STATA
TECHNICAL
BULLETIN

November 1993
STB-16

A publication to promote communication among Stata users

Editor

Sean Becketti
Stata Technical Bulletin
14619 West 83rd Terrace
Lenexa, Kansas 66215
913-888-5828
913-888-6708 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
Joseph Hilbe, Arizona State University
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	*qs*	questions and suggestions
dt	data sets	*tt*	teaching
gr	graphics	*zz*	not elsewhere classified
in	instruction		

Statistical Categories:

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

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

an37 Stata classes and programming services available

Eric Best, Best Consultants, 818-340-1146

Best Consultants is now offering Stata programming services and scheduling Stata classes at various locations. Stata classes are being offered at beginning and intermediate levels. Let us know what subjects interest you. On-site classes are available.

We also offer Stata data analysis and programming services. Services include data analysis where your time and personal resources might be limited and yet you need results quickly. Programming in Stata is available for both data manipulation and analysis needs.

Please contact:

Eric Best, Ph.D.
Best Consultants
21450 Chagall Road
Topanga, California 90290
Telephone: 818-340-1146

dm15 Interactively list values of variables

Alan Riley, Stata Corporation, FAX 409-696-4601

Stata's `list` command is adequate when looking at data sets with small numbers of variables, or when a variable name is desired adjacent to each data value (*display* format). However, when large numbers of variables are present, a listing becomes impossible to read without specifying a subset of the variable list. Another shortcoming of `list` is that there is no way to page up during the listing. The `browse` command acts as an interactive listing command, displaying only as many columns as will fit on the screen at one time, and allowing you to look through a large data set easily. The syntax of browse is

`browse` [*varlist*] [`if` *exp*] [`in` *range*] [`, nolabel noobs`]

`browse` is typically invoked without specifying any arguments, but, if you specify the *varlist*, the listing will be restricted to solely those variables just as with `list`. The same is true of the `if` *exp* and `in` *range*.

After typing `browse`, the first "page" of data is displayed, and you are requested to enter an action:

```
. browse
(determining variable widths...)

                  make     price       mpg     rep78  hdroom
 1.        AMC Concord      4099        22         3     2.5
 2.          AMC Pacer      4749        17         3     3.0
 3.         AMC Spirit      3799        22         .     3.0
 4.      Buick Century      4816        20         3     4.5
 5.      Buick Electra      7827        15         4     4.0
 6.      Buick LeSabre      5788        18         3     4.0
 7.         Buick Opel      4453        26         .     3.0
```

```
  8.        Buick Regal      5189        20         3     2.0
  9.      Buick Riviera     10372        16         3     3.5
 10.      Buick Skylark      4082        19         3     3.5
 11.       Cad. Deville     11385        14         3     4.0
 12.      Cad. Eldorado     14500        14         2     3.5
 13.       Cad. Seville     15906        21         3     3.0
 14.     Chev. Chevette      3299        29         3     2.5
 15.       Chev. Impala      5705        16         4     4.0
 16.       Chev. Malibu      4504        22         3     3.5
 17.  Chev. Monte Carlo      5104        22         2     2.0
 18.        Chev. Monza      3667        24         2     2.0
 19.         Chev. Nova      3955        19         3     3.5

browse (? for help):  .
```

Typing '**?**' and pressing *Return* (or simply pressing *F1*) provides on-line help:

```
browse (? for help):  . ?

     F-Key  Command        Function
     ------------------------------------------------------------
       F1      ?         this help screen
       F2      a         again: repeat last command
       F3      l         move left 1 variable
       F4      r         move right 1 variable
               l#        move left # variables
               r#        move right # variables
       F7      u         move up 1 page (b synonym)
       F8      d         move down 1 page (f or Return synonyms)
       F5      uh        move up 1 half page
       F6      dh        move down 1 half page
               u#        move up # lines
               d#        move down # lines
       F10     R         redisplay current page
               h vn      hold vn (varname) at left of screen
               h vn vn   hold 2 variables at left of screen
               h         release all hold variables
               p#        reset page size to # lines
               q         quit

browse (? for help):  .
```

An action is chosen by typing the "command" or by pressing the corresponding *F*-key.

The most common action is to simply press *Return*, which goes to the next page:

```
browse (? for help):  .

                  make     price       mpg     rep78  hdroom
 20.         Dodge Colt      3984        30         5     2.0
 21.     Dodge Diplomat      4010        18         2     4.0
 22.       Dodge Magnum      5886        16         2     4.0
 23.    Dodge St. Regis      6342        17         2     4.5
 24.        Ford Fiesta      4389        28         4     1.5
 25.       Ford Mustang      4187        21         3     2.0
 26.  Linc. Continental     11497        12         3     3.5
 27.       Linc. Mark V     13594        12         3     2.5
 28.   Linc. Versailles     13466        14         3     3.5
 29.       Merc. Bobcat      3829        22         4     3.0
 30.       Merc. Cougar      5379        14         4     3.5
 31.      Merc. Marquis      6165        15         3     3.5
 32.      Merc. Monarch      4516        18         3     3.0
 33.         Merc. XR-7      6303        14         4     3.0
 34.       Merc. Zephyr      3291        20         3     3.5
 35.            Olds 98      8814        21         4     4.0
 36.     Olds Cutl Supr      5172        19         3     2.0
 37.       Olds Cutlass      4733        19         3     4.5
 38.      Olds Delta 88      4890        18         4     4.0

browse (? for help):  .
```

We would have obtained the same result had we typed `f` (or `d`) and pressed *Return*, or if we simply pressed *F8*.

`browse`, unlike `list`, can go backwards. If we now press `b` (or `u`) followed by *Return*, or *F7*, we move back up one "page" of data:

```
browse (? for help):  . b

                  make     price       mpg     rep78  hdroom
  1.        AMC Concord      4099        22         3     2.5
  2.          AMC Pacer      4749        17         3     3.0
  3.         AMC Spirit      3799        22         .     3.0
  4.      Buick Century      4816        20         3     4.5
  5.      Buick Electra      7827        15         4     4.0
  6.      Buick LeSabre      5788        18         3     4.0
  7.         Buick Opel      4453        26         .     3.0
  8.        Buick Regal      5189        20         3     2.0
  9.      Buick Riviera     10372        16         3     3.5
 10.      Buick Skylark      4082        19         3     3.5
 11.       Cad. Deville     11385        14         3     4.0
 12.      Cad. Eldorado     14500        14         2     3.5
 13.       Cad. Seville     15906        21         3     3.0
 14.     Chev. Chevette      3299        29         3     2.5
 15.       Chev. Impala      5705        16         4     4.0
 16.       Chev. Malibu      4504        22         3     3.5
 17.  Chev. Monte Carlo      5104        22         2     2.0
 18.        Chev. Monza      3667        24         2     2.0
 19.         Chev. Nova      3955        19         3     3.5

browse (? for help):  .
```

We can also go up and down half pages by typing `uh` or `dh` (equivalent to *F5* or *F6*), or we can go down (up) an arbitrary number of lines by typing `d` (`u`) followed by the number, such as `d8`:

```
browse (? for help):  . d8

                  make     price       mpg     rep78  hdroom
  9.      Buick Riviera     10372        16         3     3.5
 10.      Buick Skylark      4082        19         3     3.5
 11.       Cad. Deville     11385        14         3     4.0
 12.      Cad. Eldorado     14500        14         2     3.5
 13.       Cad. Seville     15906        21         3     3.0
 14.     Chev. Chevette      3299        29         3     2.5
 15.       Chev. Impala      5705        16         4     4.0
 16.       Chev. Malibu      4504        22         3     3.5
 17.  Chev. Monte Carlo      5104        22         2     2.0
 18.        Chev. Monza      3667        24         2     2.0
 19.         Chev. Nova      3955        19         3     3.5
 20.         Dodge Colt      3984        30         5     2.0
 21.     Dodge Diplomat      4010        18         2     4.0
 22.       Dodge Magnum      5886        16         2     4.0
 23.    Dodge St. Regis      6342        17         2     4.5
 24.        Ford Fiesta      4389        28         4     1.5
 25.       Ford Mustang      4187        21         3     2.0
 26.  Linc. Continental     11497        12         3     3.5
 27.       Linc. Mark V     13594        12         3     2.5

browse (? for help):  .
```

The page size may be reset by typing `p#`, where # is the number of lines of data to display on the screen:

```
browse (? for help):  . p10

                  make     price       mpg     rep78  hdroom
  9.      Buick Riviera     10372        16         3     3.5
 10.      Buick Skylark      4082        19         3     3.5
 11.       Cad. Deville     11385        14         3     4.0
 12.      Cad. Eldorado     14500        14         2     3.5
 13.       Cad. Seville     15906        21         3     3.0
 14.     Chev. Chevette      3299        29         3     2.5
 15.       Chev. Impala      5705        16         4     4.0
 16.       Chev. Malibu      4504        22         3     3.5
 17.  Chev. Monte Carlo      5104        22         2     2.0
 18.        Chev. Monza      3667        24         2     2.0

browse (? for help):  .
```

When you reset the page size, the new size remains in effect throughout the Stata session. The page size may also be reset before entering `browse` with the `brset #` command. The default page size is 19, meaning 19 observations are displayed per page.

You may move left and right along the variables by typing `l` or `r` (or *F3* or *F4*):

```
browse (? for help):  . r

       price       mpg     rep78  hdroom     trunk
 9.    10372        16         3     3.5        17
10.     4082        19         3     3.5        13
11.    11385        14         3     4.0        20
12.    14500        14         2     3.5        16
13.    15906        21         3     3.0        13
14.     3299        29         3     2.5         9
15.     5705        16         4     4.0        20
16.     4504        22         3     3.5        17
17.     5104        22         2     2.0        16
18.     3667        24         2     2.0         7

browse (? for help):  .
```

You may use `r#` and `l#` to move right and left # of variables; `r2` moves right two variables.

During moves to the right or left, it is sometimes desirable to hold one or two identifier variables at the left of the screen. This is accomplished by typing `h` *varname1* [*varname2*]:

```
browse (? for help):  . h make

                  make     price       mpg     rep78  hdroom
 9.      Buick Riviera     10372        16         3     3.5
10.      Buick Skylark      4082        19         3     3.5
11.       Cad. Deville     11385        14         3     4.0
12.      Cad. Eldorado     14500        14         2     3.5
13.       Cad. Seville     15906        21         3     3.0
14.     Chev. Chevette      3299        29         3     2.5
15.       Chev. Impala      5705        16         4     4.0
16.       Chev. Malibu      4504        22         3     3.5
17.  Chev. Monte Carlo      5104        22         2     2.0
18.        Chev. Monza      3667        24         2     2.0

browse (? for help):  . r2

                  make     rep78  hdroom     trunk    weight
 9.      Buick Riviera         3     3.5        17      3880
10.      Buick Skylark         3     3.5        13      3400
11.       Cad. Deville         3     4.0        20      4330
12.      Cad. Eldorado         2     3.5        16      3900
13.       Cad. Seville         3     3.0        13      4290
14.     Chev. Chevette         3     2.5         9      2110
15.       Chev. Impala         4     4.0        20      3690
16.       Chev. Malibu         3     3.5        17      3180
17.  Chev. Monte Carlo         2     2.0        16      3220
18.        Chev. Monza         2     2.0         7      2750

browse (? for help):  .
```

After variables have been "held," they may be released by typing `h` with no varnames specified.

```
browse (? for help):  . h

       price       mpg     rep78  hdroom     trunk    weight
 9.    10372        16         3     3.5        17      3880
10.     4082        19         3     3.5        13      3400
11.    11385        14         3     4.0        20      4330
12.    14500        14         2     3.5        16      3900
13.    15906        21         3     3.0        13      4290
14.     3299        29         3     2.5         9      2110
15.     5705        16         4     4.0        20      3690
16.     4504        22         3     3.5        17      3180
17.     5104        22         2     2.0        16      3220
18.     3667        24         2     2.0         7      2750

browse (? for help):  .
```

Finally, `a` (or *F2*) is used to repeat the last command issued; `R` (*F10*) is used to redisplay the current "page" of observations; `q` allows you to quit `browse` and return to Stata's dot prompt.

```
browse (? for help):  . q

.
```

sg16.3 | Quasi-likelihood modeling using an enhanced `glm` command

Joseph Hilbe, Sociology & Statistics, Arizona State University, EMAIL atjmh@asuvm.inre.asu.edu

As currently structured the `glm` command allows modeling of Gaussian, binomial (logit, probit, and complementary loglog), Poisson, gamma, inverse Gaussian, and negative binomial regressions. All valid GLM links are provided, including the full range of power links. Each GLM model is based upon a probability (density) function which is a member of the exponential family of distributions. Given such a probability function, the method of moments allows one to determine the mean and variance for the regression model. The log-likelihood function is also directly determined from the probability function. The deviance function, upon which convergence is based, is defined as twice the difference between the maximal and fitted log-likelihoods; so it too is dependent upon the specific probability function. Hence, for each standard GLM model, there is a direct relationship between the probability, deviance, and variance functions.

There are times when it may be desirable to add an extra multiplicative factor to the variance, for instance when dealing with limited overdispersed-Poisson count and binomial proportion data. When such a factor is introduced into the variance function, the relationship between probability function and variance breaks down—and a new deviance function must be constructed. In 1974 Robert Wedderbrun discovered that standard GLM deviance functions could be derived, independently of a knowledge of the underlying probability functions, from the following formula:

$$2\int_{\mu}^{y} \frac{y-\mu}{V(\mu)}d\mu.$$

That is, given a variance function, it may be possible to calculate an appropriate log-likelihood and deviance function. For example, the Poisson variance is μ. Integration of $(y-\mu)/\mu$ determines the log likelihood: $y\log(\mu)-\mu$. Subtracting this LL from that of the maximal model, i.e., one where y is substituted for each μ, and multiplying by 2, results in the deviance function: $2\{y\log(y/\mu)-(y-\mu)\}$. This is the same Poisson deviance function calculated more laboriously from the Poisson probability function. Adding an extra multiplicative constant to the Poisson variance, mk, results in a deviance function which is divided by the multiplicative constant, D/k. However, the corresponding probability function is not a member of the exponential family and the model is thus not strictly speaking a GLM—it is called a quasi-likelihood (QL) model. Many QL models fortunately share asymptotic properties characteristic of GLMs; hence allowing users a wide variety of meaningful modeling and diagnostic capabilities. Other QL models can only be tenuously interpreted. However, QL modeling does allow interesting and useful extensions to the standard GLM models.

An enhanced `glm` program is supplied on the STB diskette which allows the addition of a multiplicative factor into the variance function. Moreover, I have withdrawn the `kl` option which was to be used for linearly specified negative binomial models (LNB). The LNB is a QL model with a deviance function proportional to that of the Poisson deviance. You may calculate this for yourself by using the LNB variance function, $\mu+k\mu$, as the denominator in the formula earlier described. The resultant QD function is $[1/(k+1)](\text{Poisson Deviance})$. Hence, a separate LNB program is unnecessary; simply use an overdispersed Poisson model to model a LNB. Of course, the use of the log-linked negative binomial GLM will probably yield a more satisfactory modeling result, particularly if the Poisson cells are gamma distributed. Display the `glm` help file to determine how to use this feature.

There are many other QL models which are not directly accommodated by the `glm` program. I have constructed several including generalized power variances, distribution mixtures, and so forth. All may be programmed using Stata programming facilities.

sg16.4	Comparison of `nbreg` and `glm` for negative binomial

William H. Rogers, Stata Corporation, FAX 409-696-4601

Stata 3.1's `nbreg` command (see [5s] nbreg) estimates negative binomial models. The previously published `glm` command (Hilbe 1993b) can also estimate such models. The two programs are based on very different philosophies: they estimate slightly different models and yield slightly different answers.

`nbreg` is based on a full maximum-likelihood (ML) estimates that include regression parameters for the log expected count and also the ancillary dispersion parameter α. The reported standard errors are unconditional estimates derived from the second derivative of the log-likelihood function.

`glm` is embedded in a standard power-link GLM framework (McCullagh and Nelder 1989). Using a power of zero results in estimates of the log expected count. The ancillary parameter is not estimated; it must be previously known and is specified using `glm`'s `k()` option where $k = \alpha$. (The parameter k is defined as $1/\alpha$ in McCullagh and Nelder, but `glm`'s `k()` option uses the inverse.) Standard errors are defined conditionally.

Using the `kyphsp.dta` data (Hastie and Tibshirani 1990, 200; available in Stata format in Hilbe 1993a), I estimated a negative binomial model using `nbreg` and also with `glm`. Since `glm` does not estimate the ancillary parameter, I specified its value to be that found by `nbreg`. The results were as follows: `nbreg` reported a higher log-likelihood value, but the difference was only at the 10th significant digit and the reported log likelihoods were even closer when `glm`'s `ltolera(1e-8)` option was specified. The parameter estimates agreed to four significant digits. The standard errors, however, agreed to only 1.5 significant digits.

There are four possible explanations as to why the standard errors might differ: (1) `nbreg`'s standard errors are unconditional whereas `glm`'s are conditional on the value specified for the ancillary parameter; (2) `nbreg`'s standard errors are computed numerically whereas `glm`'s standard errors are based on analytic formulas; (3) The differences in the standard errors could be due to the (small) differences in the parameter estimates; (4) `nbreg`'s standard errors are based on second derivatives whereas `glm`'s are based on another type of approximation.

Possibility 1 can be examined by setting the appropriate partial derivatives to zero and thus allowing us to compare the conditional and unconditional standard errors. I found that making `nbreg`'s results conditional would have changed the answers only slightly—at the sixth significant digit. To examine possibilities 2 and 3, I wrote a program to evaluate the derivatives at the maximum found by `glm` using a different numerical differentiation technique. This produced answers that agreed to four significant digits. Thus, possibilities 1, 2, and 3 are not the principal reasons for the observed differences in the standard errors.

Further examination revealed that the principal reason is possibility 4. Both procedures estimate standard errors from a matrix of the form $\mathbf{X}'\mathbf{W}\mathbf{X}$ where $\mathbf{W}$ is a diagonal matrix of weights. As it turns out, the GLM procedure estimates roughly the same weight for every observation whereas the maximum-likelihood `nbreg` estimates different weights for each observation. For `nbreg`, the weights are a function of the dependent variable and the independent variables. For `glm`, they are only a function of the independent variables. For a given set of independent variables, the two sets of weights have constant expected ratio. If the negative binomial regression model holds, the two answers will agree asymptotically and will be consistent. In small samples, the `glm` answer is more stable. If the model is a little off, however, the ML standard errors produced by `nbreg` are probably slightly better, although neither answer would be asymptotically consistent (correct). The ML standard errors are more conditional on the data than those of GLM.

In summary, it is difficult to know which standard errors should be considered "better." If there are substantial differences in the standard errors and the data set was large, however, chances are the negative binomial assumption should be questioned.

From a practical standpoint, the ML procedure `nbreg` is easier to use because the user does not have to specify the ancillary parameter and can test its values. With the ancillary specified, `glm` is faster. The standard errors produced by `glm` depend on the ancillary parameter's value and will be biased if this ancillary parameter is misspecified. This might be fixed by using `glm`'s `scale()` option. Still, the best way we know to find the ancillary value is to run `nbreg` first.

References

Hastie, T. J. and R. J. Tibshirani. 1990. *Generalized Additive Models.* New York: Chapman & Hall.

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

——. 1993b. sg16.2: GLM: A unified power-link based program including the negative binomial. *Stata Technical Bulletin* 14: 16–19.

McCullagh, P. and J. A. Nelder. 1989. *Generalized Linear Models.* 2d ed. New York: Chapman & Hall.

snp6	Exploring the shape of univariate data using kernel density estimators

Isaías Hazarmabeth Salgado-Ugarte, Makoto Shimizu, and Toru Taniuchi,
University of Tokyo, Faculty of Agriculture, Department of Fisheries, Japan
FAX (011)-81-03-3812-0529

There are many graphical devices for examining the distribution of a single variable. Examples include one-way scatterplots, stem-and-leaf displays, boxplots, and histograms. There are also several types of quantile plots that use the empirical cumulative distribution function (for example, see Wilk and Gnanadesikan, 1968). According to Fox (1990), though, it is easier to spot important features when an empirical distribution is displayed as a density function rather than as a cumulative distribution function. While frequency plots (histograms) provide an accurate picture of variables from discrete distributions, smooth density functions are needed to represent variables drawn from continuous distributions such as the normal distribution. Because the density function is the derivative of the cumulative distribution function, areas under the density function are probabilities.

This insert introduces some kernel estimators that provide nonparametric density estimates along with ado-files to calculate them. Kernel density estimators are an essential component of many more complicated estimators, such as semiparametric procedures. In this insert, we will stress the use of kernel estimators as tools for the exploratory stage of data analysis. In this context, kernel estimators can be regarded as nonparametric histogram smoothers. From an exploratory point of view, density estimates are valuable because they can reveal skewness, heavy or light tails, and multimodality in the data, characteristics that can be investigated further at the confirmatory stage (Silverman 1986).

This insert is based in large part on the account in Fox (1990). We begin by discussing the histogram, one of the earliest and simplest density estimators. The disadvantages of the histogram serve to motivate the discussion of density traces and kernel density estimators. We illustrate the performance of these alternative estimators on a variable that exhibits multiple modes. We close by considering techniques for selecting an optimal kernel estimator and by documenting the heavy computational requirements of these conceptually simple estimators.

The histogram: Some remarks on the classical density estimator

The method most widely used to represent the shape of a probability density function is the histogram. Tarter and Kronmal (1976) provide a critical review of this procedure and note that the histogram is useful for data description and can provide a crude estimate of density (Chambers et al. 1983). However, the histogram density estimate frequently is too crude for many purposes. The more sophisticated estimators presented below are all refinements, however, of the simple histogram.

The histogram is calculated by partitioning the real line into equal-sized line segments, called bins. The fraction of observations of the variable of interest that fall within a given bin is taken as an estimate of the probability of observing future realizations in that bin. The histogram is drawn as a sequence of bars, representing the simplifying assumption that the probability density is constant within each bin.

Silverman (1986) and Fox (1990) give a formal definition of the histogram density estimator.

Let x	=	the variable of interest,
m	=	the number of bins,
h	=	half the width of each bin,
$f(x)$	=	the density function.

If x_0 is the origin of the histogram, that is, the lower endpoint of the leftmost bin, the end points of the bins are given by the sequence

$$x_0, x_0 + 2h, x_0 + 4h, \ldots, x_0 + m(2h).$$

A particular observation X_i falls in bin j if

$$x_0 + (j-1)2h \leq X_i < x_0 + j(2h).$$

By this definition, an observation falling on a bin boundary is placed in the higher bin. Formally, the density function is continuous from the left.

The *histogram density estimator* is given by

$$\widehat{f}_H(x) = \frac{\#[x_0 + (j-1)2h \leq X_i < x_0 + j(2h)]}{n(2h)}$$

for x located in bin j. In this expression, the symbol $\#[\cdot]$ means a count of the number of observations for which the expression in brackets is true. Thus, the numerator counts the number of observations in the bin. The denominator, $n(2h)$ ensures that the

total area enclosed by the histogram is unity. It is common to ignore the denominator and instead scale the vertical axis so that the heights of the bars represent frequency or percent (Fox 1990, Chambers et al. 1983).

Figures 1 and 2 display Stata's version of the histogram. The data are adapted from Goeden (1978) and consist of 316 length observations of the coral trout *Plectropomus leopardus*. These data are included on the STB distribution disk in the file `trocolen.dta`. Figure 1 is Stata's default version, a histogram with five bins. It was produced by the command:

```
. graph length, xlabel ylabel
```

Figure 2 is a histogram for the same data using 50 bins. The command was

```
. graph length, xlabel ylabel(0,.02,.04,.06) bin(50)
```

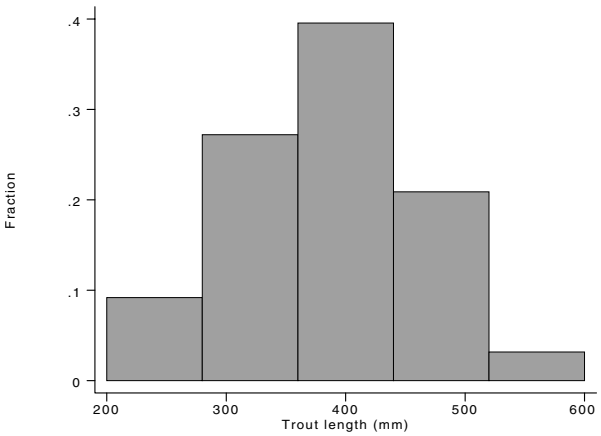

Figure 1: 5-bin histogram

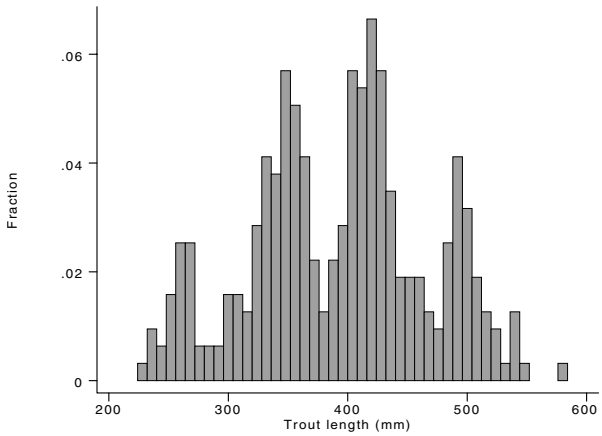

Figure 2: 50-bin histogram

These two graphs give very different impressions of the distribution of trout length. Figure 1 gives the impression that trout length follows a smooth, unimodal distribution, perhaps the normal or log-normal distribution. Figure 2, in contrast, is very irregular. It is not clear whether trout length follows a multimodal distribution or whether the narrowness of the bins highlights the noise in the data.

The differences between these two figures illustrate some of the problems with using histograms either to estimate a probability density function or to provide a good simple descriptive representation of the distribution of a batch. Fox (1990) identifies four distinct problems with the histogram:

1. The result is dependent on the origin x_0.

In constructing a histogram, the investigator must choose the position at which to place the origin of the bins. This decision is commonly made on the basis of convenience, with the bins beginning at or centered on "round" numbers. This element of choice in the construction of the histogram can interact with other choices, such as the number and width of bins, to give misleading pictures (Tarter and Kronmal 1976). Silverman (1986) and Fox (1990) present examples where changing the origin of a histogram significantly changes the impression given of the underlying distribution. For example, a change in the origin can change the apparent number of modes in the sample. To protect against this problem, it is recommended that several histograms with different origins be drawn. Unfortunately, this procedure can lead the analyst, intentionally or not, to select the histogram that best matches the analyst's prior notion of the underlying distribution.

2. The result is dependent of the width and number of bins.

As with the choice of origin, the choices of number and size of the bins can significantly affect the appearance of the histogram. Again, these choices can lead the researcher to bias, consciously or unconsciously, the presentation of data.

The number and width of bins determines the smoothness of the resulting display (Chambers et al. 1983, Silverman 1986). Tarter and Kronmal (1976) argue that the number of bins should be some function of the size of the sample. For large batches, a large number of bins gives a smooth representation of the unknown density function. Using only a few bins, however, eliminates any detail of the underlying distribution. On the other hand, choosing a large number of bins with a small data set produces a unidimensional scatterplot. Considering too few bins, though, produces a featureless picture. Frequently, the number of bins and their width are determined arbitrarily despite their importance.

More precise guides for the number and/or the width of the bins have been published, but rarely implemented: see Sturges (1926), Dixon and Kronmal (1965), Doane (1976), Velleman (1976), Scott (1979), Freedman and Diaconis (1981a, b); also see Emerson & Hoaglin (1983), Geiger (1991) and the Stata Reference Manual (Computing Resource Center, 1992).

3. The histogram is discontinuous, with jumps at the ends of the bins.

The histogram discontinuities are primarily a function of the arbitrary bin locations and the discreteness of the data rather than of the population that is sampled (Fox 1990). The local density is only computed at the midpoint of each bin, and then the bars are drawn assuming a constant density throughout each bin (Chambers et al. 1983). Silverman (1986) notes that the discontinuity of histograms causes difficulties if derivatives of the estimates are required.

4. The fixed bin width results in disproportional representation of density at the center and in the tails of the distribution.

If the bins are narrow enough to capture detail where density is high—in the center of the distribution—they may be too narrow to avoid noise where density is low—in the tails (Fox 1990). To address this problem, the bin widths can be varied (Silverman 1986). This is frequently done for the first and last bins of the histogram, which are generally constructed to contain all of the points below a certain value and all those above another value (Tarter and Kronmal 1976). However, the resulting histograms are subject to misinterpretation, since the height of a bar is no longer proportional to its area (Fox 1990).

Tarter and Kronmal (1976) argue that these problems make the histogram ill-suited for estimating the distribution of the underlying population, for inferential purposes, and for comparing the distribution of several populations. We now turn our attention to some refinements and extensions of the histogram that attempt to overcome these problems. The next sections present several of these methods along with ado-files to perform them.

Density traces

Some of the problems of the histogram derive from the technique of selecting the bins as a partition of the x-axis. Chambers et al. (1983) propose an alternative technique that mitigates these problems: they suggest computing the local density at every data point. In essence, fixed-width bins are constructed around each data point. These bins overlap where observations are clustered, thus the discontinuous appearance of the histogram is avoided. These local densities comprise the *density trace*.

Formally, for each data point x, we compute

$$\text{local density at } x = \frac{\text{number of observations in } [x-h, x+h]}{2h \times \text{total number of observations}}$$

The density trace can be defined for x that are not observed in the sample by passing a "window" through the range of the data that "smears" each data point across the interval $[x-h, x+h]$. Formally, the window is a weight function that assigns positive weight to each observed data point within the interval and zero weight outside the interval. A simple weight function is the "boxcar" weight function:

$$W(u) = \begin{cases} 1, & \text{if } |u| \leq 1/2; \\ 0, & \text{otherwise.} \end{cases}$$

This is a weight function with a rectangular shape, hence the name "boxcar." The density trace estimate at an arbitrary point x is the average of the smears of all x_i at x, that is,

$$\widehat{f}_T(x) = \frac{1}{2hn} \sum_{i=1}^{n} W\left(\frac{x - x_i}{2h}\right)$$

`boxdetra` calculates this boxcar-weighted density trace. The syntax of `boxdetra` is

`boxdetra` *x width dtrace*

where *x*, as before, is the variable being analyzed, *width* is the bin width ($= 2h$), and *dtrace* is the name of a new variable which, on output, contains the density estimate, the density trace.

Figure 3 displays a boxcar-weighted density trace of the coral-trout-length data. Figure 3 was produced by the following commands:

```
. boxdetra length 40 dtrace
  (output omitted)
. label variable dtrace "Density trace"
. graph dtrace length, xlab ylab(0,.002,.004,.006) c(l) s(.) twoway oneway box
```

In this example, the bin width ($2h$) is set at 40 mm. The density trace is calculated at each data point and the estimated densities are connected with straight lines. Boxplots and oneway scatterplots are displayed along the margins of the figure, permitting us to compare the information each of the graphical techniques conveys about the distribution of trout lengths. The density trace reveals four clearly defined modes, confirming the impression of the histogram with fifty bins. These modes are not revealed at all by the boxplot and are difficult to find in the oneway scatterplot.

One drawback of `boxdetra` is that it requires n `summarize` commands to obtain the density estimates. As a consequence, `boxdetra` may execute very slowly with a large data set. Chambers et al. (1983) suggest that, in practice, it is sufficient to calculate the density trace at a modest number of equally spaced points over the range of the data and to interpolate linearly between these points.

`boxdetr2` uses this approach in calculating the boxcar-weighted density trace. `boxdetr2` calculates the density trace at 50 points from the minimum value of x (226 mm) to the maximum (582 mm). The syntax of `boxdetr2` is

`boxdetr2` *x width dtrace midpt*

where *x*, *width*, and *dtrace* are defined as in `boxdetra` and *midpt* is a new variable that, on output, contains the 50 values of x at which the density trace is calculated. Figure 4 displays the density trace for the coral-trout-length data, again with a bin width of 40 mm.

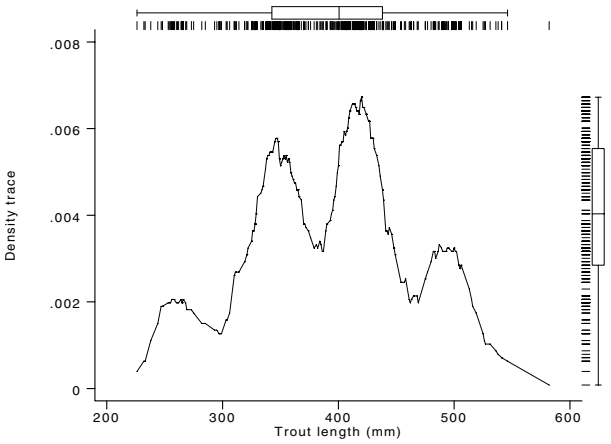

Figure 3: Density trace (all data points)

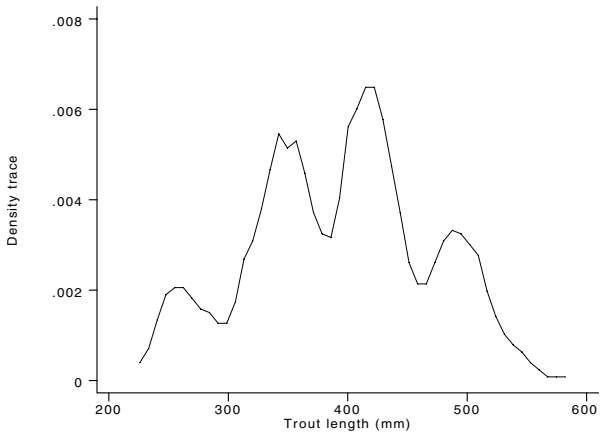

Figure 4: 50-point density trace

Figure 4 was produced by the commands:

```
. boxdetr2 length 40 dtrace2 midpt
  (output omitted)
. label variable dtrace2 "Density trace"
. label variable midpt "Trout length (mm)"
. graph dtrace2 midpt in f/50, xlab ylab c(l) s(.)
```

The estimate calculated by `boxdetr2` gives the same general impression as the estimate calculated by `boxdetra`, although the `boxdetr2` estimate is smoother.

These density traces eliminate the discontinuities at the boundaries of the histogram bins, but these estimates are still somewhat ragged. One reason for the raggedness is the rectangular shape of the boxcar weight function. To further smooth the density estimate, we can use a different weight function that varies gradually along the interval h. An example of such a function is the cosine weight function:

$$W(u) = \begin{cases} 1 + \cos 2\pi u, & \text{if } |u| < 1/2; \\ 0, & \text{otherwise.} \end{cases}$$

`cosdetra` calculates density traces using the cosine weight function. The syntax of `cosdetra` is

`cosdetra` *x width dtrace midpt*

As with `boxdetr2`, `cosdetra` calculates the density trace at 50 points spaced equally over the range of data. Figure 5 displays a cosine-weighted density trace of the coral-trout-length data.

The width of the weight function window ($2h$) also affects the smoothness of the density estimate. A wider window (larger value of $2h$) smooths the density trace by basing each value of $f(x)$ on more data points. Conversely, a narrower window (smaller value of $2h$) makes the density trace more ragged; since fewer data points are averaged into each value of $f(x)$, more data noise is retained.

Figure 6 displays the cosine-weighted density trace with $h = 20$, half the width of the density trace in Figure 5. The reduction in h gives Figure 6 a rougher appearance. The first and second modes now show a series of additional subcomponents, an additional component appears between the third and fourth modes, and additional modes appear in the right tail of the distribution.

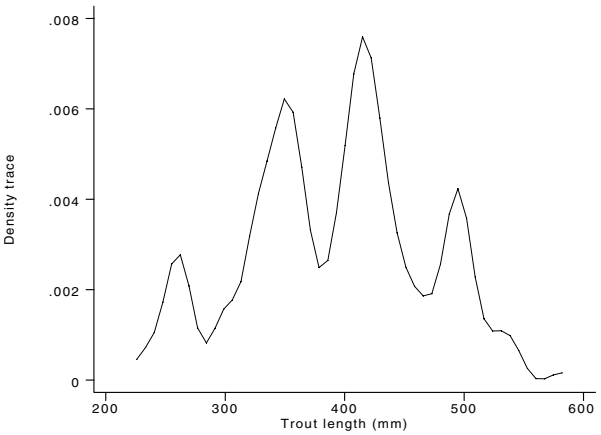

Figure 5: Cosine density trace (2h=40 mm)

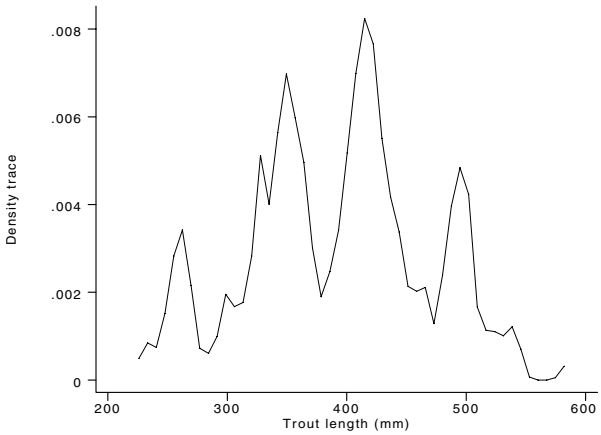

Figure 6: Cosine density trace (2h=20 mm)

A simple (naive) density estimator

Fox (1990), following Silverman (1986), motivates a density estimate with a rectangular (boxcar) weight function that provides a natural introduction to kernel density estimators. The density at a point x can be thought of as the limit of the height of a histogram bar centered at x as the half-width h of the bar goes to zero:

$$f(x) = \lim_{h \to 0} \frac{1}{2h} \Pr(x - h < X < x + h)$$

Thus a simple density estimator is one which replaces the probability in a small region (window) around x with the sample proportion, scaling the estimate so the total area under $f(x)$ integrates to unity:

$$\widehat{f}_S(x) = \frac{1}{2h} \times \frac{\#[x-h < X_i < x+h]}{n}$$

As with the boxcar-weighted density trace, this density estimator is similar to a histogram with bin width equal to $2h$, but with every point as the center of a bin, that is, with no fixed origin.

The weight function $S(\cdot)$ for this simple estimator is given by

$$\widehat{f}_S(x) = \frac{1}{nh} \sum_{i=1}^{n} S\left(\frac{x - X_i}{h}\right)$$

where

$$S(z) = \begin{cases} 1/2, & \text{if } |z| < 1; \\ 0, & \text{otherwise.} \end{cases}$$

This weight function is similar to, but slightly different from, the boxcar weight function calculated by `boxdetra` and `boxdetr2`.

`kernsim` calculates this simple density estimator. As before, the density is estimated at only 50 points to save computing time. Since this simple estimator integrates to one between $x_{(1)} - h$ and $x_{(n)} + h$, the 50 points are located (in a conventional way) from $x_{(1)} - h + (d/2)$ to $x_{(n)} + h + (d/2)$, where d is the interval defined by $\mathrm{range}/50$, and the range goes from $x_{(1)} - h$ to $x_{(n)} + h$. This convention is used by all the subsequent ado programs (except `adgakern`). The convenience of this definition is that these points (except the last) can be regarded as the midpoints of intervals in similar histograms for comparison and for Gaussian component determination and characterization.

The syntax of `kernsim` is

`kernsim` *x halfwidth density midpt*

where *x* is the variable to analyze, *halfwidth* is h, half the width of a bin, *density* is a new variable that, on output, contains the density estimates, and *midpt* is a new variable that contains the 50 points at which the density is estimated.

An estimate of the density of the coral-trout-length data (with $h = 20$) is displayed in Figure 7. Not surprisingly, this estimate is virtually identical to one obtained by `boxdetr2`, the boxcar-weighted density trace.

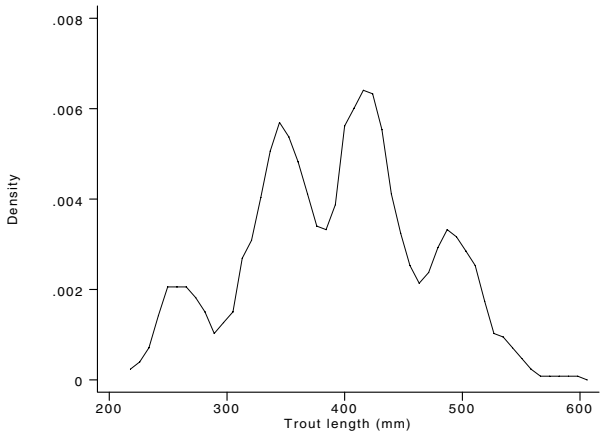

Figure 7: Simple (rectangular) density estimate (h=20 mm)

More sophisticated kernel estimators

A kernel density estimator is an estimator of the form of `kernsim` where the weight function $S(\cdot)$ is replaced by a kernel function $K[\cdot]$. A kernel function is a smooth, symmetric probability density integrating to unity:

$$\int_{-\infty}^{+\infty} K[z]dz = 1$$

In essence, a kernel estimator generates the density estimate by summing the kernels. Formally,

$$\widehat{f}_K(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left[\frac{x - X_i}{h}\right]$$

Using a smooth kernel eliminates some of the discontinuities caused by the square corners of the weight function used by the simple estimator above (Silverman 1986, Fox 1990).

Table 1, adapted from Silverman (1986), lists some kernel functions along with their efficiencies.

Table 1: Some kernels and their efficiencies

Kernal		*Efficiency*
Epanechnikov	$K[z] = \begin{cases} \frac{3}{4}(1 - \frac{1}{5}z^2)/\sqrt{5}, & \text{if } \lvert z\rvert < \sqrt{5}; \\ 0, & \text{otherwise} \end{cases}$	1
Biweight	$K[z] = \begin{cases} \frac{15}{16}(1 - z^2)^2, & \text{if } \lvert z\rvert < 1; \\ 0, & \text{otherwise} \end{cases}$	≈ 0.9939
Triangular	$K[z] = \begin{cases} 1 - \lvert z\rvert, & \text{if } \lvert z\rvert < 1; \\ 0, & \text{otherwise} \end{cases}$	≈ 0.9859
Gaussian	$K[z] = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$	≈ 0.9512
Rectangular	$K[z] = \begin{cases} 1/2, & \text{if } \lvert z\rvert < 1; \\ 0, & \text{otherwise} \end{cases}$	≈ 0.9295

To evaluate the performance of a kernel, it is necessary to consider the trade-off between variance and bias. The sum of the integrated variance and integrated bias is called the Mean Integrated (total) Squared Error (MISE). A good kernel function minimizes bias by assigning greater weight to observations close to the x value at which the density is being estimated (for example, by using a Gaussian function). Epanechnikov (1969) derived the maximally efficient (minimum MISE) kernel function named after him. The efficiencies listed in Table 1 for each kernel are their MISEs relative the MISE of the Epanechnikov kernel. As Silverman pointed out, based on MISE, there is very little to choose between the various kernels. Even the rectangular kernel function used by the simple density estimator has 93 percent relative efficiency. As a consequence, a kernel estimator can be chosen on the basis of other considerations such as the degree of differentiability or computational effort (Silverman 1986).

Implementations of the Epanechnikov and Gaussian kernel functions are provided on the STB-16 distribution diskette. These programs can easily be modified to calculate other kernel density estimators.

`kernepa` estimates the density using the Epanechnikov kernel function. The syntax of `kernepa` is

`kernepa` *x halfwidth density midpt*

where the arguments are defined as in `kernsim` above. Figure 8 displays an Epanechnikov kernel estimate of the density of the coral-trout-length data. Figure 8 is calculated using a half-width (h) of 20. This figure appears oversmoothed, although there are still indications that trout length follows a polymodal distribution. Decreasing h to 10 produces the density estimate shown in Figure 9, which gives almost the same impression as the "naive" estimator of Figure 7, with four clearly defined modes.

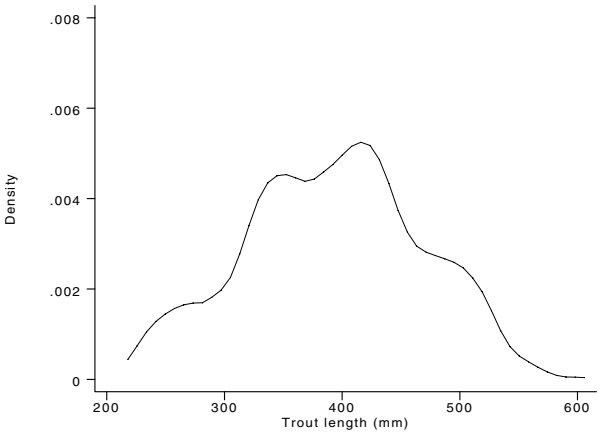

Figure 8: Epanechnikov kernel estimate (h=20 mm)

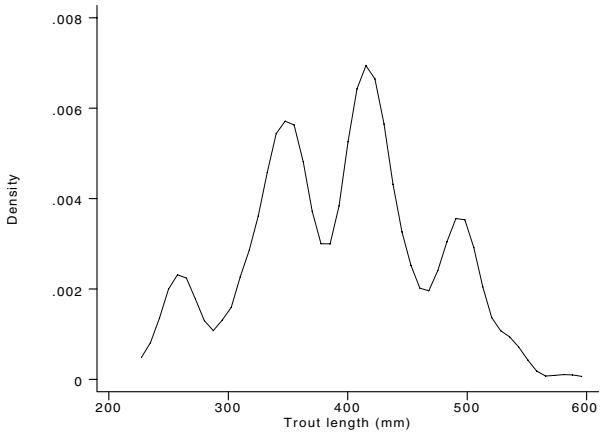

Figure 9: Epanechnikov kernel estimate (h=10 mm)

`kerngaus` calculates the Gaussian kernel density estimate. The syntax of `kerngaus` is the same as that of `kernepa`:

`kerngaus` *x halfwidth density midpt*

Figures 10 and 11 display two Gaussian kernel estimates of the density of the coral-trout-length data. Figure 10 sets h to 15 mm. Figure 11 reduces h to 5 mm. Both the noise and the detail increase as h becomes smaller, and in the latter figure it is possible to distinguish a more complex second component and an intermediate concentration between modes three and four. Some irregularities appear at the upper tail of the distribution including a lump around the maximal length observation (a mild outlier according to the boxplot of Figure 3). These irregularities may be artifacts of the fixed window width employed. Nevertheless, some of these structures seem to be meaningful in this particular case, suggesting some bimodal and multimodal age-groups (due to compound recruitment) and one age group overlapped by the third and fourth dominant age classes (clearly defined modes in Figure 11)—characteristics also identified by other methods used to estimate age and growth equations for these data (see for example Pauly 1988, Sparre et al. 1989).

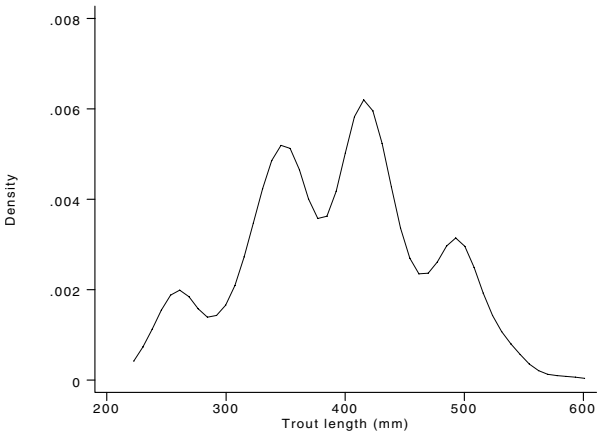

Figure 10: Gaussian kernel estimate (h=15 mm)

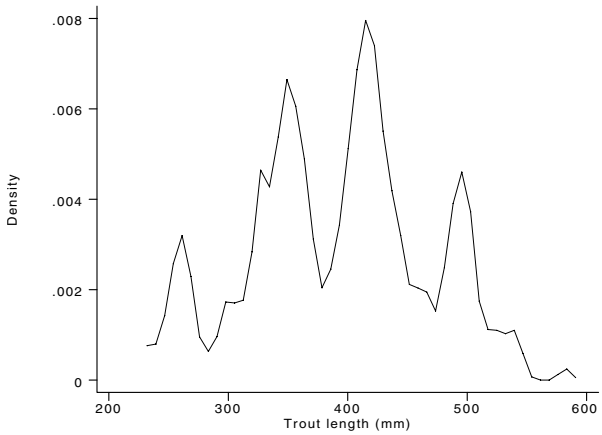

Figure 11: Gaussian kernel estimate (h=5 mm)

Adaptive kernel estimators

The kernel estimators described above all employ fixed window widths. This feature makes the estimates vulnerable to noise in the tails or any other low count interval of the distribution. Several adaptive methods have been proposed to address this problem. The idea is to reduce the window width in areas of high data densities and increase the window width in areas of low data densities. This adaptive procedure retains detail where observations concentrate and eliminates noise fluctuations where data are sparse. The algorithm to calculate this adaptive kernel estimator is taken from Fox, who adapted it from Silverman (1986). The steps are

1. Calculate a preliminary density estimate, for example, that provided by any of the fixed-window-width kernel functions, $\widehat{f}_K(x)$.
2. At each observation X_i, calculate a local window factor, w_i, that is inversely related to the density estimate:

$$w = \left(\frac{\tilde{f}_g}{\widehat{f}_K(X_i)} \right)^{1/2}$$

where

$$\tilde{f}_g \equiv \left(\prod_{i=1}^{n} \widehat{f}_K(X_i) \right)^{1/n}$$

is the geometric mean of the $\widehat{f}(X_i)$, and thus the w_i weights have a product and geometric mean of one.

3. Use the weights to calculate the adaptive-kernel estimator

$$\widehat{f}_A(x) = \frac{1}{nh} \sum_{i=1}^{n} \frac{1}{w_i} K \left[\frac{x - X_i}{w_i h} \right]$$

Note that applying the weights to h produces a varying-width window with a geometric mean half-width of h. The factor $1/w$ is required to ensure that the total area under the density estimate is unity.

4. Iterate steps 2 and 3, using $\widehat{f}_A$ in place of $\widehat{f}_K$. In practice, iteration produces little change in the estimated densities (Fox 1990).

The choice of window width

The choice of the window width h determines the qualitative features of a kernel density. One approach, suggested by Tarter and Kronmal (1976), is to vary h until a pleasing (usually smooth) figure results. This procedure relies on the subjective assessment of the researcher, but it may be adequate for exploratory purposes (Silverman 1986). Indeed, it is useful to compare several levels of smoothing, since important aspects of the density will "appear" and "disappear" as the window width changes (Silverman 1981a).

Statistical theory provides some guidance in this selection of an optimal window width. Unfortunately it is generally not possible to optimize the window width without previous knowledge of the shape of the true density. Of course, if this shape were known beforehand, there would be no estimation problem.

Following Tukey (1977), Scott (1979), and Silverman (1978, 1986), the Gaussian distribution can be employed as a reference standard in choosing h. Applying a Gaussian kernel and minimizing the MISE, the following scale estimate can be calculated (Fox 1990):

$$s = \min \left[\left(\frac{\sum (x_i - \bar{x})^2}{n-1} \right)^{1/2}, \frac{H \text{ spread}}{1.349} \right]$$

Then h can be chosen as

$$h = \frac{0.9s}{n^{1/5}}$$

s is the smaller of two estimates of the Gaussian distribution spread (scale) parameter σ: the classical, unbiased standard deviation and the robust F-pseudosigma based on the data fourth or hinge spread (Hoaglin 1983, Fox 1990). This adjustment provides resistance to heavy tails and will work well for a wide range of densities, but, as indicated by Silverman (1986), it tends to oversmooth highly skewed and multimodal distributions. If this is the case, this "optimal" window half-width can be considered as a starting point for subsequent fine tuning.

Examples of the adaptive kernel density estimator

`adgakern` calculates an adaptive Gaussian kernel density estimator using the algorithm described above without iteration. The syntax of `adgakern` is

`adgakern` *x halfwidth density*

The "optimal" window width for the coral-trout-length data was calculated according to the formula given above, using the `summarize` and `lv` (letter values) commands. For these data, $s = 74.0629$ and $F - \text{pseudosigma} = 70.8814$. Thus $h = 0.9(70.8814)/316^{1/5} = 20.18$. `adgakern` calculates an initial Gaussian kernel estimate using the specified h value, determines the w_i weights, and then calculates adaptive density values for each observation.

Figure 12 displays the adaptive kernel estimate of the coral-trout-length data. Figure 12 shows two clearly defined modes at the center and two apparently oversmoothed modes at the tails. Since `adgakern`, like `boxdetra`, calculates the density at every data point, it is possible to display the results in combination with the `oneway` and `box` graph options. In light of the previous figures, which document the multimodal distribution of these data, we conclude that $h = 20.18$ is too large and oversmooths the data.

Because of the many calculations required, `adgakern` executes very slowly. To save time, `adgaker2` evaluates the density only at 50 equally spaced points after steps 1 and 2 of the algorithm. The syntax of `adgaker2` is

`adgaker2` *x halfwidth density midpt*

where the arguments are familiar by now.

Figure 13 displays the output of `adgaker2` for the coral-trout-length data. The window half-width h was reduced to 15 mm to counteract the oversmoothing observed in Figure 12. In Figure 13, the separation and definition of the four modes is readily apparent. Because `adgaker2` adjusts the window half-width according to the number of observations in the neighborhood of each estimate, `adgaker2` is particularly useful for analyzing size frequency distributions. In contrast, fixed bin-width methods lose detail in high data concentration areas and become contaminated by noise in low data concentration areas. Another exploratory approach that applies nonlinear resistant smoothers to minimally grouped histograms is proposed in Salgado-Ugarte (1992) and Salgado-Ugarte and Curts-García (1992, 1993).

As a general practice, Fox (1990) suggests starting with "optimal" window half-width described above. Then this value can be adjusted downward to the smallest h that does not produce unacceptable roughness in the estimated density. Silverman (1986) provides additional guidelines and suggestions.

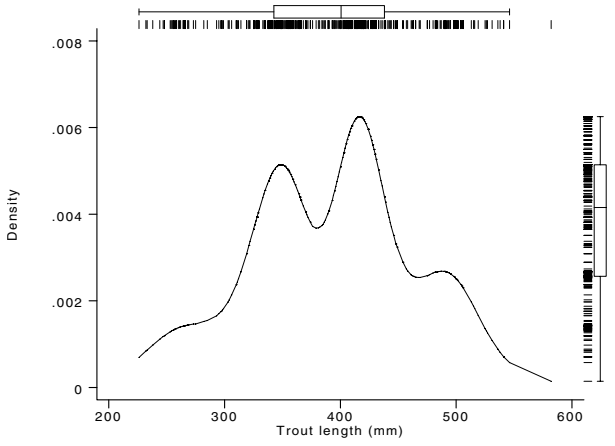

Figure 12: Adaptive kernel estimate (h=20.18 mm)

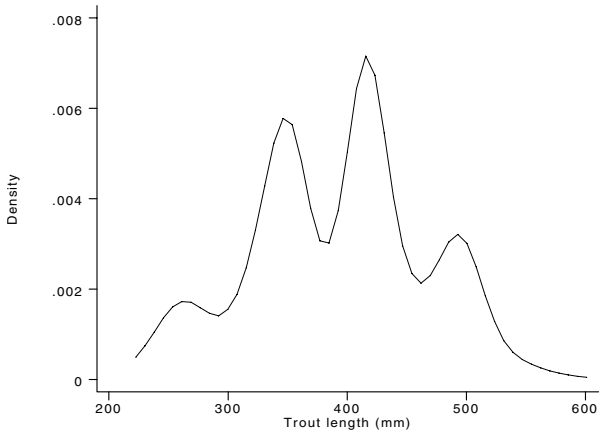

Figure 13: 50-point adaptive kernel estimate (h=15 mm)

Comments on program performance

Despite their simplicity, kernel density estimators were not proposed until 1956 in a paper by Rosenblatt. One reason for this delay is certainly the computational burden of calculating these estimates. Only when substantial computing power became available was it possible to undertake them. Indeed, the kernel estimators are well suited for computer implementation.

Surprisingly, though, these procedures are not included in any widely known computer package. Stata's programming language provides an adequate environment in which to program such procedures and to plot the results. Even in Stata, kernel density estimation programs tend to execute slowly. Table 2 reports some representative timing of each of the programs described in this insert. These are the times required to calculate density estimates for the coral-trout-length data ($n = 316$) on a 486 DOS computer using regular (not Intercooled) Stata.

Table 2: Execution times

(in minutes)

`boxdetra`	11
`boxdetr2`	3
`cosdetra`	3
`kernsim`	3
`kernepa`	4
`kerngaus`	3.5
`adgakern`	50
`adgaker2`	25

In our implementations, we incorporated some suggestions to reduce computations and accelerate execution. In all the programs using a Gaussian kernel (`kerngaus`, `adgakern` and `adgaker2`), the calculations are carried out considering only $|z| < 2.5$. In addition, most of the programs calculate the density at only 50 equally spaced points along the data range. If there are less than 50 observations in the data set, it is necessary to `set obs 50` before using any of these kernel density estimators.

Using these programs as a model, other different kernel functions can be programmed. The number of points estimated can be also modified if needed. Additional and alternative procedures as well as details are provided by Silverman (1986) and Fox (1990).

As a check, these ado-files were applied to data presented in Chambers et al. (1983) and in Fox (1990) (adapted (corrected) from Leinhardt and Wasserman (1979)). These data sets—`ozone.dta` and `infamora.dta`—are provided on the STB distribution diskette. As an additional check, the programs were rewritten in Pascal and applied to these data sets. These Pascal programs produced the same results (except for negligible rounding errors) as the Stata versions.

A final comment: How many modes?

The polymodal character of size-frequency data in Fisheries Sciences and Ecology is well known. It usually indicates mixed unimodal distributions. In this regard, the kernel density estimates provide several ways to test and evaluate multimodality. For details see Silverman (1981b, 1983, 1986). Other approaches have been suggested by Cox (1966) and Good and Gaskins (1980).

The kernel density estimates obtained by the programs presented in this insert may serve as an initial point to attempt estimating the parameters of each of the underlying components in the mix. We hope to present our programs designed to achieve this aim in a later issue of the STB.

Acknowledgements

The first author is grateful to the Ministry of Education, Science and Culture of Japan and to the National Autonomous University of Mexico (FES Zaragoza and DGAPA) for their support.

References

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

Computing Resource Center. 1992. *Stata Reference Manual: Release 3*, vol. 1. 5th ed., 185–187 Santa Monica, CA.

Cox, D. R. 1966. Notes on the analysis of mixed frequency distributions. *The British Journal of Mathematical and Statistical Psychology* 19: 39–47.

Dixon, W. and R. A. Kronmal 1965. The choice of origin and scale for graphs. *Journal of the Association for Computing Machinery* 12: 259–261.

Doane, D. P. 1976. Aesthetic frequency classifications. *The American Statistician* 30: 181–183.

Emerson J. D. and D. C. Hoaglin 1983. Stem-and-leaf displays. In *Understanding Robust and Exploratory Data Analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 7–30. New York: John Wiley & Sons.

Epanechnikov, V. A. 1969. Nonparametric estimation of a multidimensional probability density. *Theor. Probab. Appl.* 14: 153–158.

Fox, J. 1990. Describing univariate distributions. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 58–125. Newbury Park, CA: Sage Publications.

Freedman, D. and P. Diaconis 1981a. On the histogram as a density estimator: L_2 theory. *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete* 57: 453–476.

——. 1981b. On the maximum deviation between the histogram and the underlying density. *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete* 58: 139–167.

Geiger, P. 1991. gr1: Enhancing visual display using stem-and-leaf. *Stata Technical Bulletin* 1: 8–9. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 32.

Goeden, G. B. 1978. A monograph of the coral trout, *Plectropomus leopardus* (Lacépède). *Res. Bull. Fish. Serv. Queensl.* 1: 42 p.

Good, I. J. and R. A. Gaskins 1980. Density estimation and bump-hunting by the penalized likelihood method exemplified by scattering and meteorite data. *Journal of the American Statistical Association* 75: 42–73.

Hoaglin, D. C. 1983. Letter values: A set of selected order statistics. In *Understanding Robust and Exploratory Data Analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 33–57. New York: John Wiley & Sons.

Leinhardt, S. and S. S. Wasserman 1979. Exploratory data analysis: An introduction to selected methods. In *Sociological Methodology*, ed. K. F. Schuessler. San Francisco: Jossey-Bass.

Pauly, D. 1988. Fisheries research and the demersal fisheries of Southeast Asia. In *Fish Population Dynamics: The Implications for Management*, ed. J. A. Gulland, 329–348. Chichester: John Wiley & Sons.

Rosenblatt, M. 1956. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics* 27: 832–837.

Salgado-Ugarte, I. H. 1992. El análisis exploratorio de datos biológicos. Fundamentos y aplicaciones. ENEP Zaragoza UNAM & Marc Ediciones. México: 89–120; 213–233.

Salgado-Ugarte, I. H. and J. Curts-García. 1992. sed7: Resistant smoothing using Stata. *Stata Technical Bulletin* 7: 8–11. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 99–103.

——. 1993. sed7.2: Twice reroughing procedure for resistant nonlinear smoothing. *Stata Technical Bulletin* 11: 14–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 108–111.

Scott, D. W. 1979. On optimal and data-based histograms. *Biometrika* 66: 605–610.

Silverman, B. W. 1978. Choosing the window width when estimating a density. *Biometrika* 65: 1–11.

——. 1981a. Density estimation for univariate and bivariate data. In *Interpreting Multivariate Data*, ed. V. Barnett, 37–53. Chichester: John Wiley & Sons.

——. 1981b. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society*, Series B, 43: 97–99.

——. 1983. Some properties of a test for multimodality based on kernel density estimates. In *Probability, Statistics and Analysis*, ed. J. F. C. Kingman and G. E. H. Reuter: 248–259. Cambridge: Cambridge University Press.

——. 1986. *Density Estimation for Statistics and Data Analysis.* London: Chapman & Hall.

Sparre, P., E. Ursin, and S. C. Venema. 1989. *Introduction to Tropical Fish Stock Assessment. Part 1. Manual.* FAO Fisheries Technical Paper. 306.1. Rome, FAO: 57–123.

Sturges, H. A. 1926. The choice of a class interval. *Journal of the American Statistical Association* 21: 65–66.

Tarter, M. E. and R. A. Kronmal 1976. An introduction to the implementation and theory of nonparametric density estimation. *The American Statistician* 30: 105–112.

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

Velleman, P. F. 1976. Interactive computing for exploratory data analysis I: display algorithms. *1975 Proceedings of the Statistical Computing Section.* Washington, DC: American Statistical Association.

Wegman, E. J. 1972a. Nonparametric probability density estimation: I. A summary of available methods. *Technometrics* 14: 533–546.

——. 1972b. Nonparametric probability density estimation: II. A comparison of density estimation methods. *Journal of Statistical Computation and Simulation* 1: 225–245.

Wilk, M. B. and R. Gnanadesikan. 1968. Probability plotting methods for the analysis of data. *Biometrika* 55(1): 1–17.

ssi5	Equation solving by bisection

William Gould, Stata Corporation, FAX 409-696-4601

The syntax of `bisect` is

`bisect` *fcnmask* `=` exp_y `from` exp_0 `to` exp_1 [`tol` exp_t]

where exp_y, exp_0, exp_1, and exp_t are expressions (but are typically specified as numbers) and where *fcnmask* is either

(1) an expression containing `X`
(2) *progname* [*args*] `X` [*args*] `returns` {`exp`|`macro`} *name*

`tol 1e-6` is assumed if `tol` is not specified.

Description

`bisect` finds the value of x such that $f(x) = exp_y$ or, more precisely

$$|f(x) - exp_y| < \epsilon$$

where $\epsilon = exp_t \max(|exp_y|, 1)$. The search is carried out over the range $exp_0 \leq x \leq exp_1$. The function $f()$ may be specified on the command line (first syntax for *fcnmask*) or as a user-written program (second syntax).

Example 1

Find the value of χ^2 with 2 degrees of freedom that is just significant at the 5% level.

Stata does not provide the inverse χ^2 function that could directly answer this question, but it does provide a `chiprob(`d`,`x`)` function that returns the reverse cumulative for χ^2 value x with d degrees of freedom. If you type '`display chiprob(2,3)`', Stata will respond with `.22313017`, meaning a χ^2 of 3 with 2 degrees of freedom is significant at the 22% level. '`display chiprob(2,8)`' results in `.01831564`, meaning the 1.8% level. The answer for x lies somewhere between 3 and 8. Thus, we want to find x such that `chiprob(2,`x`)` = .05 and we can look for x in the range 3 to 8:

```
. bisect chiprob(2,X)=.05 from 3 to 8
           Find  chiprob(2,X)=f() == c=.05, |f()-c|<1.000e-06
                lower                 upper
iteration       bound     f()-c       bound     f()-c     midpoint     f()-c
------------------------------------------------------------------------------
       0.           3  .1731302           8 -.0316844          5.5  .0139279
       1.         5.5  .0139279           8 -.0316844         6.75 -.0157819
       2.         5.5  .0139279        6.75 -.0157819        6.125 -.0032294
       3.         5.5  .0139279       6.125 -.0032294       5.8125  .0046804
       4.      5.8125  .0046804       6.125 -.0032294      5.96875  .0005711
       5.     5.96875  .0005711       6.125 -.0032294     6.046875 -.0013662
       6.     5.96875  .0005711    6.046875 -.0013662     6.007813  -.000407
       7.     5.96875  .0005711    6.007813  -.000407     5.988281  .0000796
       8.    5.988281  .0000796    6.007813  -.000407     5.998047 -.0001643
       9.    5.988281  .0000796    5.998047 -.0001643     5.993164 -.0000425
      10.    5.988281  .0000796    5.993164 -.0000425     5.990723  .0000186
      11.    5.990723  .0000186    5.993164 -.0000425     5.991943  -.000012
      12.    5.990723  .0000186    5.991943  -.000012     5.991333  3.29e-06
      13.    5.991333  3.29e-06    5.991943  -.000012     5.991638 -4.34e-06
                                                         5.991486 -5.26e-07
```

The answer is 5.991486, shown in the last row under midpoint.

The −5.26e–07 printed to the right of the solution means that `chiprob(2, 5.991486)` $= .05 - 5.26 \times 10^{-7}$. `bisect` stopped iterating when the difference between the desired and obtained answer fell below 10^{-6}. If we would be satisfied with an answer yielding a result accurate to 10^{-4}, we would type

```
. bisect chiprob(2,X)=.05 from 3 to 8 tol 1e-4
            Find  chiprob(2,X)=f() == c=.05, |f()-c|<.0001
                lower                 upper
iteration       bound     f()-c       bound     f()-c     midpoint     f()-c
------------------------------------------------------------------------------
       0.           3  .1731302           8 -.0316844          5.5  .0139279
       1.         5.5  .0139279           8 -.0316844         6.75 -.0157819
       2.         5.5  .0139279        6.75 -.0157819        6.125 -.0032294
       3.         5.5  .0139279       6.125 -.0032294       5.8125  .0046804
       4.      5.8125  .0046804       6.125 -.0032294      5.96875  .0005711
       5.     5.96875  .0005711       6.125 -.0032294     6.046875 -.0013662
       6.     5.96875  .0005711    6.046875 -.0013662     6.007813  -.000407
                                                         5.988281  .0000796
```

Similarly, if we wanted a more accurate answer, we could specify a smaller tolerance.

Example 2

Let us repeat example 1 but undertake a slightly different solution. This time, we will write a program to calculate the function. Our program is

```
program define mychi
        global S_1 = chiprob(2,`1´)
end
```

The `` `1´ `` in our program means substitute the first argument here. Typing '`mychi 3`', for instance, will store the evaluation of `chiprob(2,3)` in the global macro `$S_1`. This time, to obtain the solution, we type

```
. bisect mychi X returns macro S_1 =.05 from 3 to 8
  (output omitted)
```

The output we see will be the same as in our first example. Let us decipher the `bisect` command. "`mychi X`" tells `bisect` that the way to evaluate the function is to issue the command `mychi` followed by the number at which the function is to be evaluated—the `X` is a placeholder. "`returns macro S_1`" tells `bisect` that `mychi` is not a Stata function but a user-written program and that it returns the results of its calculation in the macro `$S_1`. The rest of the command is just as previously; "`from 3 to 8`" says to search the range 3 to 8.

We could have written our program to take two arguments:

```
program define mychi
        global S_1 = chiprob(`1´,`2´)
end
```

We would then obtain the solution to our problem by typing

```
. bisect mychi 2 X returns macro S_1 =.05 from 3 to 8
  (output omitted)
```

Had we also written our function to return its calculation in the scalar `answer` rather than the macro `$S_1`,

```
program define mychi
        scalar answer = chiprob(`1´,`2´)
end
```

we would type

```
. bisect mychi 2 X returns exp answer = .05 from 3 to 8
```

to obtain the solution.

Example 3

Consider the recursive definition:

$$S_t = \left[\left(\frac{a_{t-1}}{\sqrt{S_{t-1}}} - 2\right)\left(\frac{1.1}{a_t}\right)\right]^{-2}$$

Assume $a_1 = 1.5$ and $a_t = 1$, $t = 2, \ldots$. A program to calculate S_t, $t = 2, \ldots, 19$, given S_1 is

```
program define simul
        quietly {
                drop _all
                set obs 19
                gen a = cond(_n==1,1.5,1)
                gen S=`1´ in 1
                replace S=((a[_n-1]/sqrt(S[_n-1])-2)*(1.1/a))^-2 in 2/l
        }
end
```

With this program, typing '`simul .1`' would create the series based on $S_1 = .1$:

```
. simul .1
. list
             a          S
  1.       1.5         .1
  2.         1   .1098073
  3.         1   .7978578
  4.         1   1.066079
  5.         1   .7767599
 (output omitted)
 19.         1   .5265746
```

Now consider the following problem: Find the value of S_1 such that $S_{19} = 1/22^2$. Solving this problem analytically is difficult. With `bisect`, the problem is easy:

```
. bisect simul X returns exp S[19] = 1/(22^2) from .1 to .5
          Find  simul X =f() == c=.00206612, |f()-c|<1.000e-06
                lower                 upper
iteration       bound     f()-c       bound     f()-c     midpoint     f()-c
------------------------------------------------------------------------------
       0.          .1  .5245085          .5  .2487174
range does not bound solution
r(409);
```

One requirement `bisect` does make is that the range you specify for the bisect contains the solution. We specified a range that is too narrow.

```
. bisect simul X returns exp S[19] = 1/(22^2) from .001 to .75
          Find  simul X =f() == c=.00206612, |f()-c|<1.000e-06
                lower                 upper
iteration       bound     f()-c       bound     f()-c     midpoint     f()-c
------------------------------------------------------------------------------
       0.        .001 -.0020287         .75  .2133531        .3755  .2503633
       1.        .001 -.0020287       .3755  .2503633       .18825  33.91137
       2.        .001 -.0020287      .18825  33.91137      .094625  .2211156
       3.        .001 -.0020287     .094625  .2211156     .0478125  2593.315
       4.        .001 -.0020287    .0478125  2593.315     .0244063  11117.92
       5.        .001 -.0020287    .0244063  11117.92     .0127031  857.6943
       6.        .001 -.0020287    .0127031  857.6943     .0068516  5.248867
       7.        .001 -.0020287    .0068516  5.248867     .0039258 -.0011359
       8.    .0039258 -.0011359    .0068516  5.248867     .0053887  .0035936
       9.    .0039258 -.0011359    .0053887  .0035936     .0046572  .0000211
      10.    .0039258 -.0011359    .0046572  .0000211     .0042915 -.0006933
      11.    .0042915 -.0006933    .0046572  .0000211     .0044744 -.0003811
      12.    .0044744 -.0003811    .0046572  .0000211     .0045658 -.0001931
      13.    .0045658 -.0001931    .0046572  .0000211     .0046115 -.0000896
      14.    .0046115 -.0000896    .0046572  .0000211     .0046344 -.0000352
      15.    .0046344 -.0000352    .0046572  .0000211     .0046458 -7.31e-06
      16.    .0046458 -7.31e-06    .0046572  .0000211     .0046515  6.81e-06
                                                         .0046487 -2.59e-07
```

$S_0 = .00465$ produces $S_{19} = 1/22^2$.

Saved Results

`bisect` saves the solution in the global macro `$S_1`.

Methods and Formulas

`bisect` engages in bisection for the solution. This is a variation of the guess-the-number game. I'm thinking of a number between 0 and 100; you make guesses and I will tell you if you are correct or whether the number is higher or lower. Your best strategy is to guess the midpoint of the range, so you should make an opening guess of $(100+0)/2 = 50$. I say higher. You now know the number lies in the range 50 to 100, so you guess $(50+100)/2 = 75$. I say lower. You know the number is bounded by 50 and 75, so you guess $(50+75)/2 = 62.5$. And thus does the game continue until you guess the right number.

To translate this to solving $f(x) = c$ for x, let x_0 and x_1 be two values of x, $x_0 < x_1$, such that $f(x_0) \leq c$ and $f(x_1) \geq c$, or that $f(x_0) \geq c$ and $f(x_1) \leq c$. If $f()$ is continuous, there must be at least one value x^*, $x_1 \leq x^* \leq x_2$ such that $f(x^*) = c$. `bisect` checks that $f(x_0)$ and $f(x_1)$ lie on either side of c and, if not, issues the error message that the range does not bound the solution. It also checks that $x_0 < x_1$ and interchanges them if not.

`bisect` then calculates $x_m = (x_0 + x_1)/2$. Either $f(x_m) = c$ (or is close enough to c to call it equal) and we have a solution, or it is not. If it is not, we can halve the width range in which the solution is known to lie by resetting either x_0 or x_1 as appropriate. We can then repeat the process. (The resetting logic is as follows: First, assume $f(x_0) < c$ and $f(x_1) > c$. If $f(x_m) < c$, the lower limit x_0 is reset to x_m; otherwise, the upper limit x_1 is reset. Now take the alternative case, $f(x_0) > c$ and $f(x_1) < c$. If $f(x_m) < c$, the upper limit x_1 is reset and otherwise we reset the lower limit.)

Bisection is not the necessarily fastest way to obtain a solution to $f(x) = c$, but it is relentless and will find a solution as long as the function is continuous, which means only that the function is not disjoint; the function need not be smooth or in any other way well behaved. Bisection is not, unfortunately, a panacea:

1. To obtain x such that $f(x) = c$, you must specify two numbers x_0 and x_1 that evaluate to results on either side of c. `bisect` makes the assertion that the "range does not bound the solution" if not. The message is carefully worded—the range does not bound the solution, but it very well may contain a solution. Think of the parabola $f(x) = x^2 - 4$; $x = \pm 2$ are solutions for $f(x) = 0$. Yet, if you perform bisection over the range -5 to 5, you will be told that the range does not bound the solution because $f(-5) = 21 > 0$ and $f(5) = 21 > 0$. In this case, you must specify a narrower range, not a wider one (say -5 to 0 or 0 to 5).
2. Even when the range does bound a solution, that does not mean the range does not include more than one solution, and bisection will never discover that fact. Consider the cubic $f(x) = (x-1)(x-2)(x-3)$, which has roots at 1, 2, and 3. A bisection search over the range 0 to 5 uncovers the root at 3; -1 to 5 the root at 2; and -2 to 5 the root at 1.

For a further discussion of the bisection method, see Press et al. (1992, 350–354).

References

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

ssi5.1	Graphing functions

William Gould, Stata Corporation, FAX 409-696-4601

The syntax of `fcnplot` is

`fcnplot` *fcnmask* `from` *exp*$_0$ `to` *exp*$_1$ [`obs` *exp*$_o$] [`, replace slow` *graph_options*]

where *exp*$_0$, *exp*$_1$, and *exp*$_o$ are expressions (but are typically specified as numbers) and where *fcnmask* is either

(1) an expression containing `X`
(2) *progname* [*args*] `X` [*args*] `returns` {`exp` | `macro`} *name*

If `obs` is not specified, `obs 101` is assumed if *fcnmask* is specified using the first syntax and `slow` is not specified; otherwise, `obs 21` is assumed. `obs` may be specified as any integer between 2 and 350.

Description

`fcnplot` graphs the specified function—which may be specified by a user-written program and so quite complicated—and optionally leaves behind the data set of values of x and $f(x)$ just graphed.

Options

`replace` specifies that the data just graphed, x and $f(x)$, be left in memory at the conclusion of the command. If `replace` is not specified, the original data in memory, if any, remains unchanged. If `replace` is specified, the original data is discarded and a data set with variables `x` and `y` is left in its place. `replace` may not be abbreviated.

`slow` is an option only when *fcnmask* is specified using the first syntax—an expression containing `X`. If the second syntax is used, `slow` is the default and therefore need not be specified. Say the first syntax is used and `slow` is not specified. Then `fcnplot` preserves the original data (if necessary, see `replace` above), discards that data, and generates x and $f(x)$ with two `generate` statements. The data is graphed and finally the original data is restored if `replace` was not specified.

In `slow` mode, the data is preserved (if necessary) but it is then left in place during the calculation of the $f(x)$. The function or user-written program is asked to calculate function values one at a time. The function or user-written program can make use of the existing data in memory or even change it if it so desires. Once all the function values have been calculated, the data is discarded (remember, the original was preserved if `replace` was not specified), the values of x and $f(x)$ loaded, the graph drawn, and then, if `replace` was not specified, the original data restored.

graph_options refers to any of the options of the `graph, twoway` command.

Example 1

You wish to draw a graph of `chiprob(2,`x`)` over $0 \leq x \leq 10$. You type

```
. fcnplot chiprob(2,X) from 0 to 10
```

See Figure 1. Any data you have in memory remains unchanged. If you typed

```
. fcnplot chiprob(2,X) from 0 to 10, replace
```

any data you have in memory would be discarded and left behind would be the points plotted:

```
. describe
Contains data
  Obs:   101 (max=  5101)                  y= chiprob(2,x)
 Vars:     2 (max=    99)
Width:     8 (max=   200)
  1. x            float  %9.0g
  2. y            float  %9.0g              chiprob(2,x)
Sorted by:
Note:  Data has changed since last save
. list
             x          y
  1.         0          1
  2.        .1   .9512295
  3.        .2   .9048374
  (output omitted)
100.       9.9   .0070834
101.        10   .0067379
```

You could then redraw the graph just shown by `fcnplot` by typing

```
. graph y x, c(l) s(o)
```

or you might make a dressier version of the graph by typing

```
. graph y x, c(l) s(i) ylab xlab border
```

You could have made the dressier version at the outset, with or without the `replace` option:

```
. fcnplot chiprob(2,X) from 0 to 10, s(i) ylab xlab border
```

The `c(l)` option is not necessary (but you could specify it) because `fcnplot` by default knows you want to connect the points; see [3] connect.

Example 2

Not all functions can be specified on the command line. Some are simply too complicated to be expressed on a single line—you need to write a Stata program. We will consider such a function in the third example below. To begin, however, let us write a program to calculate a simple function: $y = \exp(-x/6)\sin(x)$.

```
program define myfcn
        global S_1 = exp(-`1´/6)*sin(`1´)
end
```

To graph this function over the range 0 to 4π, you type

```
. fcnplot myfcn X returns macro S_1 from 0 to 4*_pi
```

In Figure 2, we show a version with a few options to make the graph look better:

```
. fcnplot myfcn X returns macro S_1 from 0 to 4*_pi obs 101, s(i) yline(0) noaxis
```

The "`s(i) yline(0) noaxis`" are standard `graph` options. Note the "`obs 101`", however, before the comma. Had we not specified this, the graph would have been drawn using 21 points over the range 0 to 4π. Functions implemented as user-written programs take longer to evaluate than functions that are built into Stata's expression parser (such as `chiprob()`). `fcnplot` tries to be speedy, so when you specify the function using a user-written program, it changes the number of points it calculates by default from 101 to 21. Specifying "`obs 101`" forces `fcnplot` to use 101 points, making a smoother-looking graph.

Our program does not have to return the result in a macro. If the result is returned in any other way, it is said to be returned in an expression. For instance, saving the result in the scalar `answer`,

```
program define myfcn
        scalar answer = exp(-`1´/6)*sin(`1´)
end
```

Figure 2 would be drawn by typing

```
. fcnplot myfcn X returns exp answer from 0 to 4*_pi obs 101, s(i) yline(0) noaxis
```

As a final note, you might be tempted to think that $y = \exp(-x/6)\sin(x)$ could have been drawn by typing

```
. fcnplot exp(-X/6)*sin(X) from 0 to 4*_pi
```

Unfortunately, it could not. Although the function is "simple," `X` appears in it more than once. When a function is specified on the `fcnplot` command line, `X` can appear only once.

Example 3

Consider the recursive definition:

$$S_t = \left[\left(\frac{a_{t-1}}{\sqrt{S_{t-1}}} - 2\right)\left(\frac{1.1}{a_t}\right)\right]^{-2}$$

Assume $a_1 = 1.5$ and $a_t = 1$, $t = 2, \ldots$. Graph the values of S_{19} as a function of S_1 over the range .001 to .50.

The first step in drawing this graph is to write a program to calculate S_t, $t = 2, \ldots, 19$, given S_1. Here is such a program:

```
program define simul
        quietly {
                drop _all
                set obs 19
                gen a = cond(_n==1,1.5,1)
                gen S=`1´ in 1
                replace S=((a[_n-1]/sqrt(S[_n-1])-2)*(1.1/a))^-2 in 2/l
        }
end
```

With this program, typing '`simul .1`' would create the series based on $S_1 = .1$:

```
. simul .1
. list
             a          S
  1.       1.5         .1
  2.         1   .1098073
  3.         1   .7978578
  4.         1   1.066079
  5.         1   .7767599
 (output omitted)
 19.         1   .5265746
```

With this program, we can now draw the graph:

```
. fcnplot simul X returns exp S[19] from .001 to .5
```

The result is shown in Figure 3. We will be better able to see the details of the function if we graph it on a log y axis. While we are at it, we will add more points to the graph:

```
. fcnplot simul X returns exp S[19] from .001 to .5 obs 101, ylog
```

The result is shown in Figure 4.

Final comment

Since `fcnplot` follows the same syntax as `bisect`, it can provide a visual check for the problems discussed with `bisect` (Gould 1993, 23).

References

Gould, W. 1993. ssi5: Equation solving by bisection. *Stata Technical Bulletin* 16: 20–23.

Figures

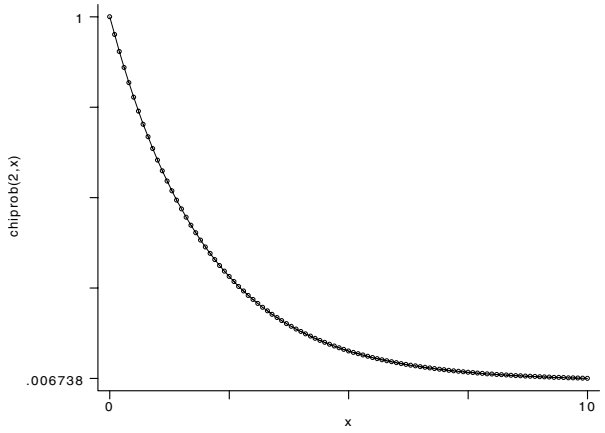

Figure 1

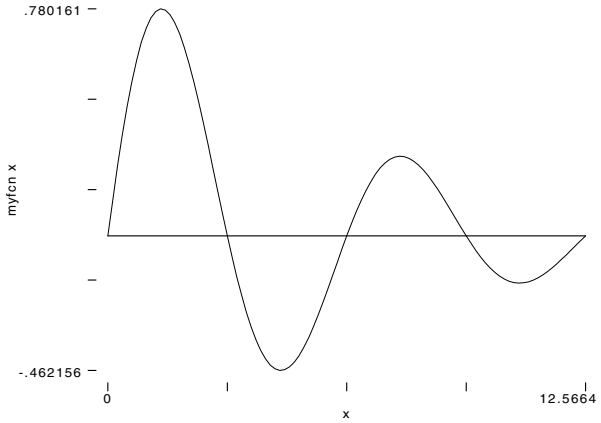

Figure 2

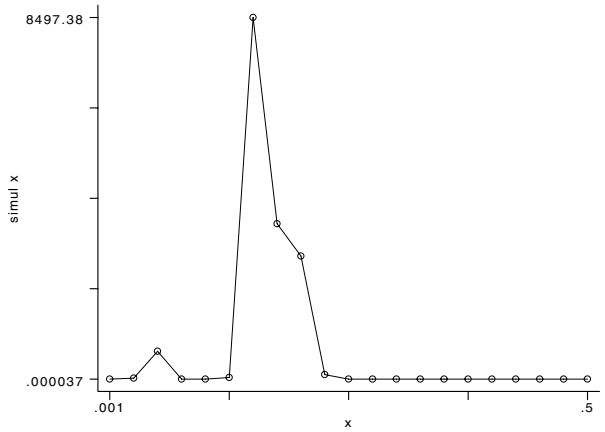

Figure 3

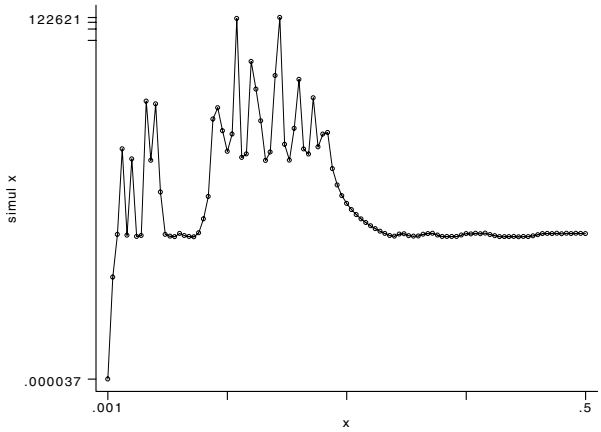

Figure 4

sts4.1	More on time series regression

Sean Becketti, Stata Technical Bulletin, FAX 913-888-6708

In *sts4* (STB-15), I presented a suite of programs for time series regression. Unfortunately, there was a small, but crucial, error in one of the ado-files on the STB-15 disk. The `program define` statement was missing from `_addl.ado`. You can either add this line to `_addl.ado` (make it the third line, after the two comment lines) or copy the corrected program from the STB-16 disk. I would like to thank Jon S. Ebeling of Chico, California who was the first one to contact me about this problem.

Also, in *sts4* I promised to publish additional time series programs in this STB. In particular, I promised to publish a program for dynamic forecasts from a time series regression. I have delayed the publication of those programs to make space for articles by other authors. These programs will be published in a later issue.

Reference

Becketti, S. 1993. A suite of programs for time series regression. *Stata Technical Bulletin* 15: 20–28.

zz3	Computerized index for the STB

William Gould, Stata Corporation, FAX 409-606-4601

Through STB-15, some 266 inserts have been published in the STB, which is to say that the STB is now old enough that finding a particular insert you only partially recollect, or finding all the inserts on a particular subject, is becoming tedious. The automated system presented below is an attempt to alleviate the difficulty.

The system is called (too gloriously) the STBinformer, the command is `stb`, and the syntax is just that: `stb`.

```
. stb

  ___  ____  ____
 /__    /   /___/
___/   /   /___/   informer
---------------------------

(indexes for STB-1 through STB-15 found)
--Top level--   Enter search specification, ?, ??, or end -> . _
```

The STBinformer allows you to find inserts based on the insert identifier (such as *zz3*), the title ("Computerized index for the STB"), the author (Gould), a word appearing in the title ("index" or "STB"), or a file associated with it (such as `stb.ado`).

For example (but not very useful), if we wanted to find out about *sg11.2*, we could type `sg11.2` to the prompt:

```
--Top level--   Enter search specification, ?, ??, or end -> . sg11.2
------------------------------------------------------------------------------
STB-13 sg11.2    Quantile regression standard errors              W. H. Rogers
                 There is no software associated with sg11.2
------------------------------------------------------------------------------

--Top level--   Enter search specification, ?, ??, or end -> . _
```

The STBinformer can produce more useful listings based on more reasonable search specifications. The search can result in no matches (in which case you can try again), in one match (as above), or in multiple matches. In the latter case, you may review the list and you may select matches one at a time to find out about the software (filenames) associated with each.

Specifying searches

When we typed `sg11.2` in response to the top-level request to enter a search specification, the `sg11.2` was taken to be an insert number. In answer to the top question, you can type

form	example	explanation
insert	`sg11.2`	look up a particular insert
partialinsert`*`		look for all inserts starting with *partialinsert*
	`s*`	all starting with `s` (statistics)
	`sg*`	all starting with `sg` (general statistics)
	`sg11*`	all starting with `sg11` (related to *sg11*)
`/`*word*		look for all inserts with *word* in title
	`/errors`	all with `errors` in title (capitalization irrelevant)
`\`*author*		look for all insert written by *author*
	`\royston`	all written by Royston (capitalization irrelevant)
`=`*filename*		all supplying *filename*
	`=arctic.dta`	all supplying `arctic.dta`
	`=pause.ado`	all supplying `pause.ado`
	`=pause`	all supplying filenames starting with `pause` (includes `pause.ado`)
	`=*.dta`	all supplying filenames ending in `.dta` (data sets)

Related insert search

We started by listing *sg11.2*, let us now find all the sg11 inserts:

```
--Top level--   Enter search specification, ?, ??, or end -> . sg11*
--Top Level--   --3 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as sg11.1 (see detail on insert)
         -> . _
```

To our query for sg11*, meaning all inserts that start with sg11, we are told that 3 matches were found. The matches are not, however, listed. We can choose to list the matches by typing '1' or '2':

```
         -> . 1
-------------------------------------------------------------------------------
STB-9  sg11      Quantile regression standard errors               W. H. Rogers
STB-9  sg11.1    Quantile regression bootstrapped standard errors      W. Gould
STB-13 sg11.2    Quantile regression standard errors               W. H. Rogers
-------------------------------------------------------------------------------
--Top Level--   --3 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as sg11.1 (see detail on insert)
         -> . _
```

Having listed the matches, we can now specify one for further examination:

```
         -> . sg11.1
-------------------------------------------------------------------------------
STB-9  sg11.1    Quantile regression bootstrapped standard errors      W. Gould
                 .ado and .hlp files:
                     _bsqreg.ado    (not installed)
                     bsqreg.ado     now part of Stata 3.1
                     bsqreg.hlp     now part of Stata 3.1
-------------------------------------------------------------------------------
--Top Level--   --3 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as sg11.1 (see detail on insert)
         -> . _
```

Insert *sg11.1* provided three files. One is not installed; the other two are now part of Stata 3.1. Typically, the file summary will not be so contradictory. When *sg11.1* was written, the utility _bsqreg.ado was necessary but, in rewriting these routines for Stata 3.1, the utility was no longer necessary. More instruction on how to interpret the file information is provided below under the heading *Interpreting the detailed information*.

We are now being asked to enter a 1, 2, or an insert number from the list. Typing nothing and pressing *Return* will return us to the top level where we can perform another search:

```
--Top Level--   --3 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as sg11.1 (see detail on insert)
         -> .  We press Enter
--Top level--   Enter search specification, ?, ??, or end -> . _
```

Category searches

One way to find useful inserts on a particular subject is to list all the inserts published in an STB category. Since you may not remember the category codes, typing '?' will prompt your memory:

```
--Top level--   Enter search specification, ?, ??, or end -> . ?
-------------------------------------------------------------------------------
an   announcements                    ip   instruction on programming
cc   communications & letters         os   operating systems, hardware, &
dm   data management                       interprogram communication
dt   data sets                        qs   questions & suggestions
gr   graphics                         tt   teaching
in   instructions                     zz   not elsewhere classified

sbe  biostatistics & epidemiology     srd  robust methods & statistical diag.
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 vars      szz  not elsewhere classified
-------------------------------------------------------------------------------

--Top level--   Enter search specification, ?, ??, or end -> .
```

(Typing '??' will provide more detailed assistance.)

dm is the STB category for data management. Let us list all inserts published in this category:

```
--Top level--   Enter search specification, ?, ??, or end -> . dm*
--Top Level--   --20 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as dm2 (see detail on insert)
         -> . 1
-------------------------------------------------------------------------------
STB-2  dm1       Date calculator                                      M. Ureta
STB-3  dm2       Data format conversion: DBMS/Copy & Stat/Transfer     J. Hilbe
STB-3  dm2.1     Vendors´ response to review                         S. Dubnoff
STB-6  dm2.2     Stat/Transfer 2.0 review update                       J. Hilbe
STB-4  dm3       Automatic command logging for Stata               D. H. Judson
STB-5  dm3.1     Typesetting correction to automatic command loggin
STB-4  dm4       A duplicate-value identification program             M. Jacobs
STB-5  dm5       Creating a grouping variable for data sets           M. Jacobs
STB-5  dm6       Utility to document beginning and ending variable
                                                                    S. Becketti
STB-7  dm7       Utility to reverse variable coding                   M. Jacobs
STB-7  dm8       Command to unblock data sets                          J. Hilbe
STB-7  dm9       An ANOVA blocking utility                    P. A. Lachenbruch
STB-9  dm10      Infiling data: Automatic dictionary creation          W. Gould
STB-12 dm11      Matching the Current Population Surveys (CPS)      F. R. Welch
STB-12 dm12      Selecting claims from medical claims data bases   R. J. Vaughn
STB-13 dm12.1    Selecting claims from medical claims data bases   R. J. Vaughn
STB-13 dm13      Person name extraction                               W. Gould
STB-13 dm13.1    String manipulation functions                        W. Gould
STB-14 dm14      Converting Julian dates to Stata elapsed dates       C. Chapin
STB-14 dm14.1    Converting Stata elapsed dates to Julian dates     S. Becketti
-------------------------------------------------------------------------------

--Top Level--   --20 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as dm2 (see detail on insert)
         -> . _
```

Title searches

Let us find all inserts with the word correlation in the title. The forward slash (/) specifies this kind of search:

```
--Top level--   Enter search specification, ?, ??, or end -> . /correlation
--Top Level--   --6 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as sg5.1 (see detail on insert)
         -> . 1
-------------------------------------------------------------------------------
STB-8  crc14     Pairwise correlation coefficients
STB-5  sg5       Correlation coefficients with significance levels  S. Becketti
STB-13 sg5.1     Correlation coefficients with significance levels  S. Becketti
STB-3  snp3      Phi coefficient (fourfold correlation)           R. Goldstein
STB-5  sts1      Autocorrelation & partial autocorrelation graphs   S. Becketti
STB-13 sts3      Cross correlations                                 S. Becketti
-------------------------------------------------------------------------------
```

Author searches

Let us find all inserts written by Hamilton. The backwards slash \ specifies this kind of search:

```
--Top level--   Enter search specification, ?, ??, or end -> . \hamilton
--Top Level--   --6 matches found--
  Enter  1 (list by insert)    2 (list by issue)    end or nothing (back up)
         insert number such as sqv8 (see detail on insert)
         -> . 1
```

```
------------------------------------------------------------------------------
STB-3  sed4      Resistant normality check and outlier identificati
                                                              L. C. Hamilton
STB-6  sed6      Quartiles, outliers, & normality: Some Monte Carlo
                                                              L. C. Hamilton
STB-13 sqv8      Interpreting multinomial logistic regression    L. C. Hamilton
STB-2  srd1      How robust is robust regression?                L. C. Hamilton
STB-1  ssi1      Monte Carlo Simulation                          L. C. Hamilton
STB-4  ssi2      Bootstrap programming                           L. C. Hamilton
------------------------------------------------------------------------------
```

Filename searches

You have on your disk the file `weisgas.dta` and wonder if you obtained it from the STB. At the top level, you enter the filename preceded by an equal sign (`=`):

```
--Top level--   Enter search specification, ?, ??, or end -> . =weisgas.dta
------------------------------------------------------------------------------
STB-10 srd14     Cook-Weisberg test of heteroscedasticity         R. Goldstein
                 .ado and .hlp files:
                     cwhetero.ado   installed in C:\ADO
                     cwhetero.hlp   installed in C:\ADO
                 "Optional" files (files not installed in normal way):
                     cwhetero.log
                     mtb_tree.dta
                     weisflok.dta
                     weisgas.dta
------------------------------------------------------------------------------
```

You could get a list of all STB inserts contributing `.dta` data sets by typing '`=*.dta`'.

Interpreting the detailed information

Once the STBinformer locks onto one insert, which can happen at the outset or can happen subsequently when you choose an insert from the list, it provides detailed information about the software associated with that insert.

One possibility is that there is no software:

```
------------------------------------------------------------------------------
STB-13 sg11.2    Quantile regression standard errors              W. H. Rogers
                 There is no software associated with sg11.2
------------------------------------------------------------------------------
```

More typically, however, the output will look like this:

```
------------------------------------------------------------------------------
STB-7  dm7       Utility to reverse variable coding                  M. Jacobs
                 .ado and .hlp files:
                     omscore.ado    (not installed)
                     omscore.hlp    (not installed)
------------------------------------------------------------------------------
```

There are two files associated with this insert, `omscore.ado` and `omscore.hlp`, and neither is installed. If you had installed *dm7*, you would see something like,

```
------------------------------------------------------------------------------
STB-7  dm7       Utility to reverse variable coding                  M. Jacobs
                 .ado and .hlp files:
                     omscore.ado    installed in C:\ADO
                     omscore.hlp    installed in C:\ADO
------------------------------------------------------------------------------
```

although the details of where the files are installed will differ across platforms. In Unix, you might see that the files are installed in `~/ado` and, on a Macintosh, in `~:Ado`. In any case, the STBinformer is able to tell whether the files are installed by looking for them in system and user directories.

When the STBinformer finds the file in a "system" directory, its report differs slightly:

```
------------------------------------------------------------------------------
STB-8  sg7       Centile estimation command                        P. Royston
                 .ado and .hlp files:
                     centile.ado    now part of Stata 3.1
                     centile.hlp    now part of Stata 3.1
------------------------------------------------------------------------------
```

The determination that centile.ado and centile.hlp are not merely installed but are now a part of Stata 3.1 was made thusly: centile.ado and centile.hlp were files supplied with *sg7*. When the STBinformer looked for those files, it found them in the "official" ado-file directory (such as c:\stata\ado under DOS or /usr/local/stata/ado under Unix or ~:Stata:ado on a Mac; the "official" directory on your computer is the one listed first when you type 'adopath'.) In the previous case of omscore.ado and omscore.hlp, it also found the files, but it found them in some other directory. In the case before that, omscore.ado and omscore.hlp were not found at all and so were deemed to be not installed.

Sometimes, you will get a mixture of these determinations:

```
------------------------------------------------------------------------------
STB-9  sg11.1    Quantile regression bootstrapped standard errors      W. Gould
                 .ado and .hlp files:
                     _bsqreg.ado    (not installed)
                     bsqreg.ado     now part of Stata 3.1
                     bsqreg.hlp     now part of Stata 3.1
------------------------------------------------------------------------------
```

These are a bit more difficult to interpret but such listings often occur when an insert is incorporated into a subsequent Stata release. At the time bsqreg was originally written, it evidently needed a subroutine called _bsqreg. When bsqreg was updated for inclusion in Stata 3.1, that subroutine was no longer required. If you see something is now an official part of Stata, assume that anything that is not installed is no longer necessary.

Now imagine you had previously installed *sg11.1*. Here is what you would see from the detailed report on the insert:

```
------------------------------------------------------------------------------
STB-9  sg11.1    Quantile regression bootstrapped standard errors      W. Gould
                 .ado and .hlp files:
                     _bsqreg.ado    installed in C:\ADO
                     bsqreg.ado     now part of Stata 3.1
                                    old version also found in C:\ADO
                     bsqreg.hlp     now part of Stata 3.1
                                    old version also found in C:\ADO
------------------------------------------------------------------------------
```

There is nothing wrong with having both the official and old versions installed, but the old version is doing you no good. Stata always executes the official version in preference to the old one. You could save some disk space by deleting the old files.

The STBinformer is only able to track installation of files installed in the "normal" way, which is to say, those copied into your personal ado directory. Files that are not installed this way are labeled "optional":

```
------------------------------------------------------------------------------
STB-14 sg18      An improved R-squared                              P. Royston
                 .ado and .hlp files:
                     brsq.ado       installed in C:\ADO
                     brsq.hlp       installed in C:\ADO
                 "Optional" files (files not installed in normal way):
                     scott2.dta
------------------------------------------------------------------------------
```

The file scott2.dta provided with *sg18* was the data used in demonstrating the command. The author thoughtfully provided the data in case we wanted to experiment with it, but it was not required that we install it. Even if we did install it, there is no telling where we might have put the file.

The term "optional" is often correct, but sometimes the "optional" files are the whole point of the insert:

```
------------------------------------------------------------------------------
STB-15 dt1       Five data sets for teaching                  J. T. Anagnoson
                 "Optional" files (files not installed in normal way):
                     executed.dta
                     fertilty.dta
                     gulfwar.dta
                     pacrim.dta
                     unionmem.dta
------------------------------------------------------------------------------
```

Finally, the STBinformer does not track updates to Stata made in *crc* inserts. Whenever you ask to see the detail on a *crc* insert, it always gives the message that this update has been installed:

```
------------------------------------------------------------------------------
STB-15 crc33     Linear spline construction
                 Your Stata has this update installed.
------------------------------------------------------------------------------
```

You are warned that at this stage of the development, the reassuring message is not necessarily true. See *Future developments and current bugs* below. Of course, if you always install the official *crc* updates upon receiving an STB diskette, as you should, the message will be true.

The STB calendar

The STBinformer has one other feature. At the top-level menu, you can type '`>cal`' to obtain the STB calendar:

```
--Top level--   Enter search specification, ?, ??, or end -> . >cal
STB volume 1:
       May  1991   STB-1     (Stata 2.1)
       Jul  1991   STB-2     (Note:  ado-files from this period must be
       Sep  1991   STB-3             modified to include "version 2.1"
       Nov  1991   STB-4             at the top; see [0] new.)
       Jan  1992   STB-5
       Mar  1992   STB-6     (Stata 2.1, Stata 3.0 announced)
STB volume 2:
       May  1992   STB-7     (Stata 3.0)
       Jul  1992   STB-8     (Note:  ado-files from this period will work
       Sep  1992   STB-9             unmodified with the current version
       Nov  1992   STB-10            of Stata.)
       Jan  1993   STB-11
       Mar  1993   STB-12
STB volume 3:
       May  1993   STB-13
       Jul  1993   STB-14    (Stata 3.0, Stata 3.1 announced)
       Sep  1993   STB-15    (Stata 3.1)
       Nov  1993   STB-16
       Jan  1994   STB-17
       Mar  1994   STB-18
```

The production of the calendar is automated, so as of May 1994, it will extend itself another year.

Future developments and current bugs

The STBinformer is being developed for ultimate inclusion in every issue of the STB. The idea is that when you install the official *crc* updates, you will automatically update the on-line indexes. That is why the code assumes that if *crc* update is listed in the index, the corresponding software must be installed—it could not be any other way.

As things stand right now, however, I have not yet worked out all the kinks. The major problem is the size of the files. The code for `stb` is only 15,360 bytes, but the data files (STB-1 through STB-15) consume 128,000 bytes and, over time, this will grow. Corresponding to each issue of the STB is a `.dta` data set containing the contents and filenames for the issue. The current design makes no attempt to conserve disk space—for instance, the title and author of an insert is repeated in the data set for every file corresponding to the insert. Titles are stored as `str50`s and authors as `str30`s so, if corresponding to an insert are 10 files, the title and author are recorded unnecessarily nine additional times for a total waste of $9 \times 80 = 720$ bytes.

Of the current 128,000 bytes of `.dta` index files, I estimate that 49,120 bytes (38%) are unnecessary. In my defense, I can only say that, at the outset, I was more concerned with features than file size.

In any case, the file-size problem is somewhat fixable. Doing away with the redundancy, the index files would shrink to 78,880 bytes. Each new issue of the STB will consume another few thousand bytes. This may still be too much for some users, so perhaps we need to rethink the automatic installation of the index.

The other bug of which I am aware is that spelling has not been verified in the index files. A bug I suspect is that `stb` is probably not robust to blanks in the folder names (such as the name of the hard disk) on a Macintosh. If you do have trouble, rename your hard disk to remove the blank and then let me know.

Before continuing with this project, I seek advice. Is the system usable and adequate?