STATA
TECHNICAL
BULLETIN

March 1993
STB-12

A publication to promote communication among Stata users

Editor

Joseph Hilbe
Computing Resource Center
1640 Fifth Street
Santa Monica, California 90401
310-393-7551 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
Lawrence C. Hamilton, Univ. of New Hampshire
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

an1.1 STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:

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

Statistical Categories:

sbe	biostatistics & epidemiology	*srd*	robust methods & statistical diagnostics
sed	exploratory data analysis	*ssa*	survival analysis
sg	general statistics	*ssi*	simulation & random numbers
smv	multivariate analysis	*sss*	social science & psychometrics
snp	nonparametric methods	*sts*	time-series, econometrics
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified

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

an29 New edition of Statistics with Stata published

Ted Anderson, CRC, FAX 310-393-7551

Statistics with Stata 3, 194 pp and diskettes, by Lawrence C. Hamilton of the University of New Hampshire and published by Duxbury Press (ISBN 0-534-18918-0 book only; 0-534-18920-2 book with DOS diskettes) has just been released and is available from Computing Resource Center and other sources. (The book with Macintosh diskette will be released shortly.)

I wish to recommend this book to all of our users. While the book is intended for students as a companion to other statistical texts (such as Hamilton 1990 and 1992), the previous edition of this book has been popular with all of our users as an introduction to Stata. The new edition is even better because it focuses on a wider range of Stata's features.

The contents include

Introduction to Stata: Starting and exiting Stata (DOS and Macintosh); Sample Stata session; Printing and saving results; Menus; Using the keyboard; Online help.

Data: Typing in a new dataset; selecting cases and correcting mistakes; Adding variables or cases; String variables; Labeling data, variables, and values; Rearranging a dataset; Inputting from a raw-data file; Inputting from other programs; Combining datasets; Large datasets; Stata programming; Generating new variables; Random numbers and sampling; Case weights.

Graphs: Histograms; Time Plots; Scatterplots; Boxplots; Symmetry and quantile plots; Bar, pie, and star charts; Saving, printing, and combining graphs; Stata Graphics Editor.

Frequency distributions and univariate statistics: Summarizing; Frequencies and crosstabulations; Tables of means; Multiple oneway or twoway tables.

t **tests, ANOVA, and nonparametric comparisons**: Paired-difference tests; Two-sample tests; Oneway ANOVA; Error-bar charts; N-say ANOVA; Analysis of covariance.

Bivariate regression: The regression table; Predicted values and residuals; Graphing; Correlation matrices; Regression with transformed variables.

Multiple regression: Basics; Dummy variables; Testing hypotheses; Stepwise; Conditional effect plots; Weighted least squares and instrumental variables.

Regression diagnostics: Correlation and scatterplot matrices; Residual vs. predicted plots; Autocorrelation; Nonnormality; DFbetas; Other case statistics; Multicollinearity.

Fitting curves: Nonparametric methods of band regression and lowess smoothing; Regression with transformed variables; Conditional effect plots; Nonlinear regression.

Robust regression: Regression with idea data; Outliers and leverage; Asymmetrical error distributions; Robust analysis of variance.

Logistic regression: Two outcomes; Conditional effect plots; Diagnostic statistics and graphs; Multiple, ordered outcomes; Multinomial logistic regression (multiple, unordered outcomes).

Principal components and factor analysis: Principal components; Rotation; Factor scores; Principal factoring; Maximum-likelihood factoring.

Common Stata commands: Data and variables; Diagnostic statistics; `egen` (Extensions to `generate`); Epidemiological tables; Functions; Graphs; Miscellaneous statistics; Model fitting; Quality control; Stepwise variants of model fitting; Survival analysis; Tests.

Monte Carlo and bootstrap methods: Random number generator; Resampling statistics; Sampling from theoretical probability distributions; Monte Carlo examples of the Central Limit Theorem and comparing median and mean; Regression models; Bootstrapping.

Included with the book are 3.5-inch diskettes containing Student Stata and the data used throughout the book—instructions are provided for installing the data without Student Stata for persons who already own Stata. A no-charge order form is provided to exchange the 3.5-inch diskettes for 5.25-inch diskettes.

The text with disks is available for $35 from CRC or may be obtained from your usual Duxbury source.

References

Hamilton, L. C. 1990. *Modern Data Analysis: A First Course in Applied Statistics.* Pacific Grove, CA: Brooks/Cole.

——. 1992. *Regression with Graphics: A Second Course in Applied Statistics.* Pacific Grove, CA: Brooks/Cole.

an30 Editorship turnover

Joseph Hilbe, Editor, STB, FAX 602-860-1446, EMAIL atjmh@asuvm.inre.as

With this issue, the *Stata Technical Bulletin* now completes its second year. During this time, the STB has grown from an idea to a publication that enjoys widespread circulation and acceptance. Contributors range from the novice user to statisticians of international renown. Articles—or inserts, as we have called them—have brought the Stata user programs which may have otherwise not been available for some time, including tests for normality, repeated measures ANOVA, Box–Cox transformations, life-tables, nonlinear smoothing, generalized linear models, and nonlinear regression. Moreover, the STB has provided users with a host of diagnostic tools as well as tips on how to get the most from Stata; the inserts on programming in Stata seem most noteworthy in this respect. All in all, the STB appears to be living up to what I had hoped for some two years ago.

The job of being editor has grown with the STB and I now think it is time, both for myself and the STB, that the job be passed along to another who will, I am certain, find the experience most rewarding.

I have thoroughly enjoyed working with you in making the STB what it currently is and will be staying on as an Associate Editor. I will continue making contributions to the STB and look forward to the many excellent inserts which will be provided by CRC and users in the future.

I am, and will always be, interested in communicating with Stata users but I must now request that if those communications have to do with the STB as opposed to with me, an author an interested, active Stata user, those communications be directed to the new editor, Sean Becketti. Submissions, in particular, should be sent to:

Sean Becketti, Editor, STB, Computing Resource Center, 1640 Fifth Street, Santa Monica, CA 90401

crc27 More extensions to generate: categorical variables

The new `egen` function (see [5d] egen) `group(`*varlist*`)` creates a single variable taking on values 1, 2, . . ., for the groups formed by *varlist*. The syntax of `egen` with the `group()` function is

`egen` [*type*] *newvar* `= group(`*varlist*`)` [`if` *exp*] [`in` *range*] [`, missing`]

varlist may contain string, numeric, or both string and numeric variables. `missing` indicates that missing values in *varlist* are to be treated like any other number when assigning groups instead of missing values being assigned to the group missing.

Example 1. You have a variable `agegrp` which takes on the values 24, 40, 50, and 65, corresponding to age groups 18–24, 25–40, 41–50, and 51 and above. Perhaps you created this coding using the `recode()` function (see [2] functions and [4] catvars) from another age-in-years variable:

```
. gen agegrp=recode(age,24,40,50,65)
```

You now wish the codes were 1, 2, 3, and 4:

```
. egen agegrp2 = group(agegrp)
```

Example 2. You have two categorical variables, `race` and `sex`, which may be string or numeric. You want to use the `ir` (see [5s] epitab) to create a Mantel–Haenszel weighted estimate of the incidence rate. `ir`, however, will allow only one variable to be specified in its `by()` option and that variable must be numeric. You type

```
. egen racesex = group(race sex)
. ir deaths smokes pyears, by(racesex)
  (output omitted)
```

The new numeric variable `racesex` will be missing wherever `race` or `sex` is missing (meaning `.` for numeric variables and `""` for string variables), so missing values will be handled correctly. When we list a small amount of data, we see:

```
. list race sex racesex
        race        sex    racesex
 1.    black     female          1
 2.    black       male          2
 3.    white     female          3
 4.    white       male          4
 5.    white       male          4
 6.                female          .
 7.    white                     .
```

`group()` began by putting the data in the order of the grouping variables and then assigned the numeric codes. Note that both observations 6 and 7 were assigned to `racesex==.` because in one case the race, and in the other the sex, were not known. (These observations were then not used by the `ir` command.)

Had we wanted to treat the unknown groups just as we would any other category, we could have typed

```
. egen rs2=group(race sex), missing
. list race sex rs2
        race        sex        rs2
 1.                female          1
 2.    black     female          2
 3.    black       male          3
 4.    white                     4
 5.    white     female          5
 6.    white       male          6
 7.    white       male          6
```

crc28 Repeat command with specified arguments

The syntax of `repeat` is

`repeat` *list* [`, any noheader nostop`]`:` *stata-cmd*

`repeat` repeats *stata-cmd* substituting `@` in *stata-cmd* with each member of list.

Most Stata commands allow a varlist. For instance, `summarize` reports summary statistics for each variable listed after the command. Some Stata commands, on the other hand, do not have such flexible syntaxes. `ttest`, for instance, performs a test of the mean for only a single variable:

```
. summarize ch1 ch2 ch3
. ttest ch1=0
. ttest ch2=0
. ttest ch3=0
```

`repeat` allows commands such as `ttest` to be repeated:

```
. repeat ch1 ch2 ch3: ttest @=0
```

The `@` symbol in the Stata command that follows the colon is a placeholder: the command will be executed once with `@` changed to `ch1`, again with `@` changed to `ch2`, and a third time with `@` changed to `ch3`.

`repeat` allows standard Stata varlist syntax, so rather than typing

```
. repeat ch1 ch2 ch3: ttest @=0
```

you could type

```
. repeat ch*: ttest @=0
```

The `@` symbol may be repeated. For instance,

```
. repeat ch*: replace @=@+1
```

would add 1 to ch1, ch2, etc. (That is, the multiple @ symbols are given the same value in each replication.)

The any option

The any option changes the assumption that the specified list is a varlist of existing variables. Without the any option, you may not type

```
. repeat u1 u2 u3: gen @=uniform()
```

to generate three new variables containing uniform(0,1) random values. The variables u1, u2, and u3 do not (yet) exist and repeat will complain. You can type

```
. repeat u1 u2 u3, any: gen @=uniform()
```

Note the location of the any option: it is an option of repeat, not generate. It would be incorrect to type "repeat u1 u2 u3: gen @=uniform, any".

Once you specify any, the list can contain literally anything. For instance, mvdecode changes coded missing values to Stata's '.' missing value and does allow a varlist. Pretend that missing values were coded either -9 or -99:

```
. repeat -9 -99, any: mvdecode var*, mv(@)
```

The nostop option

If one of the replications produces an error, the replications stop at that point. For instance, ttest ch2=0 produces the error "insufficient observations" if ch2 does not contain at least two observations. Thus,

```
. repeat ch1 ch2 ch3: ttest @=0
```

might never produce the test for ch3.

```
. repeat ch1 ch2 ch3, nostop: ttest @=0
```

will produce all the tests that are possible. If the test for ch2 results in an error, that fact will be noted, but the replications will continue.

The noheader option

Before executing a replication, repeat shows the substituted command it is about to execute:

```
. repeat ch1 ch2: ttest @=0
-> ttest ch1=0
  (output omitted)
-> ttest ch2=0
  (output omitted)
```

The noheader option suppresses the display of the command. One blank line is inserted between the command's output instead.

Intention to rename repeat

This command would better be named for and, in the next release of Stata, it will be so renamed. Unfortunately, Stata's built-in format command was allowed to be overly abbreviated and the command-name for is taken as format.

After renaming, the word repeat will continue to be understood as a synonym for for.

crc29	Macintosh report

Some minor bugs with Macintosh Stata have become evident since its release.

1. Contrary to the assertions made in the manual, problems can occur when editing the file Stata.do or when creating other do- or ado-files with certain text editors, most notably, Microsoft Word. Most users never modify Stata.do, so this does not matter, but if you wish to modify the file, instructions are provided below.

Microsoft Word has two unfortunate behaviors. First, it preempts ownership of the file and puts its own icon on it so, if you edit `Stata.do` and then double click on it, rather then entering Stata, you will reinvoke Word. Second, if you are not careful, Word can change the format of the file to be unreadable by Stata.

To avoid these problems, begin by duplicating `Stata.do` and saving the copy, along with the original, in the Stata folder. (Alternatively, you may copy the original `Stata.do` to any folder you wish and then make the copy.) Rename the copy `new.do`. Now edit `new.do` using Word. When you save `new.do`, start by selecting *Save As* from the *File* pulldown menu; click on the *File Format* button; check the *Text Only* option. If you instead work through *Save* rather than *Save As*, you will never be presented with the *File Format* button and hence never get to check *Text Only*.

The file `new.do` will now have the Word icon which you must change back to the Stata icon. Bring Stata up by double clicking on the original `stata.do`. Give the command:

```
. touch new.do, like(stata.do)
```

Exit from Stata, close the folder and reopen it, and you will find that `new.do` now has the Stata icon. Verify that it works by double clicking on it; Stata should come up. If it does, after exiting Stata, you may rename the original `Stata.do` as `oldStata.do`, rename `new.do` as `Stata.do`, and make copies of the new `Stata.do` wherever you wish.

2. While most Macintosh users prefer to use the Macintosh-style of editing, the manual claims that Macintosh users may use the DOS/Unix-style Stata editing keys if they wish. That turns out not to be quite true if you do not have the extended keyboard (a keyboard with keys like *PgUp*, etc.) and, even so, there is one problem.

   To refresh your memory, the DOS/Unix-style editing keys include, for instance, pressing *PgUp* or *Ctrl-R* to get back the previous command; see [1] keyboard. There are two keys available for every editing function: a "natural" key (such as *PgUp*) and a control-key equivalent (such as *Ctrl-R*).

   The problem with the control-key equivalents is that some of them have been preempted by the Macintosh operating system. The following keys do work: *Ctrl-U*, *Ctrl-R*, *Ctrl-B*, *Ctrl-I*, *Ctrl-G*, *Ctrl-X*, *Ctrl-O*, and *Ctrl-N*. The following keys do not work because of operating-system conflicts: *Ctrl-H*, *Ctrl-D*, *Ctrl-P*, *Ctrl-L*, *Ctrl-K*, *Ctrl-W*, and *Ctrl-E*.

   If you have an extended keyboard, this does not matter because you will want to use the "natural" keys, anyway. If you do not have the extended keyboard, you will have to use the Macintosh-style of editing lines. Regardless of the keyboard you are using, there is one more problem: insert mode is on all the time and there is no way to turn it off. Even users of the extended keyboard will have to block-out text to be deleted and then press *Del*.

3. We have experienced some erratic behavior over networks and using network printers, all of which we are looking into and admit we do not ourselves understand. What makes these problems difficult (and interesting) is that they do not affect every user—most users are experiencing no difficulty whatever.

   Nevertheless, one user reports being able to start Stata on up to two computers attached to the network, but not on a third computer. We have received two reports of gphdot crashing after printing a graph on a network printer, but no reports of crashes after printing on non-network printers even when Stata (and gphdot) are installed on the network; on our own network, we experience no such problems.

   If you are experiencing problems with your network installation, please contact us. If you have any ideas or explanation of the two aforementioned problems, please contact us.

4. Macintosh filenames can contain spaces and sometimes those spaces are in unintentional places such as at the beginning or the end of the name. We have had a number of questions on the help line that turned out to be due to this "user error." If Stata tells you "file not found" to a file that you are certain exists, look for this problem. In Finder, click on the file to blacken the file name. If you see anything other than a thin, even space on each side of the file name, it has a "hidden" space. Rename such file to eliminate the leading and trailing space characters.

5. If you use utility programs to copy Stata `.dta` files from nonMacintosh sources, the resulting copy will not have the Stata icon attached to it and Stata's find-file menu will not list the file among the possible selections. `use` the file by typing out the file name and then resave it—this will attach the Stata icon and add it to the find-file list.

6. After making extensive changes to a folder and especially to the folder that contains the current folder ('`..`' in Stata's notation), the Macintosh operating system (and thus Stata) may become confused about the identity of the current folder. The best workaround is to go back one directory (type '`cd ..`') and then change back to the directory (type '`cd` *foldername*').

7. The Apple fast-access menu keys do not work in gphdot; you must use the pulldown menu with mouse.
8. Some users have reported that printed graphs are made at less than the maximum resolution on the printer. We are working on this.

dm11 Matching the Current Population Surveys

Finis Welch, Texas A&M Univ. and Unicon Research Corp., FAX 409-847-8757

The Current Population Survey (CPS) is perhaps the most commonly used source of annual (or, for the outgoing rotations, monthly) data on individual employment and earnings. The matched samples that exploit the Survey's quasi-panel structure are used less frequently but they, too, are a major resource. In working with the March-to-March matches, I have become increasingly aware that matching is easy if handled correctly and that the variables used for identifying the matches depend on the analysis. There are *no* general-purpose matched files and it is far safer to perform your own matches than to rely on decisions of others that nontrivially affect calculations.

This describes Stata programs for adjacent-year matching of records from the CPS. The program has options to return all records, both matched and unmatched; this not only helps in experimentation to choose variables for matching but in fully matched data, it facilitates analyses of biases associated with attrition and mismatches. The master program is `cpslink.cps` and all related utilities carry the suffix `.cps`. `cpslink.cps` is documented in `cpslink.doc`. There is a companion utility `makewide.wid` with related parts ending in the suffix `.wid`. `cpslink.cps` returns data in long form with first and second year records stacked; matches are linked with an identifying variable. `makewide.wid` places the first and second year records side-by-side. It is documented in `makewide.doc`. There are utilities, `char.ado`, `char.hlp`, `getspace.ado`, and `getspace.hlp`, to copy to an ado directory. I have matched the 1979/80 through the 1990/91 files. Under Unix, the program requires roughly 30 MB of RAM.

The CPS is the Government's official source of unemployment status. It was implemented in 1948 and has been used continuously since that time. It is a monthly survey of households (the well-known "household" data in contrast to the alternative "establishment" data sometimes used for wages and employment). There are currently 50–60,000 households in the Survey which collects information from one "knowledgeable" adult about all members of the household. The questionnaire has a basic part associated primarily with labor force participation and employment and a supplement which is normally repeated the same month each year. For example, June explores fertility histories and plans and October focuses on education.

The March Survey includes the Annual Demographic Supplement which is similar to the Decennial Census. Beginning with the 1968 Survey, the March Survey has been released annually for public use and has become one of the major vehicles for Social Science research. It is presumably our most comprehensive annual measure of wages, other sources of income, and employment.

The CPS sampling rotation is 4–8–4. There are interviews for four consecutive months, an eight month interruption, and a four-monthly-interview wrap-up. Because the second four interviews occur in the same calendar months as the first four, respondents can be linked across years. With no attrition, half of those interviewed this March would have been interviewed last March; the other half will be interviewed next March. Each record in the public-use data carries month-in-sample, household, family (within household), and individual identifiers. Month-in-sample 1–4 identifies those in the first year of the survey and month-in-sample 5–8 presumably refers to the second year. Yet, for a variety of reasons the Census Bureau maintains approximate balance in numbers of households by month-in-sample. To compensate for attrition, there are month-in-sample 2–4 observations in one survey that do not appear in the previous month's survey. Similarly there are month-in-sample 5–8 records that cannot be matched in the previous year's survey.

Matching proceeds in three stages. In the first, month-in-sample is used to form datasets where matches may exist. For adjacent-year matching, the month-in-sample 1 population in one year is matched with the month-in-sample 5 population from the same monthly survey of the succeeding year. This stage is left to the user. The second stage involves matching households and the third stage consists of matching individuals within matched households. I have not explored alternatives to the household identifier that the Census Bureau provides for linking households. For individuals within matched households, it is clear that the person identifier, the line number, carries important information but it carries a degree of error as well. One can do much better but what one does depends on the purpose for which the matches are intended. As an example, consider the relation to the householder. It obviously carries useful information for matching; (records matched on line numbers showing one year that a person is a child of the householder and the next year the same person is an unrelated individual are presumably questionable), but it would probably be a mistake to use the relationship variable in matching if the objective is to study the dynamics of household or family composition.

`cpslink.cps` matches records in a series of steps and the step in which the match occurs becomes part of the record. As an example, in exploratory work I first matched individuals in matched households on line number, race, gender, and age (Step 1). In this step, I permitted up to a one-year disagreement in age, i.e., during the year between surveys the person could have aged 0, 1, or 2 years. Those unmatched after the first step were then matched on race, gender, and age without reference to line number (Step 2). Finally those unmatched after the first two steps were matched on line number alone. After inspecting the data, I discarded the third step matches but kept those from the first and second step.

Each step has two parts, an initial pass and a tie breaker. In Step 1 (above) the data within matched households were first sorted into cells distinguished by line number, race, gender, and age. Those cells that contain exactly one first- and one second-year record are uniquely matched and the records are so labeled. Cells with multiple potential matches (more than one record from one year with one or more records from the other) are submitted to a finer partition via a secondary list of variables. For example, in Step 1 described above, I used education, major activity the week before the interview, and a modified version of relation to householder for breaking ties. In the tie breaking round, the data are partitioned in exactly the same way as in the associated step. The distinction is that the partition is finer for the tie breaker. If unique matches are found in the finer partition, they are recorded. If not, one of the variables used for the partition is dropped, and the data are rearranged into a new set of less finely specified cells. The process continues, recording matches along the way, until all ties are broken or until all of the supplementary tie breaking variables are dropped. If multiple matches remain after the tie breaker, they are broken randomly.

Matched CPS samples are becoming increasingly important in the social sciences. Their sample size swamps all alternative sources of panel data and the rotating design guarantees continual refreshing. The alternative sources of panel data, the Panel Study of Income Dynamics (PSID), the National Longitudinal Surveys, the Retirement History Survey, etc., attempt to follow the same people for a long time, e.g., PSID started in 1968 and is still running. To my knowledge there have been very few studies of biases that may accumulate under the steady erosion of the sample. (See Becketti, Gould, Lillard, and Welch 1988 for an analysis of attrition biases in the PSID during its first 17 years). One naturally wonders what distinguishes the survey survivors from those that are lost.

After a quick comparison of matched and unmatched records in the CPS, I am excited about the prospects for expanding work on attrition biases using data such as these.

In closing this brief note, I will summarize some of the things I have discovered. Before doing so I relate a story from one of the first users of the programs described here—two graduate students at the University of Chicago. In examining matched records for women who became married (i.e., who changed from single in one survey to married the next) they discovered that 75% had been married previously. This is not because most brides are repeat offenders; rather, it refers to the nature of the match. A necessary condition for a match is that the residence is unchanged for two successive years. Evidently, first time brides are more likely to move at the time of their marriage!

The CPS will not match households across adjacent years if the residents move or if the interviewers are simply unable to establish contact (refusals, no one home etc.). In addition, households are sometimes not matched by design, for example there is overrepresentation of Hispanic households that are not interviewed in successive years. As noted, the Census Bureau tries to balance the number of households in each month-in-sample group. If the total sample shrinks, some of the ongoing households will be dropped.

In the eleven matches I attempted, the proportion of first year households that could be matched with second year records ranged from .695 (the 1981/82 match when the budgeted sample was reduced) to .765 (the 1989/90 match). Within matched households the maximum potential match rate based on membership only—the potential number of matches is limited to the minimum of the first and second year household membership—is amazingly constant, as a proportion of first year membership it is higher than .945 and lower than .950!

In the first step described earlier, where matches were strictly defined to require agreement on line number, race, sex, and age within one year, the successful match rate for first year individuals in matched households ranged between 88.0 and 89.7 percent. These rates average roughly 94 percent of the potential based on membership alone so the most that is possible with relaxing standards to allow for coding or reporting errors in matching criteria would be to increase the match rates by 6 or 7 percentage points.

It is important that match rates are not independent of individual characteristics and the behavior of those who are matched is not representative of those who are not. Figure 1 describes average rates of failure in matching by single year of age for first year individuals. It may be surprising that the highest match rates (the lowest failure rates) are encountered in the age 70–75 range where 18–19 percent of the individuals cannot be matched across adjacent survey years. More than two-thirds of the losses in this range are from failures to match households rather from match failures of individuals within the matched households. The maximum losses are for ages 22 and 23 where failure rates are high both in matching households and in matching individuals within households.

Tables 1 and 2 use records from the 1980–1984 and the 1987–1990 March Surveys. The rule for including surveys is that all observations must have been classified by match status. When these calculations were made the 1991 Survey was the most recent. It is not used in Tables 1 and 2 because the month-in-sample 1–4 observations require the 1992 Survey for classification of match status. Similarly, 1985 and 1986 could not be matched (household ID's were scrambled between these surveys) so they are excluded. Each observation in the remaining surveys is first classified by year of potential match; month-in-sample 1–4 observations are in the first year and month-in-sample 5–8 observations are in the second year. They are further classified as matched, unmatched in matched households, or as in unmatched households so that each observation falls into one of six classes. By comparing behavior or attribute differences among the classes, we can gain insight into the types of problems that will emerge in using balanced panels for inferences about larger groups. Table 1 reports percentages employed full time by match status, age, race and gender. It may be reassuring to note that among matched records the first and second-year employment rates are similar. There are, however, marginally higher rates for the first-year. Perhaps those who transit from working full time to not working full time are less likely to change residence (and therefore be lost to the match) than those who transit from not working full time to working full time. Of course the differences in employment rates between the matched and unmatched data swamp the differences within the matched data. For men, except for the youngest group, employment rates for unmatched individuals in matched households are much lower than the employment rates of the matched individuals. The pattern is reversed for women. Within matched households, the unmatched individuals have higher employment rates than those who are matched. With full-time employment as the metric, those in unmatched households lie between the matched and unmatched but they are notably closer to the unmatched in the matched households.

Table 2 reports the average composition of the nine surveys. In these tabulations, the number of second survey matched records in each survey was scaled to equal the number of first survey matches (within age, race and gender), and the number of second survey unmatched records was scaled by the same multiple. Before averaging over years, the number of records in each annual survey was scaled so that all surveys would receive equal weight. The numbers in this table are percentages of first survey records. The first thing to note is that the proportion of first year records that are in unmatched households is similar for blacks and whites. The next thing is that household composition dynamics are very different. Notice for whites that percentages of unmatched individuals in matched households are higher for the first survey year. Evidently the dominant dynamic is that members leave the household. The pattern is the same for young blacks but from the mid-30s on, arrivals exceed departures.

Panel data offer exciting research possibilities. I have used the March-March matches to study labor force transitions of older workers and to study coding errors or inconsistencies in education, occupation and industry. The Census Bureau changed occupational coding structures dramatically between the 1970 and 1980 Census. The 1973–1982 March CPS uses the 1970 codes and the 1983–1991 CPS uses the 1980 codes. The Census Bureau codes occupation in the March Surveys for the week preceding the interview and separately for the main job in the previous year. It also codes the number of jobs in the previous year. By restricting matched observations to individuals who reported only one job in the year preceding the second year survey, one presumably has a sample where the current job in the first survey is the same as the main job last year in the second survey. I have used the 1982–1983 matches (where the current job in the 1982 survey is coded with 1970 occupation codes and the occupation last year in the 1983 survey is listed with 1980 codes) to develop empirical crosswalks between the two coding structures. Perhaps the matching programs will be useful in ongoing research.

(*Continued on next page*)

Figure and Tables

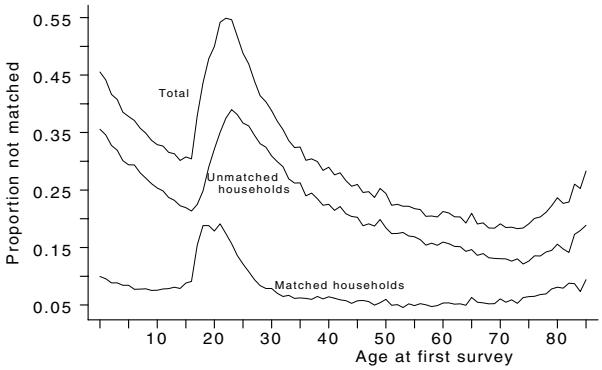

Figure 1. Match Failure by Age

Table 1. Percentages employed full time by year in survey and match status

	First Year in Sample			Second Year in Sample		
	Matched	—Unmatched—		Matched	—Unmatched—	
age		Matched Household	Unmatched Household		Matched Household	Unmatched Household
White men						
15–23	35.7	40.3	42.5	34.1	40.4	44.3
24–30	88.2	81.4	84.8	87.4	81.7	85.6
31–37	92.4	85.9	87.7	92.3	85.3	88.6
38–50	92.4	85.7	88.2	91.9	84.7	88.1
51–63	74.0	62.4	69.2	74.3	65.0	68.7
64–75	14.2	12.3	14.6	13.7	12.4	12.8
White women						
15–23	26.8	32.0	31.6	26.2	32.4	32.5
24–30	55.7	63.7	57.2	54.9	60.9	57.6
31–37	51.0	57.5	55.0	50.9	55.5	55.0
38–50	54.4	55.7	54.5	53.9	55.6	54.9
51–63	38.2	37.4	37.4	37.8	39.5	36.4
64–75	5.9	6.2	6.8	5.6	6.9	5.8
Black men						
15–23	31.0	34.0	33.3	30.0	34.2	36.1
24–30	79.2	73.1	73.5	76.7	73.1	75.9
31–37	84.9	70.8	79.9	83.3	76.2	81.7
38–50	83.4	72.9	79.5	82.8	72.6	77.8
51–63	62.9	51.0	55.3	60.9	55.9	59.5
64–75	12.5	10.8	10.9	11.5	7.0	13.7
Black women						
15–23	23.1	25.0	26.8	20.8	25.8	24.8
24–30	59.9	61.3	61.9	59.6	59.7	59.0
31–37	67.0	65.3	63.7	65.9	65.4	63.0
38–50	63.4	60.2	59.8	62.3	62.8	61.5
51–63	40.9	30.6	39.5	40.6	35.2	37.8
64–75	5.4	9.5	5.5	4.8	8.5	7.8

Table 2. Distribution of year in sample and match status by age, gender, and race (Percentages of all individuals in first year-in sample)

	First Year in Sample			Second Year in Sample		
	Matched	—Unmatched—		Matched	—Unmatched—	
age		Matched Household	Unmatched Household		Matched Household	Unmatched Household
White men						
15–23	55.6	15.6	28.8	55.6	8.6	29.1
24–30	54.3	10.5	35.1	54.3	8.8	41.4
31–37	66.8	6.4	26.8	66.8	5.5	29.3
38–50	73.6	5.6	20.7	73.6	4.8	21.8
51–63	78.9	4.7	16.3	78.9	3.7	16.1
64–75	81.0	5.7	13.3	81.0	3.2	13.4
White women						
15–23	53.8	15.4	30.7	53.8	9.0	32.3
24–30	57.1	9.1	33.8	57.1	8.1	39.4
31–37	68.4	5.7	25.9	68.4	5.3	27.3
38–50	74.2	5.4	20.3	74.2	4.9	20.4
51–63	79.3	4.8	15.9	79.3	4.6	15.6
64–75	81.8	5.2	13.1	81.8	4.0	13.5
Black men						
15–23	57.0	16.9	26.1	57.0	9.4	24.6
24–30	50.3	14.4	35.3	50.3	11.7	38.7
31–37	61.6	10.0	28.4	61.6	9.5	29.1
38–50	69.5	9.1	21.4	69.1	11.0	23.6
51–63	75.7	7.8	16.5	76.9	13.1	18.3
64–75	79.5	8.8	11.7	80.1	18.5	17.4
Black women						
15–23	58.5	14.6	26.8	58.5	9.6	29.1
24–30	57.7	9.8	32.6	57.7	8.5	36.4
31–37	66.3	6.8	26.9	66.5	8.1	27.7
38–50	73.5	6.3	20.2	74.7	12.9	19.7
51–63	78.6	6.0	15.4	78.9	14.5	15.8
64–75	81.2	6.6	12.2	80.4	18.9	15.3

The first and second year numbers are percentages of first year membership. The three first-year columns sum to 100 (within rounding error). The second year columns are normed so that the percent matched is the same as in the first year.

References

Becketti, S., W. Gould, L. Lillard, and F. Welch. 1988. The Panel Study of Income Dynamics after fourteen years: An evaluation. *The Journal of Human Resources* 6(4): 472–492.

dm12 | Selecting claims from medical claims data bases

Robert J. Vaughn, MPH, Clinical Projects Manager, Utah Peer Review Organization, FAX 801-487-2296

Hospitals use a standard billing form—the UB82—for inpatient claims submission. Data bases composed of claims data are available to hospitals and other organizations such as Medicare Peer Review Organizations (PROs), The National Center for Health Statistics, state data organizations, and insurance companies. These data are also available to researchers in the Health Care Financing Administration (HCFA) MEDPAR file. Included in the data are up to ten procedure codes and ten diagnosis codes per case. Procedures and diagnoses are coded using the International Classification of Diseases (ICD-9-CM). The number of codes available to researchers varies by data processor and by year of the claim. For many years, only three procedure and six diagnosis codes are reported.

Selecting claims based on procedure or diagnosis can be tedious. Two Stata ado-files—`anyproc` and `anydx`—make this easier. These new commands assume

1. The data procedure codes are stored in variables named `proc1`, `proc2`, ... There may be up to ten such variables (ending with `proc10`), although all variables need not exist. For instance, in a data set where only three procedure codes are available, only `proc1`, `proc2`, and `proc3` must exist. (If `proc4` does not exist, `proc5`, `proc6`, ..., `proc10` are also assumed not to exist.)
2. A similar assumption is made for the diagnosis codes—they are names `dx1`, `dx2`, ..., `dx10`, but there may be fewer than ten.

3. The `proc1`, ..., `proc10` and `dx1`, ..., `dx10` variables are stored as string variables because some codes have alphabetic prefixes (such as `E` and `V` used in ICD-9). The codes may or may not have leading and trailing blanks—it does not matter.

The syntax of `anydx` and `anyproc` is

$$\{\ \texttt{anydx}\ |\ \texttt{anyproc}\ \}\ \textit{newvar}\ \textit{code}\ [\textit{code}\ [\textit{code}\ [\ldots]]]$$

Both commands create (`byte` variable) *newvar* and set it equal to 1 where the diagnosis or procedure codes are any of the codes listed and 0 otherwise.

Say I am interested in analyzing data from claims of patients who had a diagnosis of spinal stenosis (ICD-9 codes 721.42, 721.91, 724.00, 724.02, and 724.09) and laminectomy as the procedure (ICD-9 codes 03.0, 03.02, 03.6, and 03.09.) Assuming my data has the variables `dx1`, `dx2`, ... and `proc1`, `proc2`, ..., I can type

```
. anydx spinal 721.42 721.91 724.00 724.02 724.09
. anyproc lamin 03.0 03.02 03.6 03.09
. keep if spinal & lamin
```

The way you enter the codes may differ since different sites store the codes differently (some omitting decimal points, etc.), but that does not matter. Enter the codes as you think of them. You need not worry about blanks at the beginning or end of codes—regardless of how your data is stored, they will be ignored, so typing, for instance, `036` will find `"036"`, `" 036"`, or `"036 "`.

I would like to thank William Gould of CRC for his assistance in writing the `anydx` and `adoproc` programs.

gr3.1 Crude 3-dimensional graphics revisited

William Gould, CRC, FAX 310-393-7551

The front cover of *Statistics with Stata 3* (Hamilton 1993) shows a pretty 3-dimensional graph. We have already received the question, "Can Stata draw 3-dimensional graphs?" The answer is yes—sort of. I provided the commands last year (Gould 1992). The results are not up to our usual standards (you cannot, for example, label the axes), but they otherwise work. Figures 1 and 2 were drawn using them.

These routines were written for Stata 2.1 and require a small modification (or typing '`version 2.1`') to work. They are updated to work with Stata 3.0 on the STB-12 diskette. See the original insert for instructions on using them.

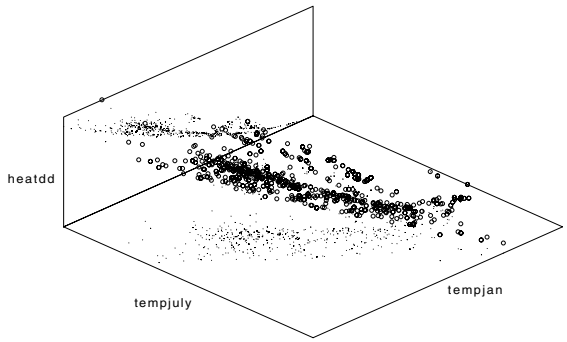

Figure 1

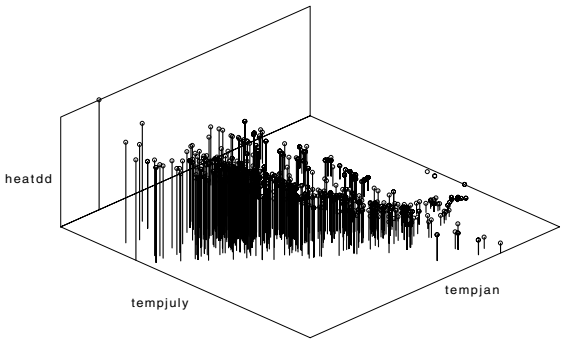

Figure 2

References

Gould, W. 1992. gr3: Crude 3-dimensional graphics. In *The Stata Technical Bulletin Reprints*, vol. 1, ed. J. Hilbe, 35–38. Santa Monica, CA: Computing Resource Center.

Hamilton, L. C. 1993. *Statistics with Stata 3.* Belmont, CA: Duxbury Press.

gr12	Graphs of means and medians by categorical variables

William Gould, CRC, FAX 310-393-7551

The syntax of `grmeanby` is

`grmeanby` *varlist* [*weight*] [`if` *exp*] [`in` *range*], `summarize(`*varname*`)` [`median` *graph-options*]

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

`grmeanby` graphs the (optionally weighted) means or medians of *varname* according to the values of the variables in *varlist*. The variables in *varlist* may be string or numerical and, if numerical, may be labeled.

Options

`summarize(`*varname*`)` is not optional; it specifies the name of the variable whose mean or median is to be graphed.

`median` is optional; it specifies the graph is to be of medians, not means.

graph-options are any of the options of `graph`'s `twoway` style.

Remarks

The idea of graphing means of categorical variables was shown in Chambers and Hastie (1992, 3). Since this was shown in the context of an S function for making such graphs, it doubtlessly has roots going back farther than that. `grmeanby` is, in any case, another implementation of what I will assume is their idea. Figure 1 was drawn by typing

```
. grmeanby foreign rep77 rep78 make, sum(log) ylab
```

after loading the automobile data. Had I wanted a graph of medians rather than means, I would have typed

```
. grmeanby foreign rep77 rep78 make, sum(log) ylab median
```

Figure 2 was drawn by typing

```
. grmeanby race hhrel imputed marstat reltohd ftpt, sum(wage) ylab
```

and is based on 29,650-observation subsample of men with labor force experience (and other restrictions) from the 1991 Current Population Survey (CPS). some labor force experience.

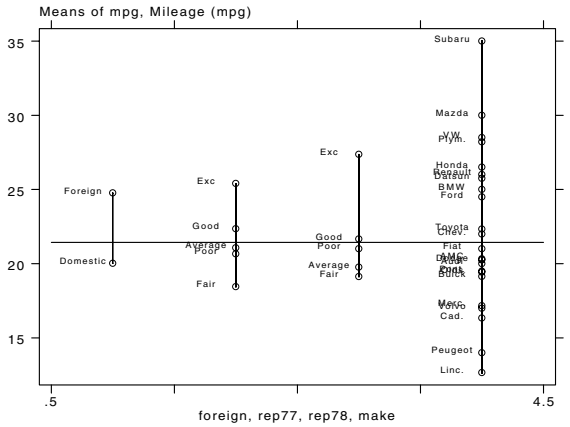

Figure 1

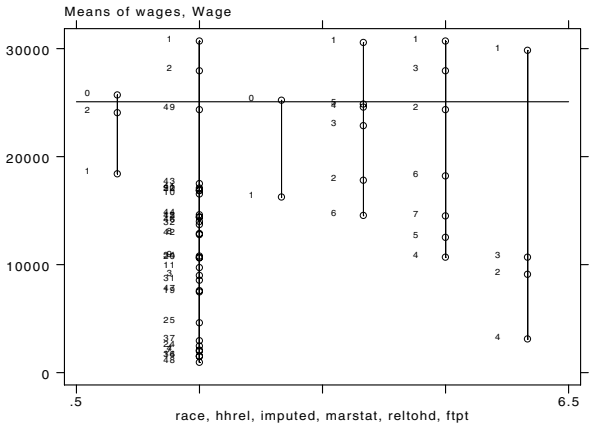

Figure 2

References

Chambers, J. M. and T. J. Hastie. 1992. *Statistical Models in S*. Pacific Grove, CA: Wadsworth & Brooks/Cole.

sbe2.1	Bailey–Makeham survival models

William Rogers, CRC, FAX: 310-393-7551

In Rogers (1991), I published a stand-alone adjunct to Stata for estimating the Bailey–Makeham survival model. This included `BAILEY.EXE` for estimating the Bailey–Makeham model; `PREDICT.EXE` for predicting coefficient values for members of a data set; and `SURFACE.EXE` for analyzing the likelihood surface in the vicinity of the solution. The original insert also provided all the source code so that these routines could be compiled on other than DOS computers. I also included a set of ado-files for integrating these three stand-alone programs into Stata, making it appear that Bailey–Makeham estimation was part of Stata.

I have updated the originally published ado-files to bring them into line with current Stata programming practices and to correct some minor bugs. The STB diskette includes the updated ado-files and unchanged copies of the original `.EXE` files. (The source code is not contained on this STB diskette but is on the original; the original documentation is republished on this diskette.)

References

Rogers, W. H. 1991. Bailey–Makeham survival model. *Stata Technical Bulletin* 2: 11–14. (Reprinted in *Stata Technical Bulletin Reprints*, vol. 1: 64–73.)

sbe2.2	Faster hpredict.exe

Henry Krakauer, U.S. Public Health Service

`PREDICT.EXE` (Rogers 1991) was compiled to run on a DOS computer. I have recompiled the program using the Metaware compiler and Pharlap DOS-extender environment to produce `HPREDICT.EXE`, a program identical to `PREDICT.EXE` but much faster. `HPREDICT.EXE` can only be run on 386/486 computers with a coprocessor and cannot be run from inside Intercooled Stata, but if you are attempting the Bailey–Makeham model in stand-alone mode on large problems, you may find the improved performance useful.

References

Rogers, W. H. 1991. Bailey–Makeham survival model. *Stata Technical Bulletin* 2: 11–14. (Reprinted in *Stata Technical Bulletin Reprints*, vol. 1: 64–73.)

sg1.5	Standard nonlinear curve fits update

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011)-44-81-740 3119
EMAIL proyston@rpms.ac.uk

The ado-files I provided in *sg1.4* of STB-11 (Royston 1993) for performing automated fits of seven common, nonlinear regression functions (three for exponential regression with one asymptote, two for the logistic function—symmetric sigmoid shape not to be confused with logistic regression—and two for the Gompertz function) contained an error. All reported results were correct, but when the user specified weights (`aweights` or `fweights`), the ado-files terminated with a syntax error rather than producing the desired results. This is fixed.

The updates on the STB-12 diskette can be installed on top of the previous code or can be installed for the first time without having installed the previous insert. On-line help is available, type '`help nlfcns`'.

References

Royston, P. 1993. sg1.4: Standard nonlinear curve fits. *Stata Technical Bulletin* 11: 17.

sg16.1 Generalized linear models using power links

Joseph Hilbe, Dept. of Sociology, Arizona State Univ., INTERNET atjmh@asuvm.inre.asu.edu

As presently structured, the `glm` command (Hilbe 1993) has nearly reached the system size limitations for an ado program. In order to provide GLM power links, it has been necessary for me to create a separate program. The new ado command called `glmp` allows modeling the full range of acceptable power links for the Gaussian, Poisson, gamma, and inverse Gaussian distributions. The syntax of `glmp` is

`glmp` *depvar varlist* [*fweight*] [`if` *exp*] [`in` *range*], `f(`{`ber`|`bin`|`gau`|`poi`|`gam`|`invg`}`)`
[`s r p(`*power*`) sc(`{`d`|`c`|`n`}`) ex(`*varname*`) o(`*varname*`) level(`#`) it(`#`) lt(`#`) eform`]

where `p(`*power*`)` is required for any power other than the default value of 1. Typically used powers are the square, `p(2)`; square root, `p(.5)`; and inverse, `p(-1)`. `glmp` can also model other `glm` links as well; identity links are modeled using `p(1)` and canonical links are as follows: Gaussian, `p(1)`; Poisson, `p(0)`; gamma, `p(-1)`; and inverse Gaussian, `p(-2)`. Note that the Poisson distribution has a natural log canonical link. In fact, all log links can be modeled with the `p(0)` option. Hence, to model a simple default lognormal regression, use the following command:

```
. glmp depvar varlist, f(gau) p(0)
```

Bernoulli and binomial distributions have a $\log(\mu/(1-\mu))$ canonical link which cannot be modeled using powers. However, these distributions may still be modeled in a noncanonical manner similar to probit and complementary loglog using powers. Recall, ill-defined models result in nonconvergence or in zero or missing coefficients in all GLM modeling situations.

I have added another feature to the `glmp` command line as well. The `sc()` option allows the user to select the scale upon which the standard errors are to be adjusted. The default is to scale by the deviance (`sc(d)`) rather than chi2 (`sc(c)`). `sc(n)` is to be selected if one desires no scaling. In general, you should use the `sc(n)` option with Poisson and the default deviance with the Gaussian, gamma, and inverse Gaussian distributions. However, if there is evidence of substantial dispersion then you should scale the Poisson standard errors by either the deviance or chi2. Both dispersion values are presented on the screen. A better fitting model is one that has low deviance and a pattern of significant explanatory variables.

In my own work I have found that `glmp` is faster and more versatile for modeling non-binomial distributions than `glm`. The full range of residuals has also been made available for `glmp` models, with an addition of residuals for Gaussian distribution power links.

The following example uses a square power link on variables from the automobile data that comes with Stata:

```
.  glmp mpg weight length, f(invg) p(2) sc(d)
Iter 1 : Dev =    0.1117
Iter 2 : Dev =    0.0765
Iter 3 : Dev =    0.0764
Iter 4 : Dev =    0.0764
                                                No obs.    =        74
Chi2       =  .0819838                          Deviance   =  .0764327
Prob>chi2  =         1                          Prob>chi2  =         1
Dispersion =  .0011547                          Dispersion =  .0010765
Inverse Gaussian: Power = 2
------------------------------------------------------------------------------
mpg      |      Coef.   Std.Err.        t     P>|t|    [95% Conf. Interval]
-------------------------------------------------------------------------------
weight   |  -.0933343     .04528     -2.061   0.043   -.1820815   -.0045872
length   |  -3.785392   1.769495     -2.139   0.036   -7.253539   -.3172446
_cons    |   1452.548   213.9344      6.790   0.000    1033.245    1871.852
------------------------------------------------------------------------------
Standard errors adjusted by sqrt(dispersion)
```

In addition, the previously published `glm` program (Hilbe 1993) has two minor errors. One effected the dispersion value when there were missing values in explanatory variables; the other concerned the reweighting of the inverse Gaussian log link. There also was a small change I made to the `_glmresd` internal command. The updated versions are automatically installed when you install the new `glmp` command.

References

Hilbe, J. 1993. Generalized linear models. *Stata Technical Bulletin* 11: 20–28.

sqv7	Cusum plots and tests for binary variables

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011)-44-81-740 3119
EMAIL proyston@rpms.ac.uk

The accompanying ado-file `cusumb.ado` facilitates plotting of a binary (0/1) variable against a continuous variable. The idea is described in the context of logistic regression analysis by Royston (1992). Here, the principle will be motivated by way of an example.

The syntax of `cusumb` is

`cumsumb` *yvar xvar* [`if` *exp*] [`in` *range*] [`, yfit(`*fitvar*`) nograph nocalc gen(`*newvar*`)` *graph_options*]

`cusumb` produces a plot of the cumulative sum (cusum) of a binary (0/1) variable *yvar* against a (usually) continuous variable *xvar*. The cusum is the partial sum of proportion-of-ones-in-the-sample minus *yvar*. `cusumb` sorts the data in the order of *xvar*. Tied values of *xvar* are broken at random, but always in identical fashion for a given data set.

Interpretation

A U- or inverted U-shaped cusum indicates respectively a negative or a positive trend of *yvar* with *xvar*. A sinusoidal shape is evidence of a non-monotonic (for example, quadratic) trend of *yvar* with *xvar*. `cusumb` displays the maximum absolute cusum for monotonic and non-monotonic trends of *yvar* on *xvar*. These are nonparametric tests of departure from randomness of *yvar* with respect to *xvar*. Approximate values for the tests are given.

Options

`yfit(`*fitvar*`)` calculates a cusum against *fitvar*, that is the partial sums of the 'residuals' (*fitvar* minus *yvar*). Typically, *fitvar* is the predicted probability of a one obtained from a logistic regression analysis.

`nograph` suppresses the plot.

`nocalc` suppresses calculation of the cusum test statistics.

`gen(`*newvar*`)` saves the cusum in *newvar*.

graph_options are any of the standard `graph`'s `twoway` options.

Example

For the automobile data set `auto.dta`, we might wish to investigate the relationship between `foreign` (0 = domestic, 1 = foreign) and car weight as follows:

```
. use auto
. cusumb foreign weight
(graph appears, see figure 1)
Variable |   Obs    Pr(1)   CusumL    zL    Pr>zL   CusumQ    zQ    Pr>zQ
---------+-------------------------------------------------------------------
 foreign |    74   0.2973    10.30   3.963  0.000     2.92   0.064  0.475
```

The resulting plot, which is U-shaped, suggests a negative monotonic relationship. The trend is confirmed by a highly significant linear cusum statistic, labelled CusumL in the output above.

Some 29.73% of the cars are foreign (coded 1). The proportion of foreign cars diminishes with increasing weight. Stated crudely, the domestic cars are heavier than the foreign ones. We could have discovered that by typing `tabulate foreign, summarize(weight)` but such an approach does not give the full picture of the relationship. The quadratic cusum (cusumQ) is not significant, so we do not suspect any tendency for the very heavy cars to be foreign rather than domestic. A slightly enhanced version of the plot shows the preponderance of domestic (coded 0) cars at the heavy end of the weight axis:

```
. label drop foreign
. cusumb foreign weight, xlabel ylabel connect(.) symbol([foreign])
(see figure 2)
```

The example is of course artificial as we would not really try to model the probability of a car being foreign given its weight!

Figures

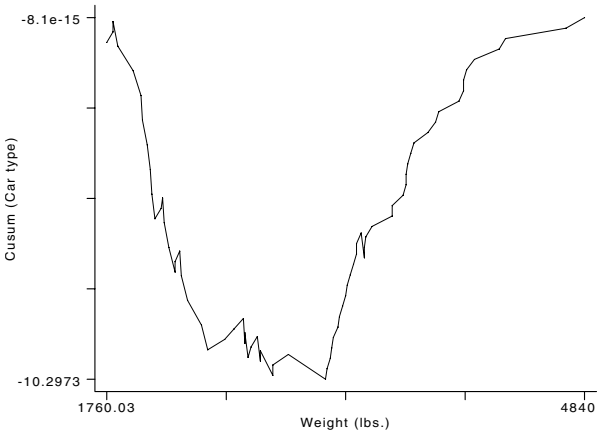

Figure 1

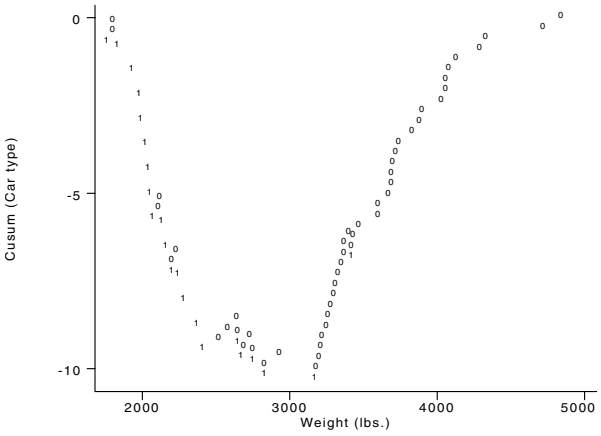

Figure 2

References

Royston, P. 1992. The use of cusums and other techniques in modelling continuous covariates in logistic regression. *Statistics in Medicine* 11: 1115–1129.

tt6	Courseware

William Gould and William Rogers, CRC, FAX 310-393-7551

Courseware is a general term for software intended to assist in teaching. Herewith, we offer a general outline—and working code—for a courseware system in Stata. The organizing component is called cw—for courseware—and provides a flexible system for defining "books," "chapters" within books, and "exercises" within chapters. The system automatically tracks the available courseware resources so that adding new courseware to the system requires only copying the defining files. All selection menus are automatically updated. A subcomponent of the system called qa—for question and answer—provides commands for automatically constructing question-and-answer dialogs.

This all is rather vague, so let's begin by seeing the system in action. Here is an actual log:

```
. cw

Keyword    Author and title
-------------------------------------------------------------------------------
Help       Instruction to using courseware
Iman       R. L. Iman, A Data Based Approach to Understanding Statistics
Enter a keyword or end -> . iman


Chapter    Contents
-------------------------------------------------------------------------------
1          Data collection:  The role of sampling in statistics
7          Hypothesis testing
Enter a chapter number or end -> . 7


Exercise   Description
-------------------------------------------------------------------------------
1          Identifying the null hypotheses
2          Psychologist´s test statistic
3          Bank passbook interest (Iman problem 7.28)
Enter an exercise number or end -> . 2
```

```
------------------------------------------------------------------------------
   7.4 A psychologist believes that children prefer the color red over the
       color blue.  He does an experiment by giving children a red and a blue
       fire truck and seeing which truck the child selects.  What is the null
       hypothesis?
       1.  The probability of choosing a red truck is <= 50 percent
       2.  The probability of choosing a red truck is 50%
                                                       answer -> . 2
------------------------------------------------------------------------------
   7.5 Your null hypothesis is an OK hypothesis to test, but it is not the
       null hypothesis presented in Iman. The difference between these two
       answers lies in what action to take if the result of the experiment
       is a surprise, and the children select a preponderance of blue
       trucks.  Given the null hypothesis that the probability of choosing a
       red truck is <= 50 percent, how would the psychologist have to react?
       1.  Accept the null hypothesis.
       2.  Conclude that children prefer blue trucks.
                                                       answer -> . 2
--->   Recall that the null hypothesis is that the probability of choosing
--->   a red truck is <= 50%.
   7.5 Your null hypothesis is an OK hypothesis to test, but it is not the
       null hypothesis presented in Iman. The difference between these two
       answers lies in what action to take if the result of the experiment
       is a surprise, and the children select a preponderance of blue
       trucks.  Given the null hypothesis that the probability of choosing a
       red truck is <= 50 percent, how would the psychologist have to react?
       1.  Accept the null hypothesis.
       2.  Conclude that children prefer blue trucks.
                                                       answer -> . 1
------------------------------------------------------------------------------
   7.6 Assume that the psychologist´s data are available in Stata:
. tabulate pickred
    pickred|      Freq.     Percent        Cum.
-----------+-----------------------------------
         0 |         44       44.00       44.00
         1 |         56       56.00      100.00
-----------+-----------------------------------
     Total |        100      100.00
                              How many children picked red? -> . end


Exercise   Description
------------------------------------------------------------------------------
1          Identifying the null hypotheses
2          Psychologist´s test statistic
3          Bank passbook interest (Iman problem 7.28)
Enter an exercise number or end -> . end


Chapter    Contents
------------------------------------------------------------------------------
1          Data collection:  The role of sampling in statistics
7          Hypothesis testing
Enter a chapter number or end -> . end


Keyword    Author and title
------------------------------------------------------------------------------
Help       Instruction on using courseware
Iman       R. L. Iman, A Data Based Approach to Understanding Statistics
Enter a keyword or end -> . end

Goodbye.

. _
```

All of the above was created by a surprisingly small amount of code (ignoring the code in the courseware system itself). For instance, the courseware code for the dialog in question 7.5, in its entirety, is

```
qa_begin 7.5
qa_text "Your null hypothesis is an OK hypothesis to test, but it is not the"
qa_text "null hypothesis presented in Iman. The difference between these two"
qa_text "answers lies in what action to take if the result of the experiment"
qa_text "is a surprise, and the children select a preponderance of blue"
qa_text "trucks.  Given the null hypothesis that the probability of choosing a"
qa_text "red truck is <= 50 percent, how would the psychologist have to react?"
qa_ans CORRECT CORRECT "Accept the null hypothesis."
qa_ANS E1 INFORM "Conclude that children prefer blue trucks."
qa_err  E1 "Recall that the null hypothesis is that the probability of choosing"
qa_err2 E1 "red truck is <= 50%."
qa_ask
```

That is the idea: to make it easy to write courseware. We do not have much courseware installed in our system yet.

The cw system

The syntax of `cw`, the command to invoke the courseware system, is simply that: `cw`; it takes no arguments or options. `cw` itself provides the on-line help for the courseware users (whom we will call students) and the amount of that documentation is intentionally minimal. `cw` can be used with little to no understanding of Stata and no training in courseware basics.

We will use `cw` for the courseware-invocation command itself and cw (in roman type) for the courseware system. We will reserve the term courseware for programs written in cw.

cw is first an organizing scheme. cw is installed somewhere, and for the purposes of explanation, we will assume that somewhere is `c:\cw`. (It does not have to be installed there nor does it even have to be installed on a DOS computer. Courseware written in cw will work on PCs, Macintoshes, and even Unix systems.) Wherever cw is installed (which `cw` will figure out for itself), the following structure is assumed:

`c:\cw`	assumed courseware directory, can vary
`c:\cw\book#`	directory for a courseware book
`c:\cw\book#\ch#`	directory for a courseware chapter
`c:\cw\book#\ch#\ex#`	directory for a courseware exercise

(For example, if cw were installed in `d:\sys\cw`, the directory for the books would be `d:\sys\cw\book#`.)

To add new courseware to the system, a new `book#` directory is created. The # you choose does not matter, although most sites would probably allocate them sequentially. That directory must contain the file `book.do` defining a keyword, author, and book title. For instance, we might create `book3`:

```
-------------------------------- book3\book.do ---------------------------------
mac def S_1 "keyword"
mac def S_2 "author(s)"
mac def S_3 "title"
---------------------------------- end of file -----------------------------------
```

If the book has no identified authors, define global macro `S_2` as nothing (`mac def S_2 ""`). When `cw` is invoked, the student is shown a list of keywords and books and asked to choose one:

```
. cw
Keyword    Author and title
-------------------------------------------------------------------------------
Help       Instruction on using courseware
Iman       R. L. Iman, A Data Based Approach to Understanding Statistics
Enter a keyword or end -> . _
```

In creating our system, we happened to allocate `book1` to help and `book2` to Iman, but it does not matter.

The next step is to define one or more chapters within the `book#` directory. Chapter 1 is placed in `book#\ch1`, chapter 2 in `book#\ch2`, and so on. For instance, continuing with our example defining `book3`, we define chapter 1 in `book3\ch1`. That directory must contain the file `chapter.do` containing the line:

```
---------------------------- book3\ch1\chapter.do -----------------------------
mac def S_1 "chapter-title"
-------------------------------- end of file ----------------------------------
```

The exercises within a chapter are defined in the book#\ch#\ex# directory and, in that directory must be a file desc.do describing the exercise. To define exercise 1 in chapter 1 of book 3:

```
---------------------------- book3\ch1\ex1\desc.do -----------------------------
mac def S_1 "description-of-exercise"
--------------------------------- end of file ---------------------------------
```

In addition, this directory must contain the exercise itself, coded either as exercise.ado or exercise.do. You have all the standard Stata commands available when writing this code, and you have the additions provided by cw and qa.

First, let us consider ado-files. The exercise you write may use:

Stata ado-files	(stored in the standard places)
courseware ado-files	(stored in, for example, c:\cw)
user-written, book-specific ado-files	(stored in c:\cw\book#)
user-written, exercise-specific ado-files	(stored in c:\cw\book#\ch#\ex#)

You do not have to adjust the S_ADO path (or even know what it is)—that is all handled automatically. (Note that there are no chapter-specific ado-files.)

Ado-files intended for different purposes may have the same name across books or exercises. For instance, if you have the ado-file myado.ado in c:\cw\book3\ch2\ex3 and refer to it in the corresponding exercise, it will not be confused with, say, myado.ado stored in c:\cw\book3\ch3\ex2.

Current directory when executing an exercise

When an exercise executes, the current directory is not the directory of the exercise—it is the directory in which the student happens to be when he or she typed cw. The following global macros have been set for your use:

$CW_DIR	courseware directory	e.g., c:\cw
$CW_BOOK	book directory	e.g., c:\cw\book2
$CW_CHAP	chapter directory	e.g., c:\cw\book2\ch3
$CW_EXER	exercise directory	e.g., c:\cw\book2\ch3\ex2
$CW_DLM	directory separator	e.g., \

For instance, one way to load the data mydta.dta in the exercise directory is

```
use ${CW_EXER}${CW_DLM}mydta
```

Do not "use ${CW_EXER}\mydta" because, on some systems, the directory separator is not backslash.

In point of fact, you do not need these macros because of other cw utilities. While you can load data sets with use, cw_use provides a better alternative. The syntax is

cw_use *filename*

Note the absence of a clear option—it is assumed. cw_use looks in the following places for *filename*.dta:

1st in the exercise directory	e.g., c:\cw\book3\ch2\ex3
2nd in the chapter directory	e.g., c:\cw\book3\ch2
3rd in the book directory	e.g., c:\cw\book3

If it is not found, an error is returned.

Obtaining student input

Two commands are provided for obtaining input from the keyboard:

cw_yesno "*prompt*"
cw_qnir "*prompt*" *dflt min max* [int]

cw_yesno presents *prompt* and seeks a yes or no answer; the user can abbreviate (for instance, typing y or n), but however the user responds, global macro $S_1 is returned containing yes or no. For instance, the command

```
cw_yesno "Do you wish to continue"
```

presents

```
                                Do you wish to continue (yes/no) -> . maybe
(Type yes or no)
                                Do you wish to continue (yes/no) -> . y
```

and would return yes (even though the user typed y) in $S_1. Note that the single call to cw_yesno repeated the question until the user types a correct response. (cw_yesno also allows the response end, meaning stop this and back up one level. The handling of this, and canceling the current exercise, is handled automatically.)

The oddly named cw_qnir requests a number. qnir stands for query number in range. cw_qnir presents *prompt* and, if *dflt* is not '.', tells the user that pressing enter is equivalent to typing *dflt* and then seeks a response between *min* and *max* and, if int is also specified, restricts the response to integers. The student's response is returned in $S_1. For instance,

```
. cw_qnir "Enter your choice" . 1 5
```

produces

```
                                            Enter your choice -> . alpha
(You must choose a number between 1 and 5.)
                                            Enter your choice -> . 16
(You must choose a number between 1 and 5.)
                                            Enter your choice -> . 2.2
```

and returns 2.2 in $S_1. The command

```
. cw_qnir "Enter your choice" . 1 5 int
```

produces

```
                                            Enter your choice -> . 2.2
(You must choose an integer between 1 and 5.)
                                            Enter your choice -> . 2
```

and returns 2 in $S_1. The command

```
. cw_qnir "Enter your choice" 1 1 5 int
```

produces:

```
                                  Enter your choice (default=1) -> .
```

In this case, the student merely pressed enter—1 was returned in $S_1.

cw_qnir, like cw_yesno, also allows the response end and automatically handles canceling the current exercise and backing up.

—more— conditions

Stata's automatically generated —more— conditions can be an irritation because they can appear in inappropriate places. You may, in that case, set more 1, which will turn off the automatic —more—. You do not need to worry about resetting it before you exit. If you do set more 1, use the cw-command cw_more rather than Stata's more to create more conditions. more will do nothing; cw_more will present a —more—.

Do not worry about presenting a —more— at the end of your exercise or otherwise indicating that the exercise has ended. That is handled automatically.

Example

cw frees the author to concentrate on the content of the courseware instead of the mechanics of making it run. cw serves largely an organizing and bookkeeping function. The purpose of the organization is to automatically update menus of alternatives (which book, which chapter, which exercise) and to make it easy to add a new book, chapter, or exercise without having to modify those that are already written.

At the beginning of this insert, we showed the student selecting the book *Introduction to using courseware* (the student chose `help`), the chapter *How to use courseware* (the student chose `1`), and the exercise *Introduction and use of the keyboard* (the student chose `1` again). While the exercise itself is not very impressive, here is the complete implementation:

```
------------------------------ c:\cw\book1\book.do ------------------------------
mac def S_1 "Help"
mac def S_2
mac def S_3 "Instruction to using courseware"
---------------------------------- end of file ----------------------------------


-------------------------- c:\cw\book1\ch1\chapter.do --------------------------
mac def S_1 "How to use courseware"
---------------------------------- end of file ----------------------------------


----------------------- c:\cw\book1\ch1\ex1\desc.do ---------------------------
mac def S_1 "Introduction and use of the keyboard"
---------------------------------- end of file ----------------------------------


----------------------- c:\cw\book1\ch1\ex1\exercise.do ------------------------
local d "noisily di in gr"
set more 1
#delimit ;
`d´ _n(2) in ye "Introduction to courseware" _n
"--------------------------" _n ;
`d´
"You´ve gotten this far, so you have basically figured it out." _n ;
(display lines omitted)
`d´ _n "When you see the word " in blue "--more--" in gr
" at the bottom of a screen, press the space" _n
"bar when you want to continue." _n ;
cw_more ;
(display lines omitted)
exit ;
---------------------------------- end of file ----------------------------------
```

The qa subsystem

qa assists in writing question-and-answer dialogs of the form:

```
--------------------------------------------------------------------------------
    5.  What is the null hypothesis to test whether a coin is fair in the
        sense that heads are no more or less likely than tails?
        1.  Pr(head)<.5
        2.  Pr(head)=.5
        3.  Pr(head)>.5
                                                    answer -> . 1
---->   The null hypothesis is what you are assuming by default, which is
---->   Pr(head)=Pr(tail).  Remember, too, that Pr(head)+Pr(tail)=1.
    5.  What is the null hypothesis to test whether a coin is fair in the
        sense that heads are no more or less likely than tails?
(dialog continues)
--------------------------------------------------------------------------------
```

The 5 at the beginning is called the *question number* (and is optional). "What is the null" through "less likely than tails?" is called the `text` of the question. This question has two lines of text, but other questions may have one line or many lines. The three alternatives below the question are called *answers*. In this case, there are three answers and each answer has one line of text associated with it, but there can be any number of answers and any number of lines associated with each answer. In the dialog, the student is then prompted to choose an answer and this happens automatically. What occurs following the student's choice is called an *action*, and actions are dependent on the student's choice. In this case, the student chose answer 1 and the corresponding action was to present an *error explanation* and then re-ask the question.

The idea behind the qa subsystem is to allow the courseware author to define the question number, text, answers, and actions, and then, with a single command, turn it loose. The qa subsystem itself will handle displaying the question and answers, obtaining the student's response, and executing subsequent actions. qa is designed so that, with merely a description on the courseware author's part, the majority of question-and-answer dialogs can be automatically handled. qa has a number of advanced features that allows the author to intervene and override qa's default behavior for special cases.

The syntax of the qa commands is

```
qa_drive progname      [progname      [...]]
qa_begin "question-number"
qa_dense
qa_text  "text"
qa_xeq   {show|noshow|quietly} stata-cmd
qa_ans   action        action         "text"
qa_ans2  "text"
qa_err   E#                           "text"
qa_err2  E#                           "text"
qa_ask
qa_clear
```

where *action* is `CORRECT` or `0` (meaning the same thing); `INFORM`; `PROCEED`; `RESTART`; `E#`; or *progname*.

The qa code to produce the preceding dialog is

```
qa_begin 5.

qa_text "What is the null hypothesis to test whether a coin is fair in the"
qa_text "sense that heads are no more or less likely than tails?"

qa_ans E1      INFORM  "Pr(head)<.5"
qa_ans CORRECT CORRECT "Pr(head)=.5"
qa_ans E1      INFORM  "Pr(head)>.5"

qa_err E1 "The null hypothesis is what you are assuming by default, which is"
qa_err2 E1 "Pr(head)=Pr(tail).  Remember, too, that Pr(head)+Pr(tail)=1."

qa_ask
```

Before explaining the lines one-by-one, let us review the organization. All the lines except the last—the `qa_ask`—are merely definitional in that they cause nothing to happen on the screen—qa is merely collecting the information from the author. When `qa_ask` is executed, the dialog begins and all subsequent actions dealing with this question are handled automatically. There are a number of ways that the dialog might go. The question could be asked and correctly answered; it could be asked, a nonsense answer given (in the sense that it is not one of the three alternatives), and then the student would need to be told what the valid choices are (namely, 1, 2, and 3). At that point it would be asked again and, perhaps, this time, the student gives an incorrect answer. An explanation would be presented and the question re-asked. The student might now give the correct answer or give another incorrect answer. In the latter case, given how the courseware author defined the actions, the student would then be told the correct answer. And throughout, the student might type `end`, meaning cancel this exercise and take me back a level. `qa_ask` will handle all of this.

Let's take the lines one-by-one:

`qa_begin 5.`
: This begins the definition of a qa; this is the question we are calling 5. We did not have to number the question and, numbering it, we do not have to restrict ourselves to simple numbers; we could have numbered it 7.2 by typing `qa_begin 7.2`.

`qa_text "What is the null hypothesis to test whether a coin is fair in the"`
`qa_text "sense that heads are no more or less likely than tails?"`
: This is the text of our question. This question has two lines; if it had more, we would type additional `qa_text` commands.

```
qa_ans E1     INFORM "Pr(head)<.5"
qa_ans CORRECT CORRECT "Pr(head)=.5"
qa_ans E1     INFORM "Pr(head)>.5"
```

These are the answers we wish to offer. Associated with each answer are two actions, the action to be taken the first time and then the second. For instance, if our first answer is selected, qa is to take the action `E1` (which means present error explanation 1 and re-ask the question—we will explain actions in great detail below). If that answer is selected again, the action `INFORM` (which means inform the student of the correct answer) is performed.

The action associated with the second response is `CORRECT`. This is specified twice. If the student gives this answer the first time, it is the correct answer. If the student gives, say, the first answer and then, upon seeing the error explanation `E1`, gives the second answer, the second answer is still correct.

The third answer has the same actions as the first answer; actions can be reused.

In this case, all of our answers are single-line answers; we will explain how to have multiple-line answers below—it requires using `qa_ans2`.

```
qa_err E1 "The null hypothesis is what you are assuming by default, which is"
qa_err2 E1 "Pr(head)=Pr(tail). Remember, too, that Pr(head)+Pr(tail)=1."
```

This is the definition of the error text accompanying action `E1`. When the action `E1` is invoked, this text will be displayed and then the question will be asked again. The `qa_err` command defines a new explanation and `qa_err2` adds more text to the explanation. There can be any number of `qa_err2` commands.

```
qa_ask
```

All we have done so far is define the qa sequence; when we issue the `qa_ask` command, the sequence will be executed.

Let us now consider another qa:

```
-------------------------------------------------------------------------------
  qa_begin 6.

  qa_text "Examine the following Stata output:"
  qa_xeq show "tabulate smokes sex, chi2"
  qa_text "What is the null hypothesis?"

  qa_ans E1      INFORM  "Pr(smokes|female)<Pr(smokes|male)"
  qa_ans CORRECT CORRECT "Pr(smokes|female)=Pr(smokes|male)"
  qa_ans E1      INFORM  "Pr(smokes|female)>Pr(smokes|male)"
  qa_ans CORRECT CORRECT "Males and females are equally likely to smoke"
  qa_ans CORRECT CORRECT "Smokers are equally likely to be male or female"

  qa_err E1 "<an explanation appears here>"
  qa_err2 E1 "<perhaps the explanation continues>"
  qa_ask
-------------------------------------------------------------------------------
```

Again, taking the lines one-by-one:

```
qa_begin 6.
```

We've seen this before; this is question 6.

```
qa_text "Examine the following Stata output:"
qa_xeq show "tabulate smokes sex, chi2"
qa_text "What is the null hypothesis?"
```

`qa_text` we've seen before, but the `qa_xeq` is new. This tells qa that, in the text, it is to insert the Stata command `tabulate smokes sex, chi2`. It is to appear as if we typed the command interactively at this point and the command's output will follow. It will look like this:

```
--------------------------------------------------------------------------------
    2.  Examine the following Stata output:
 . tabulate smokes sex, chi2
            | sex
     smokes |      male     female |     Total
 -----------+----------------------+----------
        yes |        21         12 |        33
         no |       100        100 |       200
 -----------+----------------------+----------
      Total |       121        112 |       233
          Pearson chi2(1) =   2.1100   Pr = 0.146
        What is the null hypothesis?
--------------------------------------------------------------------------------
```

Had we specified `noshow` rather than `show` with `qa_xeq`, the output from `tabulate` would have appeared, but the display of the command would have been suppressed. (Thus, we could display the table without emphasizing how we got it.) Had we specified `quietly` rather than `show` or `noshow`, the command would have been executed, but all output from the command would have been suppressed. (We'll illustrate the use of this below.)

```
qa_ans E1     INFORM  "Pr(smokes|female)<Pr(smokes|male)"
qa_ans CORRECT CORRECT "Pr(smokes|female)=Pr(smokes|male)"
qa_ans E1     INFORM "Pr(smokes|female)>Pr(smokes|male)"
qa_ans CORRECT CORRECT "Males and females are equally likely to smoke"
qa_ans CORRECT CORRECT "Smokers are equally likely to be male or female"
```

There is something new here, too: More than one answer is correct! That is okay; at least one answer must be `CORRECT`, but more than one may be `CORRECT`, too.

There should be no need to review `qa_err`, `qa_err2`, and `qa_ask` as we have already discussed them.

`qa_xeq quietly` can be put to good use, for instance:

```
qa_text "Examine the following Stata output:"
qa_xeq quietly drop _all
qa_xeq quietly set obs 100
qa_xeq quietly gen female=uniform()>=.5
qa_xeq quietly gen smokes=uniform<=cond(female,.14,.25)
qa_xeq show tabulate smokes, female, chi2
qa_text What is the null hypothesis?
```

In this example, the table varies each time the question is executed. The table even varies when the question is redisplayed after the student gives an incorrect answer.

Actions

The syntax of `qa_ans` is

`qa_ans` *action action* "*text*"

The actions are

`CORRECT` | `0`
`INFORM`
`PROCEED`
`RESTART`
`E#`
progname

The first action is the action taken the first time the user responds with this answer. The second action is the action taken when the user responds with this answer after the question is repeated because of first actions `E#` or *progname* from any previous answer. If the second action is `E#` or *progname*, the third action is assumed to be `INFORM`.

CORRECT (or, equivalently, 0) means that this is a correct answer; the dialog ends when the user specifies it.

INFORM means that the dialog ends when the user specifies this answer but that the user should be informed of the correct answer(s).

PROCEED means the dialog ends when the user specifies this answer and that the user should *not* be informed as to the correct answer(s). (qa_ask passes back to the caller in $S_1 whether the final response was correct. If you specify PROCEED, it becomes your responsibility to handle the incorrect response.)

RESTART means that the program currently executing is to be restarted from the beginning when the user specifies this answer. RESTART can only be used when your program is being executed by qa_drive (which is explained below).

E# means the explanation error message # is to be presented and the question re-asked.

progname means that the user-written routine *progname* is to be executed and then the question asked again.

The E# action

Explanation error messages, also known as E# actions, are created with

qa_err E# "*text*"

and additional lines of text can be added to E# with

qa_err2 E# "*text*"

The message number must be 1 or larger and, in general, you should restrict yourself to small numbers (below 100). Most everything between qa_begin and qa_ask is transitory; that is, it vanishes once the question is asked. That is not true, however, for explanation error messages. You must not define any such messages before a qa_begin, but you can freely reuse messages that you defined for previous questions. The cw system itself discards all explanation error messages when the exercise completes.

We recommend you use the messages E1 through E10 for messages specific to a particular question—messages which you will redefine freely—and use E11 and above for more general—or "permanent"—messages that will be used by various responses to various questions.

For instance, let's assume you have defined message E11 as a general explanation of null hypotheses. This message might be quite long:

```
qa_begin
qa_err  E11 "The null hypothesis is defined ..."
qa_err2 E11 "which allows ..."
qa_err2 E11 "following the ..."
qa_err2 E11 " "                     /* this is how you insert a blank line */
qa_err2 E11 "You can then ..."
qa_err2 E11 "etc."
```

You put this at the top of your file because you will use E11 frequently. (You can qa_begin without ever doing a qa_ask.)

You now define a question:

```
qa_begin
qa_text "Consider ..."
qa_ans E1 E11 "Pr(heads)<Pr(tails)
qa_ans CORRECT CORRERCT "Pr(heads)=Pr(tails)"
qa_ans E1 E11 "Pr(heads)>Pr(tails)"
qa_err E1 "When dealing with coins ..."
qa_ask
```

Here is what will happen: The question will be asked. Assume the student gives an incorrect response. The message E1 will be displayed, a message specific to this question, and the question repeated. Assume the student gives the wrong response again. The message E11, a more detailed but general explanation, will be displayed and the question asked again . If the student is wrong one more time, the student will be told the correct answer and we will continue.

The RESTART action

You have qanda.ado which you have written using the qa routines. It asks a series of five questions concerning identification of the null hypothesis. For questions 3, 4, and 5, if the student specifies the wrong answer two times in a row, you want to restart qanda.ado from the beginning. You might code a wrong-answer response as

```
qa_ans E3 RESTART "Pr(x)>Pr(y)"
```

This means that if this answer is chosen, first the explanation error message 3 is presented and then, qanda.ado is to be restarted from the beginning.

You may only specify RESTART, however, if qanda.ado is being executed by qa_drive. If you were doing a courseware exercise, the file exercise.ado would contain the single line:

```
qa_drive qanda
```

When the exercise is executed, qa_drive is used to execute your ado-file qanda.ado.

Special handling of incorrect responses

There may be occasions when incorrect responses need special handling that cannot be coded into the qa system. There are two ways to handle such cases. One way is to use the PROCEED action. PROCEED, from the qa point-of-view, works much like CORRECT—the qa_ask is terminated but the student is given no indication that he or she provided an incorrect response. qa_ask defines the following global macros at its conclusion:

$S_1	0 if the last given response is CORRECT and 1 otherwise
$S_2	the response last given
$S_3	the response given before that (or "" if there is none)
$S_4	the response given before that (or "" if there is none)

Thus, using standard Stata programming techniques, you can handle the situation:

```
qa_begin
qa_text "Consider ..."
qa_ans E1 PROCEED "Pr(heads)<Pr(tails)
qa_ans CORRECT CORRERCT "Pr(heads)=Pr(tails)"
qa_ans E1 PROCEED "Pr(heads)>Pr(tails)"
qa_err E1 "When dealing with coins ..."
qa_ask
if $S_1==1 {                    /* if the 2nd response was incorrect */
        (Standard Stata code or, perhaps, ask another question)
}
```

The macros $S_1, $S_2, $S_3, and $S_4 are defined not only when you specify PROCEED, they are always defined. Thus, the sample code above is always valid and, even when you do not specify the PROCEED action, you can know whether the student finally delivered the correct answer and, if not, what answers the student did deliver.

The second alternative is to specify the action *progname*. That is, you write an ado-file, say, problem1.ado, and then specify the action as problem1:

```
qa_begin
qa_text "Consider ..."
qa_ans E1 problem1 "Pr(heads)<Pr(tails)
qa_ans CORRECT CORRERCT "Pr(heads)=Pr(tails)"
qa_ans E1 problem1 "Pr(heads)>Pr(tails)"
qa_err E1 "When dealing with coins ..."
qa_ask
```

Here is what will happen: The question will be asked. Assume the student gives an incorrect response. The message E1 will be displayed, a message specific to this question, and the question repeated. Assume the student gives the wrong response again. The ado-file problem1.ado will be executed and is free to do whatever you wish to code (including using qa to perform other question-and-answer dialogs). When your ado-file concludes, the original question is asked again. If the student is wrong one more time, the student will be told the correct answer and we will continue.

Other qa commands

`qa_dense`. When `qa_ask` presents a question, it usually inserts a blank line after the question, between each of the answers, and between the last answer and the request for the student response. Specifying `qa_dense` eliminates these blank lines. (In most of the examples above, we specified `qa_dense` but did not mention it.) You may specify it anywhere between the `qa_begin` and `qa_ask`; it must be respecified for each question that is be presented densely.

`qa_clear`. You should never have reason to call `qa_clear`; it removes all `E#` explanation error messages. `qa_clear` is used by cw to free memory after running an exercise. Existing explanation error messages may be redefined freely without clearing them first.

Request for comments

This is a first draft of a courseware system. All the software is on the STB diskette (including on-line help). It works. We are reasonably satisfied. We can, however, already think of shortcomings. Occurring to us is

1. The use of the words book, chapter, exercises, and student may be unfortunate in that the ideas we have implemented have broader application than working only with students in lab/classroom situations (Stata tutorials occur to us). Perhaps many users who might be interested in cw have already stopped reading because of the jargon. Worse, the jargon is built into the software. There needs to be overrides.
2. We have not done anything about recording performance of (real) students (in terms of the number of questions answered correctly and incorrectly, etc.). This information could be collected and even saved for turning in with other class assignments.
3. There needs to be a way to temporarily drop the "student" into Stata.
4. While performance is most satisfactory on 386/25MHz and above, performance is poor on an old 10MHz 8088 system on which we also tested (it took 5 seconds to complete each of the front menus and there was a 3- to 5-second delay in the qa subsystem). We do not know if anybody cares, but we are concerned that student labs may be using such slow computers.
5. We have given no thought to graphs. This is not to say there is a problem; we just have not thought about any problems that may exist. Perhaps some special handling is needed.
6. One feature of the system not mentioned is that the "student" can type `end` to any query to have cw stop whatever is going on and backup one level. We think there also needs to be another such anytime-you-want sequence (such as `info`) that displays a statement of context (such as redisplaying the question, etc.)

We would not be surprised to hear that there are other shortcomings as well. Therefore, we request comments.

tt6.1	Courseware guidelines

William Gould and William Rogers, CRC, FAX 310-393-7551

Our implementation of cw *[tt6 above—Ed.]* has made us think about courseware in general. What is it and how should it be written?

The first question that needs to be answered is whether statistical courseware should even be written in Stata. That is, courseware—whatever it is—deals at a programming level with concepts and problems that are quite foreign to Stata. Should not a courseware system be written outside of Stata where a new language can be designed, without constraints, intended to address the courseware author's problems? We think that, in general, the answer is that it should. For statistical courseware, however, there may be offsetting advantages to embedding the system in a statistical package.

First, authors need not tediously type out statistical output to include in their questions and explanations; that output is readily available from the statistical package itself. Moreover, the underlying statistical engine allows randomly generating data sets and statistical results therefrom, so that the exercises need not be based on prerecorded problems; they can change every time the student runs them. Second, the student can use the statistical package in the course of determining the answer.

In thinking about courseware, we have found the following decomposition useful:

1. Executive-summary courseware intended for use in place of a written text.
2. Practice-session courseware intended for use with a text and designed to reinforce knowledge or to test knowledge.
3. Exploratory courseware intended to allow the motivated student to explore on his or her own.

In testing our own system, we have tried to develop a small piece of courseware corresponding to each of these goals. The instructions on using `cw` in cw itself is an example of executive summary (not shown), the questions-and-answers about hypothesis testing (shown in the opening example of *tt6*) are examples of practice courseware, and here is an example of exploratory courseware:

```
. cw

Keyword    Author and title
------------------------------------------------------------------------------
Help       Instruction on using courseware
Iman       R. L. Iman, A Data Based Approach to Understanding Statistics
Enter a keyword or end -> . iman


Chapter    Contents
------------------------------------------------------------------------------
1          Data collection:  The role of sampling in statistics
7          Hypothesis testing
Enter a chapter number or end -> . 1


Exercise   Description
------------------------------------------------------------------------------
1          Estimating television viewership
Enter an exercise number or end -> . 1


Estimating television viewership
--------------------------------

There are 10 television stations among which you are considering advertising.
The total viewership on any given night is 1 million people, it is just a
matter of determining how the viewership is distributed.

To find out, you may survey the viewership by calling viewers chosen at random
and asking them what they are watching.  The cost of the survey is $4,000 plus
$10 for each person you ask.

One in a hundred persons who watch your advertisement will call your 800 number
and purchase the product and you will make $4 from each call.

Your problem is to choose the size of the survey.  Remember, there are 1 mil-
lion people watching TV each night.  You will, we think, be surprised by how
small the survey can be.  In the simulation that follows, you will choose the
number of people to survey and then be shown the survey results.  You will then
choose the station on which to advertise.  You can repeat the simulation until
you think you know the "best" number.

Surveys take time and large surveys take even more time, even in this simula-
tion.  Try surveying 1,000 people to begin.
                                  Enter the number to survey -> . 1000

Survey results are back:
     channel|      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          1        0.10        0.10
          2 |        122       12.20       12.30
          3 |        293       29.30       41.60
          4 |        120       12.00       53.60
          5 |         14        1.40       55.00
          6 |        231       23.10       78.10
          7 |         35        3.50       81.60
          9 |         27        2.70       84.30
         10 |        157       15.70      100.00
------------+-----------------------------------
      Total |       1000      100.00
(nobody reported watching channel 8)
                                       Choose your channel -> . 6
```

```
     Are you sure you don't want to choose channel 3 (yes/no) -> . n
                                          Choose your channel -> . 3

(time passes)
2757 people are reported calling your 800 number.
------------------------------------------------------------------------------
Memo to:  Director of marketing
   From:  the CFO
     Re:  results of ad campaign

         Profit from 2757 calls:           11028.00
         Cost of surveying 1000 people:    14000.00
                                      ------------
         Net                               -2972.00

Looking forward to receiving your resignation.
------------------------------------------------------------------------------



                                  Want to try again (yes/no) -> . yes
                                  Enter the number to survey -> . 500

Survey results are back:
     channel|      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         16        3.20        3.20
          2 |         10        2.00        5.20
          3 |          7        1.40        6.60
          4 |         69       13.80       20.40
          5 |        104       20.80       41.20
          6 |          3        0.60       41.80
          7 |          8        1.60       43.40
          8 |        147       29.40       72.80
          9 |         28        5.60       78.40
         10 |        108       21.60      100.00
------------+-----------------------------------
      Total |        500      100.00

                                          Choose your channel -> . 8

(time passes)
2737 people are reported calling your 800 number.
------------------------------------------------------------------------------
Memo to:  Director of marketing
   From:  the CFO
     Re:  results of ad campaign

         Profit from 2737 calls:           10948.00
         Cost of surveying 500 people:      9000.00
                                      ------------
         Net                                1948.00
------------------------------------------------------------------------------



                                  Want to try again (yes/no) -> . y
                                  Enter the number to survey -> . 30

Survey results are back:
     channel|      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          6       20.00       20.00
          2 |          2        6.67       26.67
          3 |          8       26.67       53.33
          5 |          6       20.00       73.33
          8 |          4       13.33       86.67
          9 |          4       13.33      100.00
------------+-----------------------------------
      Total |         30      100.00

(nobody reported watching channels 4 6 7 10)

                                          Choose your channel -> . 3

(time passes)
1649 people are reported calling your 800 number.
(Too bad you didn't choose channel 9.  You'd have gotten 2660 calls.)
```

```
------------------------------------------------------------------------------
Memo to:  Director of marketing
   From:  the CFO
     Re:  results of ad campaign

         Profit from 1649 calls:            6596.00
         Cost of surveying 30 people:       4300.00
                                          ----------
         Net                                2296.00
------------------------------------------------------------------------------


                                  Want to try again (yes/no) -> . n
(That concludes the exercise.)

Exercise   Description
------------------------------------------------------------------------------
1          Estimating television viewership
Enter an exercise number or end -> . end


Chapter    Contents
------------------------------------------------------------------------------
1          Data collection:  The role of sampling in statistics
7          Hypothesis testing
Enter a chapter number or end -> . end


Keyword    Author and title
------------------------------------------------------------------------------
Help       Instruction on using courseware
Iman       R. L. Iman, A Data Based Approach to Understanding Statistics
Enter a keyword or end -> . end
Goodbye.
```

In writing our examples, we have discovered the need for guidelines. First, let us say that neither of us are experts in how courseware should be written. We suspect the same can be said of most academics. As novice courseware authors, we often had to make choices, such as how questions should be phrased, how much explanation should be offered and when, and so on. We would have appreciated some guidelines. The one guideline we did make for ourselves, and which we found useful, was deciding for each exercise whether we were writing executive summary, practice, or exploratory courseware. We believe that courseware exercises that jump across these boundaries simply do not have an audience.

We envision a booklet for courseware authors providing examples of each of the types of courseware and explanations, at the courseware level, of why these are good examples with recommendations and advice for avoiding pitfalls. For example:

1. In executive-summary courseware, capitalize on your implicit knowledge of the student's progress to speed up or slow down the pace.
2. In practice courseware, make sure that there is a general explanation that the user is referred to if he or she is not performing well.
3. In exploratory courseware, be sure the student has a sense of controlling the direction in which the exercise goes.

Along these lines, we seek the guidance of those who, because they teach, know more than us.

tt6.2	Duxbury Press looking for courseware authors

Stan Loll, Editor, Statistical Computing, Duxbury Press, 415-637-7596

Duxbury Press is an imprint of Wadsworth Publishing Company. We publish exclusively in statistics and such associated fields as operations research, decision sciences, and quality control. We publish textbooks and we publish, for instance, *Statistics with Stata 3*. We are also interested in publishing courseware.

Duxbury hopes the cw system outlined above *[tt6—Ed.]* will give the creative teacher the ability to fashion an interactive educational environment to transcend the traditional lecture/textbook model and, if it does, we would be interested in publishing such courseware. We are interested in courseware that uses Stata's powerful graphics and computational power to engage, motivate, and teach the student. We are interested in demonstrations, simulations, and built-in interactive problem sets that can be merged with trips into Stata itself to work directly on problems.

I am excited about cw and looking for courseware authors. Examine cw and, if you see possibilities in higher education and want to develop materials, please contact me:

Stan Loll, Editor
Duxbury Press
Phone: (415) 637-7596
Fax: (415) 592-9081
Email: stan_loll@wadsworth.com