring Data* (Polity Press, 1988), and Lawrence Hamilton, *Modern Data Analysis* (Brooks/Cole, 1990).
2. Hamilton, 458–460.

2a. Note: most graphs shown in this article have been enhanced for presentation using Stage, the Stata Graphics Editor. In some cases, placement of data points has been slightly adjusted to improve readability.

3. See, for example, Velleman and Hoaglin, 48–50; Marsh, 205–209; Hamilton, 163–165; Mosteller and Tukey, 79–88.
4. Cleveland, 276–279.
5. William Tucker, *The Excluded Americans: Homelessness and Housing Policies*, (Regnery Gateway, 1990).
6. Data sets on disk: `ans1.dta`, `ans2.dta`, `ans3.dta`, `ans4.dta`, `world.dta`, `homeless.dta`, `caldelg.dta`, `xyzbox.dta`.

Figures

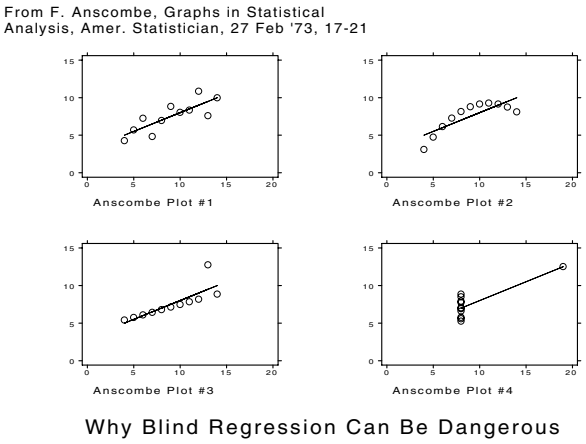

Figure 1

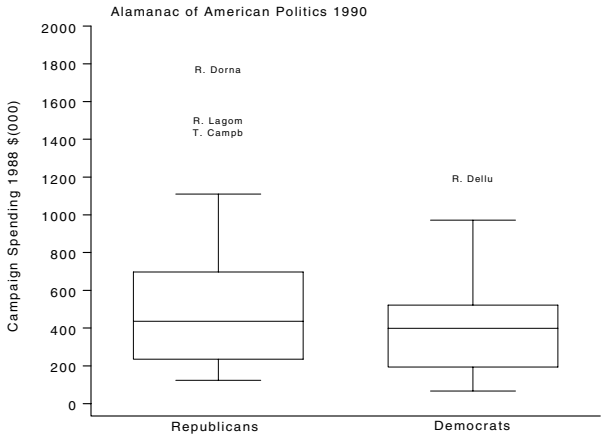

Figure 2

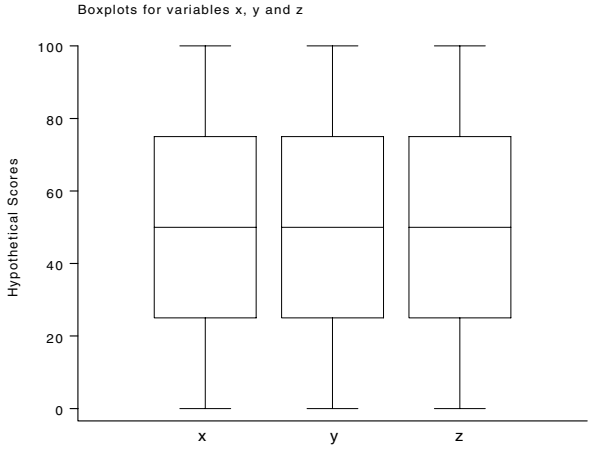

Figure 3

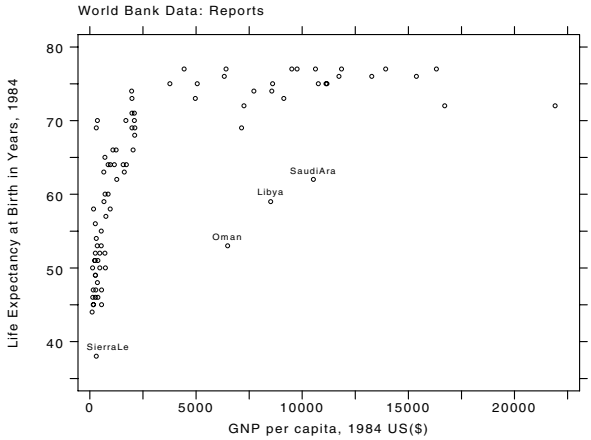

Figure 4a

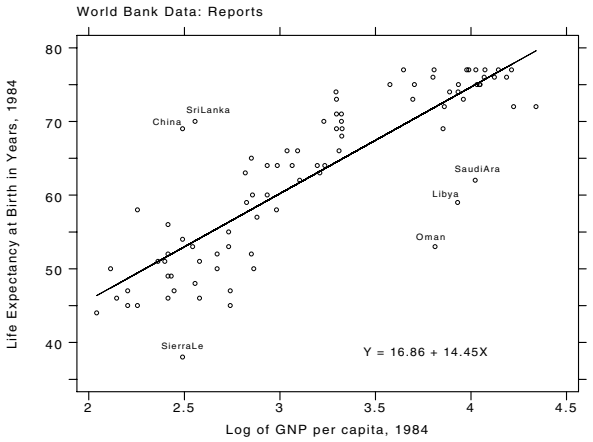

Figure 4b

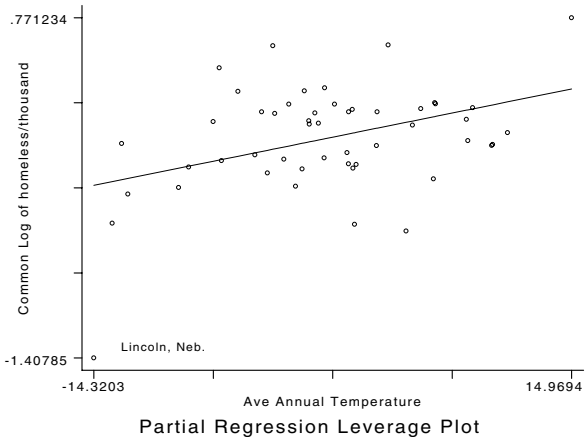

Figure 5

sg1 Nonlinear regression command

Francesco Danuso, Istituto di Produzione Vegetale, Udine, Italy. FAX (011)-39-432-558603

[The English adaptation and revision is by Ed. The program described here represents the consolidation of eight modules into a single, menu-driven ado-file that offers the user a variety of options. The Editor has compared results with those of SAS *and* SPSS *using both a logistic function and a negative exponential growth curve model. Other formulae have not, as yet, been evaluated. Sample output for these data runs are presented in this insert.]*

`nonlin`, provided on the STB diskette, uses a modified Gauss-Newton iterative method for estimating the parameters of a nonlinear function using least squares. The user does not need to provide derivatives; the program uses a Taylor-series algorithm.

`nonlin` is menu-driven. After loading the data into memory, one simply types '`nonlin`' and answers the questions:

```
. infile yvar xvar using nltest1
(9 observations read)

. list
         yvar       xvar
  1.      .04          5
  2.      .06         12
  3.      .08         25
  4.       .1         35
  5.      .15         42
  6.       .2         48
  7.      .25         60
  8.       .3         75
  9.       .5        120

. nonlin
                    NONLINEAR REGRESSION

 Menu:              0. Exit
                    1. Model Fit
                    2. Frequency distribution of residuals (Yo-Yp)
                    3. Graph Yp vs. Yo
                    4. Graph of residuals vs. Yp
                    5. Graph of residuals vs. Xi
                    6. Graph of Yp vs. Xi
 Choice: . 1

 Cases (1=all): . 1

 Variable Y   : . yvar
 List of Xi   : . xvar
 Function     : . 1/(%b1+%b2*exp(%b3*xvar))
 # parameters : . 3

Initial value for b1. 2
Initial value for b2. 30
Initial value for b3. -0.04
How many iterations:. 4
```

A detailed description of the questions and possible answers follows:

`Choice`
Choose '1' if you wish to estimate a model. The other choices are shown on the opening menu and should be self-explanatory.

`Cases`
Choose '1' if you wish to estimate the model using all the observations. Alternatively, you may type a condition just as you would after an `if` on an ordinary Stata command. Typing '`xvar==1`', for instance, would use only observations for which the value of `xvar` is 1.

`Variable Y`
Type the name of the dependent variable.

`List of variable Xi`
Type the list of independent variables just as you would on the Stata command line: '`xvar1 xvar2 xvar3`'.

`Model`
Type the right-hand side of the regression function using the names `%b1`, `%b2`, etc., for the names of the parameters to be estimated. For example, the logistic function `yvar` $= 1/(C + A * \exp(B * \texttt{xvar}))$ would be typed as '`1/(%b1+%b2*exp(%b3*xvar))`'.

`How many parameters`
Type the number of parameters; in the example above, we use `%b1`, `%b2`, and `%b3`, so we type '3'.

`Initial value for b1`
`Initial value for b2`, etc.
Type an estimate for each of the parameters. These will be used as the starting values.

`How many iterations`
Type the initial number of iterations to be performed. You will be asked later if you wish to perform additional iterations. `nonlin` will then perform the indicated number of iterations unless the estimates are clearly inappropriate or the model is inconsistent. `nonlin` is fairly sensitive to starting values that are relatively distant from the true parameters.

After the iteration run, the following selection menu will appear:

```
Other initial values->p; Out_u; Continue->c:
Type ´p´ if you wish to input new initial values.
Type ´u´ if you wish to see the output.
Type ´c´ if you want to proceed through more iterations.
```

`nonlin` does not provide a convergence criterion—at each iteration, the current parameter estimates are shown and it is left to you to decide when the process has converged. Choice '`p`' allows you to enter new starting values and restart the estimation. Choice '`u`' will display the results after asking whether you also wish them listed on the printer. Choice '`c`' will perform more iterations.

The following is an example run using a logistic model. The output is continued from the previous example—we have just typed that we want '4' iterations:

```
How many iterations:. 4
 Iteration n.  1
 B1= 1.7541885
 B2= 25.007356
 B3= -.03914615
 Iteration n.  2
 B1= 1.7808726
 B2= 25.737273
 B3= -.03927536
 Iteration n.  3
 B1= 1.7809655
 B2= 25.739122
 B3= -.039262
 Iteration n.  4
 B1= 1.7809352
 B2= 25.738384
 B3= -.03926107
 Other initial values ->p; Out ->u; Continue ->c :. c
How many iterations: . 1
 Iteration n.  5
 B1= 1.7809337
 B2= 25.738358
 B3= -.03926103
```

```
 Other initial values ->p; Out ->u; Continue ->c :. u
 Print of the results (y/n): . n
          =====NONLINEAR REGRESSION RESULTS=====
 File:                                    N. of iterations: 5
 Variable Y  : yvar
 Variables Xi: xvar
 Model: yvar=1/(1.7809336+25.738356*exp(-.03926103*xvar))
 Data selection: if 1
 Residual statistics
Residual Average = -.00053143      Stand. Dev. = .01430633
Skewness         = .14967284       Kurtosis    = 2.3239515
------------------------------------------------------------
  Variation       d.f.         SS                MS
------------------------------------------------------------
  Model            3         .48496016        .16165339
  Residual         6         .00163991        .00027332
  Total            9         .48660007        .05406667
  Corr Total       8         .173             .021625
------------------------------------------------------------
  R^2 = .9905
------------------------------------------------------------
  Parameter        Standard Error         t         Prob. t
------------------------------------------------------------
 b1  1.7809336       .11585868      15.371603      4.791e-06
 b2  25.738356       4.8631533      5.2925241      .00184296
 b3  -.03926103      .00414855      -9.4637958     .00007924
------------------------------------------------------------

 === CORRELATION COEFFICIENT AMONG PARAMETERS ===
        |       a1       a2       a3
--------+---------------------------
      a1|   1.0000
      a2|   0.6164   1.0000
      a3|  -0.7874  -0.9346   1.0000
```

For those who are not familiar with nonlinear ANOVA tables, Model or Regression Sum of Squares represents the sum of the squared predicted values $\sum Y_p^2$. Residual SS takes its usual meaning, $\sum (Y_o - Y_p)^2$ while the Uncorrected Total Sum of Squares is the sum of the squared dependent variable values, $\sum Y_o^2$. The Corrected Total SS is the sum of squared deviations from the mean. R^2 represents the proportion of total variation of the dependent variable (`yvar`) from its mean that is predicted by the model. R^2 can take on negative values if the model fails to fit as well as the mean. (Confidence intervals based on the listed standard errors of the estimated parameters are asymptotic and correlation coefficients are based on asymptotic approximations. The usual cautions regarding highly correlated parameters apply.)

[I invite comments, certifications, encountered problems, and suggestions. You may fax Dr. Danuso directly or contact me and I will forward any comments to Dr. Danuso for his response.—Ed.]

sg2 | Exact and cumulative Poisson probabilities

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

Then syntax of the `epoisson` and `cpoisson` commands is

`epoisson` *population_ratio total_observations cases*

`cpoisson` *population_ratio total_observations cases*

These commands act as a hand calculator in that they display the calculated result.

Suppose that you know that the overall 1990 hospital mortality ratio for the state of Arizona—the total number of deaths divided by the total number of admissions—is .03 (which it is not). You wish to determine the exact Poisson probability for 20 deaths which occurred in 1990 at St. Nowhere Hospital in Elsewhere, AZ. The hospital's total admissions for the year are 845:

```
. epoisson .03 845 20
  Exact Poisson probability => .04831361
```

Twenty out of 845 is a significant (at the 5% level) departure from what we might expect. `cpoisson` works in the same way:

```
. cpoisson .03 845 20
  Cumulative Poisson probability => .88035984
```

Both commands are included on the STB diskette.

sg3 Skewness and kurtosis tests of normality

William Gould, CRC, FAX 213-393-7551

You have a sample of x_i's and you want to know whether you can reject the hypothesis that the x_i's have been drawn from a normal distribution. Such tests have recently been based on the observed skewness and kurtosis of the sample x_i's—normal distributions have skewness $s_3 = 0$ and kurtosis $s_4 = 3$—and so should their empirical counterparts. We recently introduced the `sktest` command to perform this test.

Our test is a variation on the "well-known" (although we admit that we did not know it at the time) Bera–Jarque test which is discussed, for instance, in Judge (1988). In this test, $Z_3 = s_3\sqrt{\bar{x}}/\sqrt{6}$ is the test of skewness while $Z_4 = (s_4 - 3)\sqrt{\bar{x}}/\sqrt{24}$ is the test of kurtosis. Both statistics are distributed $\mathrm{N}(0,1)$. The combined test statistic is $Z_1^2 + Z_2^2$ and is distributed $\chi^2(2)$. Our variation substituted the empirically determined numbers 2.277 and 4.81 for $\sqrt{6} = 2.449$ and $\sqrt{24} = 4.899$, respectively.

Another variation was recently reported by D'Agostino, et al., (1990). This variation substitutes more complicated expressions for Z_1 and Z_2 but combines the two statistics in the same way. Their Z_1 is an approximation by D'Agostino and their Z_2 is an approximation by Anscombe and Glynn.

Thus, the question arises: Which is better? The results of the following Monte-Carlo experiments may help you answer the question:

True Distribution	Test	1%	5%	10%	True Distribution	1%	5%	10%
Normal					Contaminated Normal			
	sktest	.024	.054	.084		.967	.971	.973
	Bera-Jarque	.020	.044	.070		.967	.970	.972
	D'Agostino	.018	.059	.100		.965	.970	.973
Uniform					Long-tail Normal			
	sktest	.007	.652	.938		.130	.216	.283
	Bera-Jarque	.002	.567	.914		.118	.197	.259
	D'Agostino	.985	.997	.999		.081	.179	.263
t(5)					t(20)			
	sktest	.549	.641	.693		.096	.151	.198
	Bera-Jarque	.535	.624	.677		.088	.135	.179
	D'Agostino	.453	.595	.673		.069	.137	.197
chi2(5)					chi2(10)			
	sktest	.926	.985	.995		.667	.834	.906
	Bera-Jarque	.892	.972	.992		.609	.786	.873
	D'Agostino	.883	.977	.995		.606	.806	.895

The numbers reported are the fraction of samples that are rejected at the indicated significance level. We performed tests by drawing 10,000 samples, each of size 100, from the indicated distribution. Each sample was then run through each test and the test statistic recorded.

For instance, in the upper-left of the table we report results when samples were drawn from a normal distribution. One would expect, then, that the recorded numbers under 1% would be .010; under 5%, .050; and under 10%, .100. Our interest in these results, however, is to compare the three tests against each other and not to compare performance to theoretical expectations. Thus, we will leave it to you to notice the over-rejection of all three methods at the 1% level and merely state that all three tests produce similar results.

The "Contaminated Normal" refers to elements drawn with $p = .95$ from a $\mathrm{N}(0,1)$ distribution and with $p = .05$ from a $\mathrm{N}(0,10)$ distribution. All three methods not only perform similarly, results are almost exactly the same.

The results under "Uniform" provide the most striking evidence in favor of the D'Agostino test. At the 1% level, both `sktest` and Bera–Jarque perform horribly, with `sktest` rejecting only 0.7% of the uniformly-distributed samples as non-normal, while D'Agostino finds the non-normality in 98.5% of the cases. By the 10% level, however, the simpler tests are performing adequately but not well.

Were this the end of the story, the evidence would be overwhelmingly in favor of the D'Agostino test. It is not, however. The distribution $\mathrm{t}(5)$ refers to a Student t distribution with 5 degrees of freedom, a not unreasonable real-life distribution. In this case, `sktest` outperforms D'Agostino, rejecting 55% rather than 45% of the samples at the 1% level. For a Student t with 20 degrees of freedom, `sktest` rejects 10% rather than 7% of the samples at the 1% level.

The "Long-tail Normal" refers to a doctoring of normal deviates by

```
. gen u = invnorm(uniform())
. replace u = 1.25*u if abs(u)>1.5 & uniform()>.5
```

Again, results are in line with the results obtained with the Student t distribution; at the 1% level, `sktest` rejects 13% of the samples while D'Agostino rejects 8%.

So far, all tests have concentrated on kurtosis. In order to check behavior to skewness, results were generated for $\chi^2(5)$ and $\chi^2(10)$ random deviates. Results again favor `sktest`. In the case of $\chi^2(10)$, at the 1% level `sktest` rejected 67% of the samples while D'Agostino rejected 61%.

So which is better? If one is going to use the simple formulation, these results indicate that the empirically-determined constants used by `sktest` outperform the theoretically-determined constants of Bera–Jarque. The choice is not so easily made between `sktest` and D'Agostino, however. D'Agostino provides strikingly better results for the uniform distribution, but this is contrasted against gently improved results by `sktest` for the less extreme t and χ^2 distributions.

I promised Richard Goldstein, who told me about these tests, that I would replace `sktest` with the D'Agostino calculation if the results clearly indicated the test's dominance. Given these results, I will leave the choice to you. Among the support materials associated with this insert on the STB disk is included `sktestd` which makes the D'Agostino calculation. You may prefer the results to `sktest`.

I have also included the materials to produce the above results. You can use these materials to perform other experiments or as the basis for learning how such experiments can be programmed in Stata. In the latter case, also see *ssi1*. You are warned that these tests are a considerable cpu burner. Performing all of the above tests took just over 6 hours on a DECstation 3100 running Ultrix, a very fast machine.

References

D'Agostino, R. B., A. Balanger, and R. B. D'Agostino, Jr. 1990. A suggestion for using powerful and informative tests of normality. *The American Statistician* 44(4): 316–321.

Judge, G. G. 1988. *Introduction to the Theory and Practice of Econometrics.* 2d ed. John Wiley & Sons, p. 891.

sqv1	Additional logistic regression extensions

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

The January 1991 *Stata News* and subsequent Support Disk provided users of the logit procedure with a number of additional statistics. Options are now available to calculate odds ratios and errors, 95% confidence intervals, delta*coef values, model predictive values, pseudo R-squares, ROC curves, and model fit statistics. Their use has been documented on the respective support disk help files; i.e., '`help logiodds`', `help logifit`', and '`help roc`'. The following program listing adds further extensions to logistic regression. It is called `extlogit.ado`. Changes must also be made to the calling program—`logiodds.ado`. Rather than changing the CRC-supplied `logiodds`, a new command, `logiodd2`, is provided. All materials are included on the STB disk.

The syntax for use of the (new) `logiodd2` command is similar to that of `logiodds`:

`logiodd2` *depvar varlist* [`=`*exp*] [`if` *exp*] [`in` *range*] [`, e f l st level(#) roc saving(`*filename*`)`]

The `e` option provides the additional statistics calculated by `extlogit.ado`. To get a quick look at the output provided by the new options, load an appropriate file and type

```
. logiodd2 depvar varlist, e f l st roc
```

The `e` option provides the following statistics:

1. Model goodness-of-fit statistics including a modified Hosmer–Lemeshow goodness-of-fit statistic, and both -2LL(model) and -2LL(intercept) statistics, all presented with chi-square significance levels. The modified H-M statistic is not based on replications of covariate patterns; hence, it will work when `logifit`'s goodness-of-fit statistics may not.
2. Wald statistics and their respective chi-square significance levels. Although the t significance values used by Stata are nearly the same as those for the Wald statistics, many programs supply Wald statistics. Comparisons can thus be made.
3. Partial correlation values between the classification or dependent variable and each of the independent variables. These values range from $+1$ to -1. Higher positive values indicate a higher likelihood for the success or occurrence of the model event; conversely, as values approach -1, success is less likely.

The '`e`' option also creates several diagnostic variables which can aid in assessing influence and fit. Among the foremost are `logpred`—probability of success. `resid`—residual. `stresid`—standardized (Pearson) residual. `hat`—diagonal of hat matrix.

`dev`—deviance. `cmod`—confidence interval displacement; measure of the effect of each covariate pattern on estimated parameter values (modified Cook's distance). `deltad`—measure of the effect of each covariate pattern on model fit, based on deviance. `deltax`—same as `deltad`, except based on Pearson chi-square.

The hat values are based on covariate replications; i.e., they represent a more correct m-asymptotic distribution (unlike several of the other major software packages). `delta*` values and the modified Cook's distance statistic, since they are calculated using hat values, are also based on m-asymptotic covariate distributions.

There are many obstacles in assessing influence and fit. For those who want a rather quick and unsophisticated look at their model, having first found it significantly acceptable, use the following three commands:

To check for cases influencing model fit:

```
. graph deltad logpred, xlab ylab yline(4)
. graph deltax logpred, xlab ylab yline(4)
```

Cases above 4 significantly effect fit. To check for cases influencing parameters:

```
. graph cmod logpred, xlab ylab yline(.9)
```

Cases above 0.9 significantly influence coefficient values.

The following example uses the `cancer.dta` data set found on the Stata disks. `drug` is changed to a set of indicator variables by '`tabulate drug, gen(drg)`' and `drg1` is then used as the reference variable. Only the `e` option is demonstrated.

```
. use cancer, clear
(Patient Survival in Drug Trial)
. tabulate drug, gen(drg)
   Drug type|
 (1=placebo)|      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         20       41.67       41.67
          2 |         14       29.17       70.83
          3 |         14       29.17      100.00
------------+-----------------------------------
      Total |         48      100.00

. logiodd2 died age drg2 drg3, e
Variable| Odds Ratio  Std. Error  [95% Conf. Interval]  Delta*Coef
--------+-------------------------------------------------------
    age |  1.975711*  .8030674   .8713186   4.479917     .6809284*
   drg2 |  .0287022   .0351004   .0024446   .3369952    -3.550783
   drg3 |  .0387934   .0458072   .0035966   .4184295    -3.249504
---------+-------------------------------------------------------
                      *) Note:  Delta = 1 SD rather than 1 unit.

MODEL GOODNESS OF FIT STATISTICS
** ChiSq>.05 fails to reject hypothesis that model fits
H-L Goodness of Fit (Mod) =>   56.5838
   ChiSq sign. (df:N-k)   =>    0.0966
-2 LL(Model)              =>   43.0511
   ChiSq sign. (df:N-k)   =>    0.5122
-2 LL(Intercept)          =>   62.3988
   ChiSq sign. (df:N-k)   =>    0.0352

WALD STATISTICS & PARTIAL CORRELATIONS
No  Var         Wald   Prob(Chi)    Partial Corr
================================================
1   age        2.8064   0.0939            0.1137
2   drg2       8.4305   0.0037            0.3210
3   drg3       7.5733   0.0059            0.2989

Additional diagnostic variables created...
    logindex =  Logit; Index value
    sepred   =  Standard error of index
    logpred  =  Probability of success (1)
    resid    =  Residual
    stresid  =  Standardized residual (Pearson)
    hat      =  Hat matrix diagonal
    dev      =  Deviance
    cmod     =  Influence on est parameter values
    deltad   =  Change in Deviance
    deltax   =  Change in Pearson chi-square
```

```
. list died logpred age drg2 drg3 deltad if deltad>=4
        died    logpred       age      drg2      drg3     deltad
 1.        1   .1793184        47         1         0   4.141695
41.        0   .9662145        58         0         0   9.167167
. list died logpred age drg2 drg3 cmod if cmod>=.8
        died    logpred       age      drg2      drg3       cmod
 1.        1   .1793184        47         1         0    .812959
41.        0   .9662145        58         0         0   2.591738
```

To obtain a graphical interpretation of the relation between `deltad` and `logpred`, weighted by `cmod`, type

```
. graph deltad logpred =cmod, xlab ylab yline(4) border t1(DELTAD by LOGPRED)
```

I have compared results produced by `extlogit.ado` to those of SAS and SPSS. In the jargon of each program, I have compared:

extlogit	SAS	SPSS
stresid	Pearson residual	Zresid
dev	Deviance residual	Dev
hat	Hat matrix diagonal	Leverage
cmod	C	Cook's D
deltax	DIFCHISQ	(none)
deltad	DIFFDEV	(none)

For `hat`, `cmod`, `deltax`, and `deltad`, comparisons were based on replications (m-asymptotic); hence `extlogit` gives different (preferable) values than SAS or SPSS for multiple-covariate-pattern, hat-value based statistics. By removing the following lines from the program code, Stata and SAS will yield identical results:

```
sort logpred
by logpred: replace hat=sum(hat)
by logpred: replace hat=hat[_N]
```

One final warning: The sort order of the data is changed by `logiodd2` and `extlogit`.

ssa1 | Actuarial or life-table analysis of time-to-event data

Henry Krakauer & John Stewart, Office of Research, Health Care Financing Administration

Stata already contains the Kaplan–Meier procedure for the analysis of time-to-event data with allowance for right censoring. This procedure is particularly useful with smallish data sets but becomes quite cumbersome with data sets containing tens of thousands of observations. Under those circumstances, the actuarial or life-table approach is both convenient and quite adequate. The theory is covered in, for example, *Analysis of Survival Data* by D. R. Cox and D. Oakes (1984, 53–56).

In the life-table method, times-to-event (failures and censorings or withdrawals without failure) are grouped into convenient intervals. The ratio of cases that failed to the number at risk at the beginning of the interval is computed for each interval, and, from this, the cumulative proportion of survivals and failures. An important assumption is that cases censored within an interval are at risk of failing for half the interval or, as implemented, that half the censored cases are at risk in that interval.

The program also permits the stratification of cases on the basis of values of their characteristics—e.g., whether a treatment was provided—and provides estimates of hazards and failures for each stratum at each interval. Two tests of equality over strata are provided. One is based on the likelihood-ratio statistic and the other is an invocation of the Stata `logrank` command which computes the log-rank statistic.

The user interface departs from the Stata standard as an experiment. The specification of the parameters are not passed through command-line arguments of the do-file (‘`run lftbl`’) but are instead provided by responding to questions asked interactively. Below is a sample session with only the graphical output omitted. (*Caution:* in specifying intervals other than the defaults, be certain that the start of the first interval is 0 and enter the intervals in ascending order. Otherwise, the results will be incorrect, but without warning of error.)

```
 . use \mkhm\rvsm
 . run lftbl
        SPECIFICATION OF REQUEST
(The phrases beginning with "Specify" and ending with ":" are
the prompts.)
Specify time-to-event variable:.  psurv
Specify event marker variable:.   rd
Specify time unit (day/week/month/year--day is default):. year
Specify grouping  variable (enter for none):  . rvs
```

```
       SPECIFICATION OF TIME CATEGORIES
This system provides the following default categories for the
life table (values are in days and starting value is first):
    0 - <  7     7 - < 15    15 - < 30    30 - < 60   60 - <90
   90 - <180   180 - <360   360 - <540   540 - <209    > 720
Specify starting value of first interval desired or just press
enter to use default values (the first interval must start with 0,
enter values in ascending order, press enter after entry):  . Enter Pressed

       CREATING DEFAULT INTERVALS.  THIS WILL TAKE A FEW SECONDS

          LIFE TABLES NOW BEING CALCULATED,
                    PLEASE WAIT.



                      RESULTS OF LIFE TABLE ANALYSIS--FAILED
Interval Beg Totl  Deaths  # Lost  Cum Fail Std Err Upper Lim Lower Lim
  0 days     2810     353     353  0.000000 0.0000   0.0000   0.0000
  7 days     2457     131     131  0.125623 0.0072   0.1396   0.1116
 15 days     2326     112     112  0.172242 0.0075   0.1870   0.1575
 30 days     2214      91      91  0.212100 0.0081   0.2280   0.1962
 60 days     2123      69      69  0.244484 0.0085   0.2611   0.2279
 90 days     2054     108     108  0.269039 0.0086   0.2860   0.2521
180 days     1946     166     195  0.307473 0.0092   0.3255   0.2895
360 days     1751     105     945  0.366992 0.0100   0.3865   0.3475
540 days      806      32     710  0.416928 0.0104   0.4374   0.3965
720 days       96       0      96  0.456882 0.0121   0.4805   0.4332
    VARIABLES:   TIME TO EVENT= psurv    EVENT MARKER=rd    TIME UNIT=year
                      DATASET USED:  \mkhm\rvsm.dta
                      GROUPING VARIABLE rvs = 0
                         PRESS ENTER TO CONTINUE.



                      RESULTS OF LIFE TABLE ANALYSIS--FAILED
Interval Beg Totl  Deaths  # Lost  Cum Fail Std Err Upper Lim Lower Lim
  0 days      280      18      18  0.000000 0.0000   0.0000   0.0000
  7 days      262       8       8  0.064286 0.0157   0.0950   0.0336
 15 days      254       7       7  0.092857 0.0179   0.1279   0.0578
 30 days      247       1       1  0.117857 0.0198   0.1567   0.0790
 60 days      246       1       1  0.121429 0.0196   0.1598   0.0830
 90 days      245       1       1  0.125000 0.0198   0.1639   0.0861
180 days      244       7      11  0.128571 0.0201   0.1679   0.0892
360 days      233       4     132  0.153778 0.0222   0.1973   0.1102
540 days      101       0      92  0.173807 0.0238   0.2205   0.1271
720 days        9       0       9  0.173807 0.0233   0.2194   0.1282
    VARIABLES:   TIME TO EVENT= psurv    EVENT MARKER=rd    TIME UNIT=year
                      DATASET USED:  \mkhm\rvsm.dta
                      GROUPING VARIABLE rvs = 1
                         PRESS ENTER TO CONTINUE.



                    RESULTS OF LIFE TABLE ANALYSIS--HAZARD
Interval Beg Totl  Cum Fail Std Err   Hazard   Std Err Upper Lim Lower Lim
  0 days     2810 0.00000   0.0000   0.01915   0.00102  0.02114  0.01716
  7 days     2457 0.12562   0.0072   0.00685   0.00060  0.00802  0.00568
 15 days     2326 0.17224   0.0075   0.00329   0.00031  0.00390  0.00268
 30 days     2214 0.21210   0.0081   0.00140   0.00015  0.00169  0.00111
 60 days     2123 0.24448   0.0085   0.00110   0.00013  0.00136  0.00084
 90 days     2054 0.26904   0.0086   0.00060   0.00006  0.00071  0.00049
180 days     1946 0.30747   0.0092   0.00050   0.00004  0.00057  0.00042
360 days     1751 0.36699   0.0100   0.00046   0.00004  0.00054  0.00037
540 days      806 0.41693   0.0104   0.00039   0.00007  0.00053  0.00026
720 days       96 0.45688   0.0121         .         .        .        .
    VARIABLES:   TIME TO EVENT= psurv    EVENT MARKER=rd    TIME UNIT=year
                      DATASET USED:  \mkhm\rvsm.dta
                      GROUPING VARIABLE rvs = 0
                         PRESS ENTER TO CONTINUE.
```

```
                    RESULTS OF LIFE TABLE ANALYSIS--HAZARD
Interval Beg Totl  Cum Fail Std Err   Hazard   Std Err Upper Lim Lower Lim
  0 days      280 0.00000   0.0000   0.00949   0.00224  0.01387  0.00511
  7 days      262 0.06429   0.0157   0.00388   0.00137  0.00656  0.00119
 15 days      254 0.09286   0.0179   0.00186   0.00070  0.00324  0.00048
 30 days      247 0.11786   0.0198   0.00014   0.00014  0.00040 -0.00013
 60 days      246 0.12143   0.0196   0.00014   0.00014  0.00040 -0.00013
 90 days      245 0.12500   0.0198   0.00005   0.00005  0.00013 -0.00004
180 days      244 0.12857   0.0201   0.00016   0.00006  0.00028  0.00004
360 days      233 0.15378   0.0222   0.00013   0.00007  0.00026  0.00000
540 days      101 0.17381   0.0238   0.00000         .        .        .
720 days        9 0.17381   0.0233         .         .        .        .
    VARIABLES:   TIME TO EVENT= psurv    EVENT MARKER=rd    TIME UNIT=year
                         DATASET USED:  \mkhm\rvsm.dta
                         GROUPING VARIABLE rvs = 1
                            PRESS ENTER TO CONTINUE.

Likelihood ratio test statistic for homogeneity (group=rvs):
Chi2 (1) = 84.136719,   P = 4.617e-20
Logrank test of homogeneity (group=rvs):
Group            Events            Predicted
------------------------------------------------
0                1167              1089.3527
1                47                124.64737
Chi2( 1 ) =  53.903946 , P =  2.105e-13
```

(Please send questions or comments regarding this insert c/o the Editor, STB*, and they will be forwarded.)*

ssi1 | Monte Carlo simulation

Lawrence C. Hamilton, Department of Sociology, University of New Hampshire

Computer-intensive methods like Monte Carlo simulation dramatically extend the frontiers of statistical knowledge, exploring problems too complex to solve by mathematical reasoning. Until recently, such work required mainframe computers—and even there, it was costly. Stata proves to be quite capable of Monte Carlo simulation, however. The slower pace of desktop computing is offset by the ease and speed of using Stata to investigate the sometimes surprising analytical results.

This article illustrates some Stata commands useful for Monte Carlo simulation, and presents an example Monte Carlo ado-file.

Random Variable Generation

Stata's random-number function, `uniform()`, adapts to generate random data from a variety of theoretical distributions. Random variables can be added to any existing data set through `generate` or `replace`. Alternatively, we can create artificial data sets from scratch. For example:

```
. set seed 111180
. set obs 50
. generate case = _n
. generate newvar = uniform()
```

The `set seed` command specifies the seed value for Stata's random number generator. Any large number will do; if this number is even, however, Stata adds 1 to make it odd (so 111180 and 111181 have identical effects). If we do not specify a seed, Stata always begins with 1001—and so always generates the same sequence of random numbers. Setting the seed deliberately helps to keep this fact in mind—we can either use the same seed as last time (to generate the same sequence) or choose a different seed (to ensure a different sequence), depending on our needs.

`set obs` tells Stata the new data set will contain $n = 50$ observations or cases. `generate case=_n` creates a variable holding case ID numbers from 1 to 50. `generate newvar=uniform()` creates a random variable consisting of sixteen-digit values sampled from a uniform (rectangular) distribution over the open interval from 0 to 1.

If we wish newvar to follow a uniform distribution over the interval $(0, 128)$ instead of $(0, 1)$:

```
. generate newvar = 128*uniform()
```

These will still be 16-digit numbers. Perhaps we want only integers from 1 to 128:

```
. generate newvar = 1+int(128*uniform())
```

The following sections give further examples of random variables.

Normal (Gaussian) Distributions

Standard normal distributions ($\mathrm{N}(0,1)$) have mean $\mu = 0$ and standard deviation $\sigma = 1$. To create a variable called `Z1`, with values drawn from a standard normal distribution:

```
. generate Z1 = invnorm(uniform())
```

Values from a normal distribution with $\mu = 500$ and $\sigma = 75$:

```
. generate X1 = 500+75*invnorm(uniform())
```

Two standard normal variables with a population correlation $\rho = .7$:

```
. generate Z1 = invnorm(uniform())
. generate Z2 = Z1*.7+invnorm(uniform())*sqrt(1-.7^2)
```

The .7 in this example could be replaced with any other correlation desired. After they are generated, variables can be moved to different means and standard deviations without affecting their correlation. For example, to simulate `Z1` $\sim \mathrm{N}(500, 75)$ and `Z2` $\sim \mathrm{N}(100, 15)$, after generating as above:

```
. replace Z1 = 500+75*Z1
. replace Z2 = 100+15*Z2
```

Sample means, correlations, etc., will not exactly equal the population parameters specified.

Other theoretical distributions

A is drawn randomly from an exponential distribution with mean $\mu = 2$:

```
. generate A = -2*log(uniform())
```

For other means, substitute different values for 2.

If Z follows a normal distribution, $B = exp(Z)$ follows a lognormal distribution. For example, to form a lognormal variable `B` based upon a standard normal distribution:

```
. generate B = exp(invnorm(uniform()))
```

The normal distribution may have any mean and standard deviation. Taking logarithms, of course, normalizes a lognormal variable.

A chi-square (χ^2) variable with d degrees of freedom (and mean $\mu = d$) equals the sum of d independent, squared standard normal variables. For example, to generate values from a χ^2 distribution with one degree of freedom:

```
. generate X2_1 = (invnorm(uniform()))^2
```

From a χˆ2 distribution with two degrees of freedom:

```
. generate X2_2 = (invnorm(uniform()))^2 + (invnorm(uniform()))^2
```

and so forth. Like exponential and lognormal distributions, χ^2 distributions are positively skewed and include no negative values. The skewness of χ^2 lessens with increasing degrees of freedom.

If `Z` is a standard normal variable and `X2_df` is a χ^2 variable, independent of `Z` and having `df` degrees of freedom, then the ratio `Z/sqrt(X2_df/df)` follows a Student's t distribution with `df` degrees of freedom. To generate random values from a t distribution with `df` $= 3$:

```
. generate Z = invnorm(uniform())
. generate X2_3 = (invnorm(uniform()))^2 + (invnorm(uniform()))^2
                + invnorm(uniform()))^2
. generate t = Z/sqrt(X2_3/3)
```

The steps could be combined. Like the standard normal distribution, t distributions are symmetrical, centered on zero, and range from positive to negative infinity.

The ratio of two independent χ^2 variables, each divided by its degrees of freedom, follows an F distribution. If `X2_df1` is distributed as χ^2(`df1`) and `X2_df2` is distributed as χ^2(`df2`), then `(X2_df1/df1)/(X2_df2/df2)` follows an F distribution with `df1` and `df2` degrees of freedom. For example, to generate values from $F(3, 12)$, first obtain χ^2 variables `X2_3` (sum of three squared $\mathrm{N}(0,1)$ variables) and `X2_12` (sum of 12 squared $\mathrm{N}(0,1)$ variables) as described above. Then

```
. generate F = (X2_3/3)/(X2_12/12)
```

F distributions contain only positive values.

Contaminated distributions

Contaminated distributions mix two or more simpler distributions. For example, assume Y is distributed:

N(0,1)	with probability .95
N(0,10)	with probability .05

Thus for 95% of the population, Y follows a standard normal distribution. For the remaining 5%, though, Y follows a normal distribution with much wider spread—a standard deviation of 10 instead of 1. Contaminated distributions, simulating data with a small proportion of wild errors, have often been employed in robustness research to test estimators' resistance to gross outliers. To draw samples from such a distribution:

```
. generate randnum = uniform()
. generate Y = invnorm(uniform())
. replace Y = 10*Y if randnum<.05
```

We first generate uniform random numbers, then generate `Y` as a standard normal variable. The third line multiplies `Y` by 10 (corresponding to a tenfold increase in standard deviation) for a randomly-selected subset of cases. Over the long run, about 5% of the cases will be so modified.

Example

Monte Carlo experiments generate random data, perform some analysis, record the results, then generate new data—repeated hundreds or thousands of times. Custom-written Stata ado-files can manage this operation. File `monte.ado`, listed below, carries out a simple experiment illustrating statistical properties of the mean and median, applied to 200 random samples of $n = 50$ values (from a standard normal population) each.

```
program define monte
set seed 31977
set more 1
log using monte
macro define _it 1
while %_it<201 {
     quietly {
          drop _all
          set obs 50
          generate Z=invnorm(uniform())
          summ Z, detail
     }
     display %_it
     display _result(3)
     display _result(10)
     display
     macro define _it=%_it+1
}
log close
drop _all
infile iterate mean median using monte.log
label data "Monte Carlo experiment"
label variable iterate "Iteration number"
label variable mean "Sample mean, Z normal"
label variable median "Sample median, Z normal"
save monte
end
```

Log file `monte.log` records iteration number, mean and median from each artificial sample. After 200 iterations, an `infile` statement reads this log file, labels the results, and saves file `monte.dta` containing 200 means and medians. Summarizing this data set should reveal both estimators' unbiasedness (means near zero, the population mean of standard normal Z) and the mean's superior efficiency compared with the median (i.e., its lower standard deviation).

Variations on the theme of `monte.ado` are straightforward—change the sample size, increase the number of iterations, or apply mean and median to nonnormal distributions. The mean's advantages with normally-distributed variables are well known, but see how mean and median perform when applied to heavier-tailed distributions like contaminated normals, for instance. A series of experiments, over a range of sample sizes, could empirically demonstrate the Central Limit Theorem.

Without much additional work, this approach adapts to more serious research purposes. In *Regression with Graphics* (in press, Brooks/Cole) I present results from Monte Carlo evaluations of ordinary least squares and two robust regression methods, applied to three different regression models. The experiment highlights strengths and weaknesses of each method. All calculations for this experiment (and for the book as a whole) were performed using Stata.

In Practice

Monte Carlo simulation with a personal computer is quite feasible, but requires patience. The `monte.ado` program takes about 19 minutes to execute on an 8-MHz 286, or less than two minutes on a 16-MHz 386 running Intercooled Stata. More ambitious projects may tie up even the fastest computers, but since "core time" is free, the machines can be left to crunch numbers all night.

For background research on *Regression with Graphics*, I wrote a Monte Carlo program that generated and regressed 100 pairs of contaminated (X, Y) values. The program repeated this operation for 800 artificial $n = 100$ random samples. Within each sample, the program performed 1,000 bootstrap iterations—resampling and reestimating 800,000 regressions. This kept a 386 busy for several days; meanwhile a different computer worked on another 800,000 $n = 10$ samples. The goal was evaluation of bootstrap confidence interval procedures with ill-behaved data. Recent hardware/software advances (notably Intercooled Stata) have brought such work for the first time into the realm of personal computing.

At the other extreme, simple programs like `monte.ado` (if necessary, scaled down to 50 or 100 iterations) can make Monte Carlo experiments accessible even to students stuck with the slowest equipment. This provides a hands-on, empirical approach to understanding abstract ideas like standard error, bias, efficiency, and robustness. And it is not a large step from pedagogically motivated experiments with predictable outcomes (like sampling behavior of mean and median with normal data) to curiosity and original research.