STATA
TECHNICAL
BULLETIN

March 1996
STB-30

A publication to promote communication among Stata users

Editor

Sean Becketti
Stata Technical Bulletin
8 Wakeman Road
South Salem, New York 10590
914-533-2278
914-533-2902 FAX
stb@stata.com EMAIL

Associate Editors

Francis X. Diebold, University of Pennsylvania
Joanne M. Garrett, University of North Carolina
Marcello Pagano, Harvard School of Public Health
James L. Powell, UC Berkeley and Princeton University
J. Patrick Royston, Royal Postgraduate Medical School

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

an58	Change of editors

Sean Becketti, Stata Technical Bulletin

This, the thirtieth issue of the *Stata Technical Bulletin*, is my last as editor. I am pleased to report that H. Joseph Newton is the new editor of the STB starting with the next issue. Professor Newton is a distinguished statistician and the head of the statistics department at Texas A&M University. He has extensive experience in time series analysis and time series computing, and he is the author of TIMESLAB: *A Time Series Analysis Laboratory*, a time series text and computer software package. He is also the current President of the Interface Foundation of North America, the organizing body of the annual Symposium on the Interface of Computing Science and Statistics, the largest annual meeting in the area of computational statistics. Under his able guidance, I am confident the STB will grow even more useful and interesting to the community of Stata users than it is now.

In my three years as editor, I have tried to remain as invisible as possible in order to let the readers and contributors determine the character of the STB. For instance, I have never listed my professional affiliation in the STB. As it happens, my professional training is as an economist and econometrician, but I hope and believe that this fact could not be discerned from the list of articles I have chosen to include in the STB (although it is fairly obvious from the examples in my own inserts). In addition, I subscribe to and read all the messages sent to the Stata listserver, but, except in special circumstances, I have not sent messages myself, again to avoid influencing the direction of debate and comment.

Now I would like to break with precedent. Since this is my last issue as editor, I feel it is appropriate to share with you some of my personal views about the past, present, and future of Stata and the STB.

I had the good fortune to be present at the birth of Stata. As a graduate student, I supported myself by working as a programmer on another commercial statistics package. As an assistant professor at UCLA, I had worked with Stata's creators on research projects unrelated to statistical software, but my previous programming experience was common knowledge. As a consequence, when they decided to create a new statistical package for PCs, I was asked to help test the early versions and to write a bit of the documentation.

I have to confess that I began the project with a skeptical attitude. PCs were just emerging from the toy stage, and their capabilities were limited. Also, my prior experience convinced me that I knew the right way to design statistical software and user interfaces. In my first meetings with Stata's development team, it became clear that we disagreed on the entire design of Stata.

Fortunately for you, the development team ignored my prejudices and produced the Stata you know and use. Instead of designing a pale imitation of a mainframe package, they completely rethought the interface between the statistician/data analyst and the software. New Stata users are sometimes confused initially—the most common question on the listserver seems to be "Why doesn't Stata implement procedure X the same way as package Y?"—but experienced users appreciate that Stata removes the barriers between you and your data. When Stata is at its best, it almost seems to disappear. You feel as if you are operating directly on your data. In my career, I have used more statistical programs than almost anyone alive, and I sincerely believe that Stata is the most convenient and most powerful package for the sophisticated user. Moreover, Stata's logical design and extensibility guarantee that it will never become outdated.

When the STB was introduced, I was skeptical again. I wasn't convinced that enough Stata users would invest the time and effort to produce high-quality Stata programs and readable, informative articles. Again, I was wrong. I was hooked as a reader with the first issue, and, by STB-7, I was a contributor. Every year, the variety, utility, and sophistication of STB inserts has increased noticeably. A key contribution of the STB has been its role in accelerating the development cycle. Many inserts have stimulated other users to contribute extensions and refinements of the initial program. In several of these cases, the high level of user interest and involvement convinced the development team at StataCorp to build these new functions into the next version of Stata.

After three years as editor of the STB, I can see changes in the Stata community that are changing the character and role of the STB. Three years ago, the STB was the primary channel for user communication. There was no Stata listserver, nor was there a Stata Web site. (In fact, there was no Web.) These innovations have increased the amount and quality of interaction within the Stata community far beyond anything imaginable at the time.

The STB has evolved, and will continue to evolve, in response to these changes. Early in the life of the STB, the pages were filled with questions from readers; these now are handled almost entirely by the listserver, the Web site, and StataCorp's EMAIL and help line services. The listserver also provides many of the "quick-and-dirty" Stata programs that comprised a large share of the inserts in the first few volumes.

The STB now specializes in programs of increasing sophistication, programs that would be difficult to distribute, explain and document via EMAIL. Many of the inserts are much longer, reflecting the complexity of these programs.

Another sign of the growing sophistication of STB submissions is the number that integrate Stata programs with scripts and programs in other languages: C, Pascal, Awk, Perl, and so on. This approach breaks down the boundaries between Stata and the

rest of the computing environment, eliminates any limits on what Stata (properly assisted) can do and emphasizes the relative strengths of Stata vis-a-vis other common software tools. I expect this trend to accelerate in the future.

Despite the increasing power and complexity of some of the programs published in recent issues, the simpler data management programs we publish continue to be the most popular contributions. Regardless of one's field of research, data still have to be prepared for estimation and reporting, and programs that make this chore easier will always play a prominent role in the STB.

In closing, I want to thank the many contributors who have made my job as editor interesting and fulfilling. You ultimately are responsible for the STB. I have simply helped to communicate your programs and your ideas to the wider Stata community. Without you, there is no STB and no Stata. I regret that I have been unable to meet most of you in person. Nonetheless, I feel I know several of you well. Thank you for help.

dm41 | Online documentation for _result() contents

John R. Gleason, Syracuse University, 73241.717@compuserve.com

Stata programs often save useful results in either `S_#` objects, or in `_result()`. The latter possibility is available only to built-in commands (for example, `correlate`), and not directly to programs (such as `alpha`) which are loaded from ado-files. On occasion, a user may wish to know whether `_result()` contains saved values from an operation just performed—for example, to capture some result to greater precision than shown on the screen. (The command `disp_res` will display the current contents of `_result()`, but without any explanation.) More commonly, a program author wants to invoke a built-in command to perform some necessary calculation, and needs to know which command(s) will deposit the required value in `_result()`, and at which position.

In either case, one will generally need to consult the *Saved Results* section of (possibly many) entries scattered through Volumes 2 and 3 of the Reference Manual. The search can be tedious, partly because about 50 commands are acknowledged to save in `_result()`, directly or by invoking other commands that do so. And, while the contents of `S_#` macros can be divined by examining ado-files, this technique fails for `_result()` because a built-in command has no ado-file. Only the Reference Manual has the required information, and it may not be close at hand.

This insert offers a solution, namely, online documentation for the contents of `_result()`. The *Saved Results* section of the Reference Manual entry for each command that saves in `_result()` has been entered into one of 16 Stata help files. These `.hlp` files are formatted, and used, exactly as though they contained help for ado-files named *fn*`.ado`, where *fn* is some filename. In particular, these files should be copied to a directory somewhere in the `S_ADO` path, just as one might install a new command and its help file; see [2] ado. Having done so, typing '`help` *fn*' displays the contents of *fn*`.hlp` on the screen, but there are several other ways to use the 16 `.hlp` files.

If you want to know what a specific command saves in `_result()`, there is the program `_result`, with syntax

`_result` { *cmdname* | `_all` }

where *cmdname* is the name of a Stata program or built-in command. If *cmdname* deposits in `_result()`, then `_result` finds and displays the appropriate help file; otherwise, it reports that the Reference Manual does not document *cmdname* as saving in `_result()`. For instance, to learn what the command `correlate` saves,

```
. _result corr
-------------------------------------------------------------------------------
_result( ) for correlate
-------------------------------------------------------------------------------

        1   no. of observations
        2   (not used)
        3   1 or variance of 1st variable
        4   correlation or covariance
        5   1 or variance of 2nd variable
```

Thus, for example, `correlate` deposits the correlation or covariance that it computes in `_result(4)`. On the other hand, `canon` is a command that does not save in `_result()`.

```
. _result canon
_result() contents not documented for canon
```

Note that for certain `matrix` commands, *cmdname* consists of two words; for example,

```
. _result mat ml
------------------------------------------------------------------------------
_result( ) for matrix mlout
------------------------------------------------------------------------------

        1   no. of observations
        3   degrees of freedom (if specified)
```

In all cases, *cmdname* may be abbreviated exactly as if it were itself the command being executed. Finally, note that '`_result _all`' will display all of the help files, with command names appearing in quasi-alphabetical order.

Another way to examine the help files for `_result()` is to use the index in the file `_res_idx.hlp`. This index lists each command that, according to the Reference Manual, saves in `_result()`, and beside it, the filename of the `.hlp` file pertinent to that command. The index can be viewed with `help`:

```
. help _res_idx
------------------------------------------------------------------------------
   _result() Help File Index
 Manual    Command    Help File
Section      Name      <fn>.hlp
------------------------------------------------------------------------------
  [5d]     count       _res5dcn
  [5d]     describe    _res5dds
  [5d]     inspect     _res5din
  [5s]     _qreg       _res5sqr
  [5s]     anova       _res5srg
(etc.)
```

In this display, commands that save in `_result()` are listed in the second column, with the filename *fn* of the relevant help file shown in the third column. The commands are sorted alphabetically within Reference Manual sections, which are displayed in the first column. Thus, to learn what `describe` saves in `_result()`, you could type

```
. help _res5dds
------------------------------------------------------------------------------
_result( ) for describe
------------------------------------------------------------------------------

        1   no. of observations
        2   no. of variables
        3   width
(etc.)
```

Better still, in Stata for Windows, you can view `_res_idx.hlp`—or any other `.hlp` file—with `winhelp`: Click on *Help* at the top of the Stata window, type `_res_idx` in the edit box, click the button next to *Stata command* (if necessary), and then click on *OK*. The contents of `_res_idx.hlp` will then be displayed in a scrollable window, with both command names and help filenames highlighted in green, indicating Windows hypertext links to other help files. Thus, if you click on the `anova` item, Stata's help file for the `anova` command will be displayed. If instead you click on the `_res5srg` item, the help file explaining the contents of `_result()` following the `anova` command will be displayed. (See [1] help and *Getting Started with Stata for Windows* for additional detail.)

As an alternative, the entire set of 16 `.hlp` files has been collected into a single file named `_res_all.hlp`. The entries in this file are in quasi-alphabetical order—quasi-, because in some cases one entry serves for several commands. Here, for example, is the entry for commands related to [7] maximize; it is also the help file `_res7max.hlp`:

```
------------------------------------------------------------------------------
_result( ) for blogit    bprobit   clogit    cnreg     cox       dprobit
                         logistic  logit     mlogit    ologit    oprobit   probit
                         swcnreg   swcox     swlogis   swlogit   swologit  swoprbt
                         swprobit  swtobit   tobit
------------------------------------------------------------------------------

        1   no. of observations
        2   log-likelihood value
        3   chi-square degrees of freedom
(etc.)
```

Note that typing '`help _res_all`' is equivalent to the command '`_result _all`'.

The program author trying to decide how to place a necessary value in `_result()` might type '`help _res_all`'. But a better way is to load `_res_all.hlp` into a text editor, and allow the editor's *Find* or *Search* command to do most of the work.

For this reason, the contents of _res_all.hlp have also been stored in an ASCII text file named _res_all.txt. The only difference between the two files is that, in _res_all.txt, all highlighting characters (^ and @) have been removed, and each occurrence of '.-' has been expanded to a string of 79 '-' characters—which makes _res_all.txt easier to read. To use, load _res_all.txt into an editor, choose a word or phrase describing the computed result that is required, and tell the editor to find text containing that word or phrase.

To illustrate, suppose your program needs to know how many observations on some float variable happen to be integer-valued. You are about to program that calculation when it occurs to you that a command may already be available for this task. You make an inspired guess and tell your editor to search for the word 'integer' in _res_all.txt. A popular shareware editor for the Windows environment produced this result:

```
Searching for: integer
_res_all.txt(73): 5   no. of negative, integer observations
_res_all.txt(74): 6   no. of positive, integer observations
Found 2 occurrence(s)
```

(The numbers in parentheses after _res_all.txt are line numbers.) Moving to lines 73–74 reveals that they are part of the entry for the command inspect, and that following inspect, the information you require will be saved in _result(3), _result(5), and _result(6).

ip8.1 An even more enhanced for command

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011) 44 181 740 3119

Royston's (1995) for2 command overcame certain limitations of Stata's for command, but it is still limited in that only one *forlist* is allowed. for3 allows multiple *forlists*, which significantly increases its potential applications.

As a simple artificial, but instructive, example, suppose I have three variables called a1, a3 and a5 whose means I wish to compare in a paired sense with those of three other variables, va, vb and vc. That is, I want to compare a1 with va, a3 with vb and a5 with vc. I could of course type

```
. ttest a1 = va
. ttest a3 = vb
. ttest a5 = vc
```

to perform the three paired Student t tests, which is acceptable if somewhat tedious to key in. A compact version using for3 is

```
. for3 a1 a3 a5 \ va vb vc : ttest @=@@
```

If the two sets of three variables were stored in the right order then

```
. for3 a1-a5 \ va-vc : ttest @=@@
```

would achieve the same result, as would

```
. for3 1-5/2 \ va-vc, ltype(numeric varlist) : ttest a@=@@
. for3 1-5/2 \ va-vc, ltype(n) : ttest a@=@@
```

The principle is that the *forlist*(s) which follow for3 are matched with the @ symbols which follow the colon and the Stata command (in this case, ttest). In the first example, the items in the first *forlist* (a1 a3 a5) are substituted for the single @ and those in the second *forlist* (va vb vc) for @@. There is no upper limit to the number of *forlists* allowed.

The *forlists* are separated by a backslash (\).

for3 has a new option not found in for2: ltype(typelist). This defines the type of each *forlist* in the same order as the *forlists* appear. Permissible types are varlist (the default), any and numeric. The last two of these are identical to the any and numeric options in for2—see the help file for3.hlp for details. In the third example above, 1-5/2 is a numeric *forlist* and is expanded by for3 to 1 3 5, whereas va-vc is of type varlist and is expanded to va vb vc. Items varlist, any and numeric in ltype() may be abbreviated to v, a and n respectively.

The syntax of `for3` is

`for3` *list1* [*list2* [*list3* ...]] [, noheader ltype(*type* [*type* [*type* ...]]) pause nostop] : *stata_cmd*

The other options are described in the help file `for3.hlp`.

Special characters

You may occasionally wish to include a comma or a colon in a *forlist* of type `any`. To avoid being mistaken for part of the syntax of the `for3` command itself, these characters must be repeated, so `::` will be interpreted as `:` and `,,` as `,` . If a *forlist* of type `any` is to contain embedded space(s), they must be entered as `#`'s. For example, the *forlist* item `if#x==1` will be interpreted as `if x==1` when *stata_cmd* is executed.

The examples given above are very simple; there are many other possible uses of `for3`. As always, this is work in progress, and I would be happy to have feedback from users regarding improvements, limitations, etc.

Acknowledgment

I am grateful to Tony Tam for suggesting the idea behind `for3`.

References

Royston, P. 1995. ip8: An enhanced `for` command. *Stata Technical Bulletin* 26: 12.

sg49 An improved command for paired t tests

John R. Gleason, Syracuse University, 73241.717@compuserve.com

Stata's `ttest` command offers two ways to perform the classic matched-pairs (repeated measures) t-test; see [5s] ttest. One possibility is that the sample pairs (say, y_1 and y_2) are stored as two distinct variables. The other approach requires that the difference between paired observations (say, $d = y_2 - y_1$) be stored as a variable. However, it often happens that y_1 and y_2 are stored as values of a single response variable, corresponding to different levels of a blocking or within-subjects factor. In this case, one would need to create one of the two situations assumed by the `ttest` command. While this is not an insurmountable problem, it does require a bit of data manipulation that can be inconvenient and perhaps error-prone, especially for a novice Stata user. One could, of course, obtain the correct p-value and the squared t statistic by using the `anova` command. This would require no data manipulation, but the resulting output would not include means and standard deviations (or standard errors), and would likely be unhelpful to many novice users.

This insert presents a command `rmttest` that performs the repeated measures (or, matched-pairs) t test when pairs y_1 and y_2 are stored as values of a single response variable y. In addition to automatically manipulating the data as required, `rmttest` offers options not available from `ttest`, including confidence intervals for the population means, and a scatterplot of the paired samples to help visualize the results of the t test and confidence intervals.

The syntax of `rmttest` is

`rmttest` *depvar* [`if` *exp*] [`in` *range*] [,

by(*byvar* [*val1 val2*]) ci graph id(*idvar*) level(*cilevel*) *graph_options*]

depvar is the dependent variable (y) among whose values the paired samples are to be found. `by(`*byvar*`)` and `id(`*idvar*`)` are not in fact optional: *byvar* is a variable (typically, a within-subjects factor) two of whose values select the samples to be compared. *idvar* is a variable (typically containing subject identifiers) that defines the pairings of values in the two samples. The option `ci` requests confidence intervals for each mean; `level(`*cilevel*`)` sets the confidence level. (The default value of *cilevel* is .95.) The option `graph` requests a scatterplot of the sample pairs. *graph_options* stands for most of the options allowed with `graph, twoway`; such options are ignored unless `graph` is present. The following examples explain these items in greater detail.

To illustrate, consider some results on maze learning in rats, given in Table 11.26 of Bliss (1967, p. 327), who attributed these data to an earlier unpublished report by Kaplan et al. (1951). The data also appear in the file `maze.dta` included with this insert:

```
. use maze
(Maze learning in rats)
. describe
Contains data from maze.dta
  Obs:    50 (max= 30486)                    Maze learning in rats
 Vars:     4 (max=    99)                    27 Jan 1996 12:50
Width:     8 (max=   200)
  1. rt           int    %8.0g               Running time (sec.)
  2. rat          byte   %9.0g               Rat no.
  3. trial        byte   %9.0g               Trial no.
  4. l10_rt       float  %9.0g               log10(Running time)
Sorted by:
```

The basic response variable (`rt`) is the time, in seconds, required for a rat to run through a certain maze, measured for each of 10 rats on five non-consecutive trials of a learning experiment:

```
. tabulate trial, sum(rt)
           |    Summary of Running time (sec.)
   Trial no.|        Mean   Std. Dev.       Freq.
------------+------------------------------------
          2 |       216.4    124.2839          10
          5 |         105   68.071857          10
          8 |        50.3   31.552602          10
         11 |        39.3   23.967802          10
         14 |          19   9.2376043          10
------------+------------------------------------
      Total |          86   95.694667          50
```

However, Bliss chose (with good reason) to analyze the logarithm of the running times (`l10_rt`).

The usual within-subjects ANOVA for `l10_rt` is easily performed,

```
. anova l10 rat trial
                           Number of obs =      50     R-squared     =  0.8923
                           Root MSE      = .178866     Adj R-squared =  0.8534
                  Source |  Partial SS    df       MS           F     Prob > F
              -----------+----------------------------------------------------
                   Model |  9.54356043    13  .734120033      22.95     0.0000
                         |
                     rat |  3.03461679     9  .337179644      10.54     0.0000
                   trial |  6.50894364     4  1.62723591      50.86     0.0000
                         |
                Residual |  1.15175391    36  .031993164
              -----------+----------------------------------------------------
                   Total |  10.6953143    49  .218271721
```

and $MS_{\text{Residual}} = 0.032$ can be used as a variance estimate to compare individual trials. One might prefer instead to use paired t-tests, perhaps to avoid the sphericity assumption on which the foregoing ANOVA is based.

Suppose we wish to compare the mean of the log running times on trials 2 and 5 using a paired t test :

```
. rmttest l10 if trial==2 | trial==5, by(trial) id(rat)
     trial |      Obs        Mean   Std. Err.
-----------+--------------------------------
         2 |       10    2.253779    .0952236
         5 |       10    1.947757    .0824908
-----------+--------------------------------
      Diff |       10   -.3060223     .082314
  Ho:  mean(Diff) = 0    df = 9    t = -3.72    Pr > |t| = 0.0048
```

Thus, there was a significant decrease in mean log running time between trials 2 and 5. Notice that in this example, the 20 observations are paired in terms of the *idvar* `rat`, the two samples correspond to two levels of the *byvar* `trial`, and that those two trials are selected by the `if` clause. Both the `by` and `id` options must be present.

It may seem awkward to select values of the *byvar* with an `if` clause, and more natural to simply name the *byvar* and the relevant pair of its values; `rmttest` also permits that syntax. Suppose we wish to compare trials 5 and 11 of the experiment, and obtain a 95% confidence interval for the mean log running time on each trial, as well as for the difference in those means:

```
. rmttest l10, by(trial 5 11) id(rat) ci
    trial |      Obs        Mean    Std. Err.        [95% Conf. Interval]
----------+---------------------------------------------------------------
        5 |       10    1.947757    .0824908         1.76115    2.134364
       11 |       10    1.487385    .1132231        1.231256    1.743513
----------+---------------------------------------------------------------
     Diff |       10   -.4603722    .1165071       -.7239294   -.1968149

  Ho:  mean(Diff) = 0    df = 9    t = -3.95    Pr > |t| = 0.0033
```

In this style, `by(`*byvar val1 val2*`)` serves both to name the *byvar* and to confine the analysis to cases where *byvar* = *val1* or *byvar* = *val2*. An `if` or `in` clause can still be used to select observations with respect to other criteria.

Graphing the data for a paired t test

A scatterplot of the data used to perform a paired t test may be helpful in several ways—for example, to serve as a quick check for bivariate outliers, or to visualize the confidence intervals produced by the `ci` option. The `graph` option produces such scatterplots after performing the t test . To illustrate, suppose we wish to compare the rats' performance on trials 11 and 14, and to examine the data both with respect to the observed running times (`rt`) and their log transforms (`l10_rt`). First, the comparison in terms of `rt`:

```
. rmttest rt, by(trial 11 14) id(rat) gr xlab ylab
    trial |      Obs        Mean    Std. Err.
----------+--------------------------------
       11 |       10        39.3    7.579285
       14 |       10          19    2.921187
----------+--------------------------------
     Diff |       10       -20.3    5.587784

  Ho:  mean(Diff) = 0    df = 9    t = -3.63    Pr > |t| = 0.0055
```

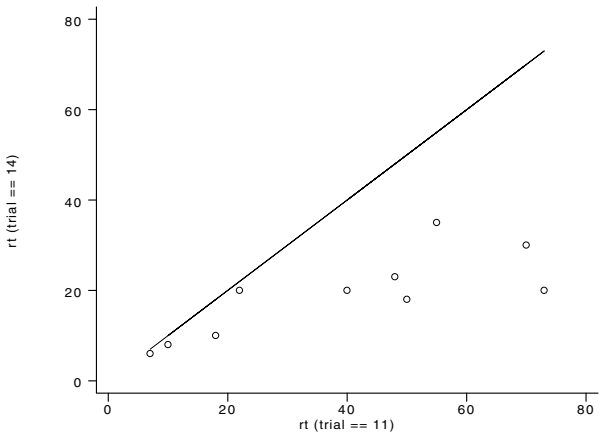

Figure 1

Figure 1 shows the 10 running times from trial 11 on the horizontal axis, and trial 14 times on the vertical axis. Note that both axes are automatically labeled with the appropriate values of *byvar* (although each label can be suppressed with titles). The diagonal line is the line of equality (ordinate = abscissa); under the null hypothesis of the paired t test, the sample pairs should scatter randomly about that line. It is clear from Figure 1 that each of the 10 rats ran the maze in less time on trial 14 than on trial 11.

Secondly, here is the comparison of trials 11 and 14 on the log scale, along with 97.5% confidence intervals:

```
. rmttest l10, by(trial 11 14) id(rat) gr xlab ylab ci l(.975)
    trial |      Obs        Mean    Std. Err.      [97.5% Conf. Interval]
----------+---------------------------------------------------------------
       11 |       10    1.487385    .1132231         1.18338     1.79139
       14 |       10    1.222252     .078611        1.011181    1.433323
----------+---------------------------------------------------------------
     Diff |       10   -.2651327    .0534877       -.4087478   -.1215176

  Ho:  mean(Diff) = 0    df = 9    t = -4.96    Pr > |t| = 0.0008
```

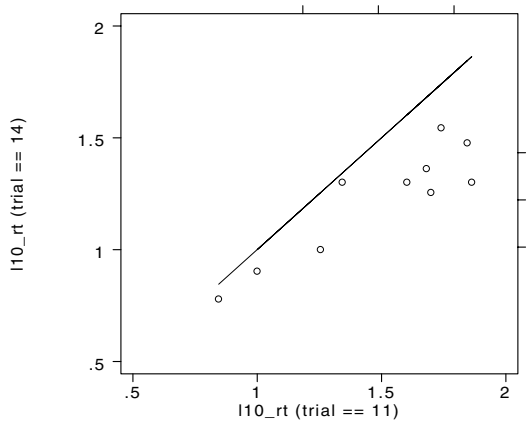

Figure 2

Figure 2 displays an additional feature that is activated by supplying both the `graph` and `ci` options. The three tick marks on the top axis correspond to the sample mean log running time on day 11 (1.487) and the 97.5% confidence limits (1.183 and 1.791) for the population mean. Similarly, the tick marks on the right axis represent the sample mean on day 14 and associated 97.5% confidence limits.

Remarks

1. As noted in the examples above, observations may be selected by combinations of `if` and `in` clauses and the required `by` option. However accomplished, the selection must result in a set of observations in which *byvar* assumes exactly two distinct values.
2. Typically, *byvar* and *idvar* will be integer-valued variables. It is possible to use a variable of type `float` as the *byvar*, but because of precision problems, some care must be taken; see [2] precision. For example, when selecting two *byvar* values with an `if` clause, use `if x==float(1.5) | x==float(2.5)` rather than `if x==1.5 | x==2.5`. If the *byvar* values are selected with the `by` option, the `float` function is automatically applied when *byvar* is of type `float`. That is, if `x` is a `float` variable, typing `by(x 1.5 2.5)` has the same effect as if you had typed `by(x float(1.5) float(2.5))` — which, by the way, is not permitted.
3. `rmttest` uses its own `level` option, rather than Stata's `S_level` parameter, to set confidence intervals. The default, as with `S_level`, is for 95% confidence. The `level` option argument can be supplied either as a fraction or as a percentage; that is, `level(.975)` and `level(97.5)` both request 97.5% confidence intervals.
4. Most options available with Stata's `graph, twoway` command may be supplied along with `rmttest`'s `graph` option. Here are the exceptions: the `ttick` and `rtick` options are unavailable when the `ci` option is in effect, as is the `connect` option in any case. The `symbol` option may be used to set the plot symbol for the data (the default is `o`), but the line of equality is always drawn with an invisible symbol. If the `pen` option is supplied, its first and second arguments select the pens for plotting the data, and drawing the line of equality, respectively.

References

Bliss, C. I. 1967. *Statistics in Biology*, Vol. I. New York: McGraw–Hill.

Kaplan, S. J., C. D. Tait, P. D. Wall, and R. B. Payne. 1951. Behavioral changes following radiation: 1. Study of retention of a partially learned maze habit. U.S. Air Force School of Aviation Medicine unnumbered project, Report 1.

sg50	Graphical assessment of linear trend

Joanne M. Garrett, Division of General Medicine and Clinical Epidemiology,
University of North Carolina, FAX 919-966-2274, EMAIL garrettj@med.unc.edu

This insert presents a simple method of graphing log odds of a binary outcome or means of a continuous outcome by categories of a continuous predictor variable.

Background

A scatter plot can be useful to examine the relationship between a continuous predictor, X, and a continuous outcome, Y. Visual patterns may suggest a linear positive (or negative) relationship, a curvilinear relationship, or no relationship at all. However, when Y is binary, scatter plots become ineffective due to the excessive number of ties on Y. It would be helpful to have a graphical method to examine this relationship as an exploratory step before modeling. A suggested approach is to group X into intervals and calculate the proportion of observations where $Y = 1$ within each interval (Harrell 1985, Hosmer 1989).

The logistic regression model can be written in a linear form by using a logit (i.e., "log odds") transformation

$$\text{logit}\, p(Y = 1|X) = \alpha + \beta X$$

where

$$\text{logit}\, p = \ln[p/(1-p)]$$

It follows that if the proportions of $Y = 1$ calculated within defined intervals of X are transformed to log odds, and the log odds are plotted against midpoints of the X intervals, we have a graphical method of assessing the relationship between Y and X. Graphs such as these can be useful for examining whether a linear relationship exists. If the relationship is linear, it is appropriate to model X in its original form. If a linear relationship does not exist, the graph may suggest a nonlinear alternative (for instance, a quadratic or logarithmic transformation) or natural groupings of X to recode into categories.

How to categorize X and then what values of X to use for the X-axis category midpoints needs to be determined. Below are three possible choices:

1. Divide X using some meaningful cut point. For example, round age into 5 year increments, e.g., 20, 25, 30, etc. These values become the midpoints of the categories. Although this method is intuitively appealing, some of the categories may have small sample sizes, making the calculation of proportions within a category unstable.
2. Divide X into some specified number of groups (categories) with an equal number of observations in each group. For example, suppose age ranges from 20 to 60 years and there are 500 observations. The data could be divided into 10 groups of 50 observations in each. The age ranges would tend to vary from group to group. These age categories may not make as much intuitive sense as rounding, but with near equal sample sizes, the calculated proportions would tend to be more stable. Next we need to decide what values of X to use for the X-axis midpoints. A logical choice would be the mean. For instance, suppose the first group consists of 50 observations with a minimum value for age of 20, a maximum value of 27, and a mean age for the 50 observations of 25.5. This mean would represent the first age category on the X-axis. The second group of 50 observations might include ages 28 to 31 with a mean age of 29.7, etc.
3. If X consists of only a few possible values with many observations for each value, using the original values as categories may be appropriate. For instance, age might already be grouped into 5-year increments and coded as 20, 25, 30, etc. Or an ordinal variable, such as severity of symptoms (none, mild, moderate, severe) might be coded as the integer values 1, 2, 3, and 4.

This insert presents `lintrend`, a program that graphically examines the relationship between the log odds of a binary outcome, Y, by categories of an ordinal or interval independent variable, X. Alternatively, if the outcome is continuous, `lintrend` will calculate means of Y rather than proportions and log odds. X is divided into categories using any of the three methods described above. The idea and methods for this ado file come from Harrell and Lee (1985).

Syntax and options

`lintrend` *yvar xvar* [`if` *exp*] , { `groups(#)` | `integer` | `round(#)` } [`plot(`*plot-type*`)` *graph-options*]

xvar can be an interval or ordinal independent variable. *yvar* can be either a binary (0,1) outcome, resulting in the calculation of proportions and log odds by categories of *xvar*, or a continuous outcome, resulting in the calculation of means by categories of *xvar*.

The first option—which is a choice of one of three options—indicates how *xvar* is to be categorized and must be specified. The choices are

`groups(#)` divides *xvar* into # of categories of the same sample size. Ties on *xvar* at group cut points are placed in the lower category. Therefore, sample sizes may vary depending on the number of ties. The mean values of *xvar* by category are used as the values for X.

`round(#)` divides *xvar* into categories rounded to the nearest #. The rounded values of *xvar* are used for X.

`integer` uses the original values of *xvar* for X.

`plot(`*plot-type*`)`, an optional option, requests a graph. The available plot types are

plot-type	Meaning
`mean`	graph of means (continuous *yvar*) by *xvar* categories
`prop`	graph of proportions (binary *yvar*) by *xvar* categories
`log`	graph of log odds (binary *yvar*) by *xvar* categories
`both`	graph of both proportions and log odds categories

Example 1

We wish to examine the relationship between age and hypertension (`hbp`) in data taken from a study of 1784 adults between the ages of 24 and 51 years (James 1992). Using `lintrend`, we divide the sample into ten groups of approximately equal size, calculate the proportions and log odds of being hypertensive (1=hypertensive, 0=normal) within each age group, and request both a graph of proportions and a graph of log odds.

```
. use hyperten
. describe
Contains data from hyperten.dta
  Obs:  1784 (max= 74936)
 Vars:     4 (max=    99)                    26 Jan 1996 15:36
Width:     7 (max=   200)
  1. sbpavg       float  %9.0g                 Average Systolic Blood Pressure
  2. age          byte   %8.0g                 Current Age
  3. ses          byte   %8.0g       seslbl    Socioeconomic Status
  4. hbp          byte   %8.0g       yesno     High Blood Pressure
. lintrend hbp age, groups(10) plot(both) xlab ylab
 The proportion and log odds of hbp by categories of age
  (Note: 10 age categories of equal sample size;
     Uses mean age value for each category)
    age       min       max       d   total     hbp  logodds
   26.0        24        27      24     246    0.10    -2.22
   28.5        28        29      27     166    0.16    -1.64
   30.6        30        31      26     169    0.15    -1.70
   32.4        32        33      34     179    0.19    -1.45
   34.5        34        35      43     151    0.28    -0.92
   36.5        36        37      44     164    0.27    -1.00
   39.0        38        40      70     220    0.32    -0.76
   41.9        41        43      66     152    0.43    -0.26
   45.5        44        47      86     200    0.43    -0.28
   49.0        48        51      65     137    0.47    -0.10
```

(graph appears, see Figures 1 and 2)

The mean (`age`), minimum (`min`), and maximum (`max`) values for each age category are reported, as well as the number of hypertensives (`d`), the total sample size (`total`), proportion of hypertensives (`hbp`=`d/total`), and log odds of the proportion. A table like this is always printed, whether or not a graph is requested. Note that the ranges for the age categories differ. The sample sizes are not equal, although they are divided as evenly as possible without splitting the tied values for age. We can see that the proportion of hypertensives is increasing as age increases. The first graph (Figure 1) plots the proportion of hypertensives versus the mean age for each category, and the second graph (Figure 2) plots the log odds versus the mean age for each category, with a "best fit" linear regression line. The graph suggests a positive linear relationship between age and hypertension.

(*Graphs on next page*)

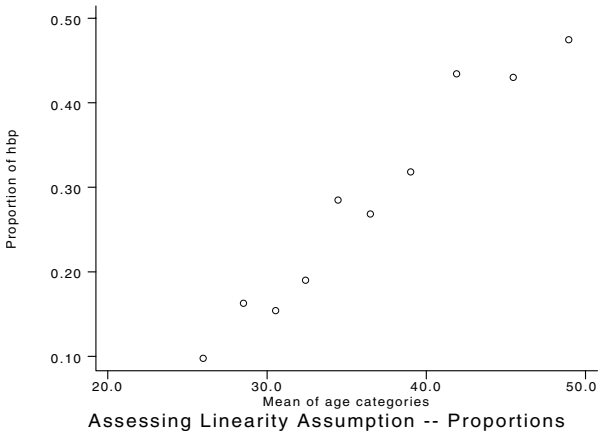

Figure 1

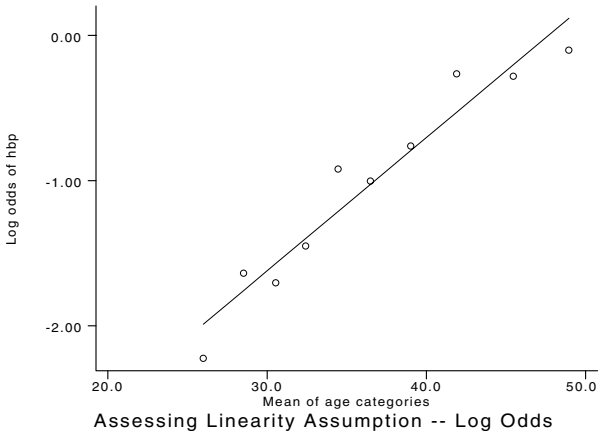

Figure 2

Example 2

Using the same data and variables, our second example rounds age into groups of five-year increments. Again, both graphs of proportions and log odds are requested.

```
. lintrend hbp age, round(5) plot(both) xlab ylab
The proportion and log odds of hbp by categories of age
  (Note: age in categories rounded to nearest 5)
    age       min       max       d   total       hbp   logodds
     25        24        27      24     246      0.10     -2.22
     30        28        32      72     436      0.17     -1.62
     35        33        37     102     391      0.26     -1.04
     40        38        42     117     323      0.36     -0.57
     45        43        47     105     249      0.42     -0.32
     50        48        51      65     137      0.47     -0.10
```

(graph appears, see Figures 3 and 4)

The rounded value for age is used for the category midpoint and each category range is five years. Note that the minimum possible age value for the category `age==25` is 23, but, since there was no one in the sample of that age, the actual lowest value (24) is used for the minimum. The same is true for the maximum value for `age==50` (51 rather than 52). Once again the graph of the log odds (Figure 4) suggests a linear trend, although hypertension may be increasing more rapidly before the age of 40, and less rapidly over 40.

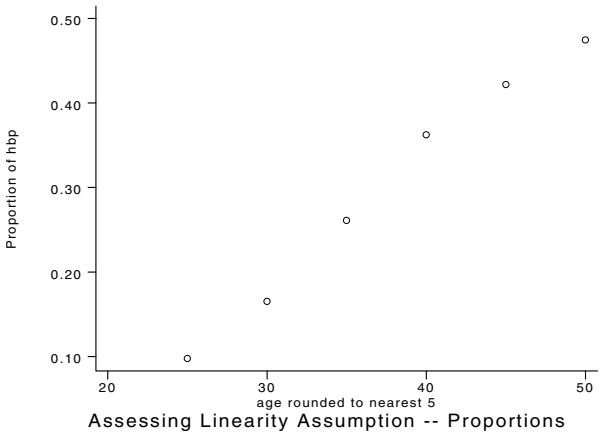

Figure 3

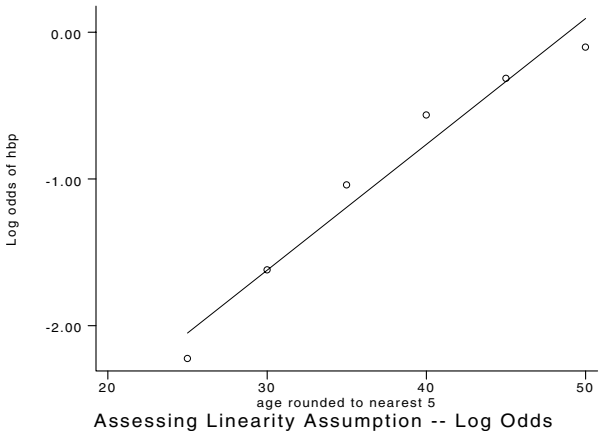

Figure 4

Example 3

Another variable in the hypertension data set is a measure of socioeconomic status (`ses`). This is an ordinal variable grouped into three categories: low socioeconomic status (1), middle socioeconomic status (2), and high socioeconomic status (3). This example requests the proportion and log odds of hypertension for each of the categories of `ses`, and a graph of the log odds. From the table and graph, we see that hypertension decreases as `ses` increases.

```
. lintrend hbp ses,  integer plot(log) xlab(1,2,3) ylab

The proportion and log odds of hbp by categories of ses

  (Note: ses in categories using original values)

     ses      d    total     hbp  logodds
   1:Low    215      670    0.32    -0.75
2:Middle    138      537    0.26    -1.06
  3:High    117      512    0.23    -1.22
```

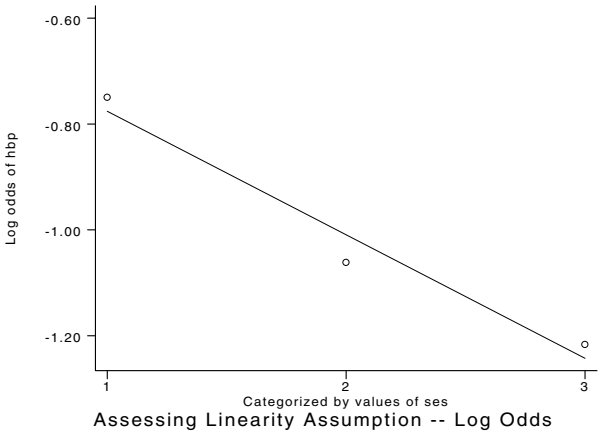

Figure 5

Example 4

The previous examples examined a binary outcome. Had Y been continuous, `lintrend` would have calculated means rather than proportions and log odds. Suppose that in the first example the outcome had been average systolic blood pressure (`sbpavg`) instead of hypertension. We divide the sample into ten groups of approximately equal size, calculate the mean systolic blood pressure within each age category, and request a graph of the mean systolic blood pressures by the mean age for each category.

```
. lintrend sbpavg age, groups(10) plot(mean) xlab ylab

The mean of sbpavg by categories of age

  (Note: 10 age categories of equal sample size;
     Uses mean age value for each category)

      age       min        max    total      sbpavg
     26.0        24         27      246      120.83
     28.5        28         29      166      119.07
     30.6        30         31      169      120.27
     32.4        32         33      179      122.30
     34.5        34         35      151      124.16
     36.5        36         37      164      124.79
     39.0        38         40      220      123.96
     41.9        41         43      152      129.62
     45.5        44         47      200      131.05
     49.0        48         51      137      132.59
```

(graph appears, see Figure 6)

The age categories are the same as in Example 1. We see that systolic blood pressure increases with increasing age. There are fluctuations around the line, but essentially the increase is linear.

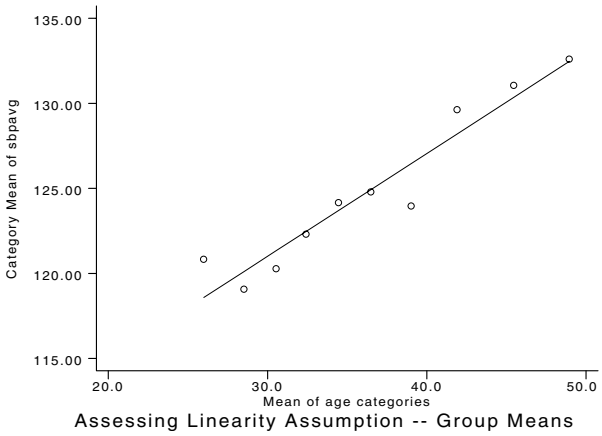

Figure 6

Example 5

This final example uses data from a study of low back pain (Carey 1995). A sample of 1552 patients with low back pain were followed for six months to determine characteristics related to recovery. For this example, we examine the relationship between a continuous measure of how disabling the back pain is at baseline (`score`: possible values from 0=no disability to 23=totally disabled) and the probability of recovery by eight weeks (`better`: 1=all better, 0=not better). The variable for disability, `score`, is grouped into twelve categories of like sample size. Since `better` is binary, proportions and log odds are calculated. A graph of the log odds is requested.

```
. use backpain

. describe

Contains data from backpain.dta
  Obs:  1552 (max= 74936)                    Recovery from back pain
 Vars:     4 (max=    99)                    6 Feb 1996 16:30
Width:     6 (max=   200)
  1. better       byte   %8.0g       yesno    Better at 8 Weeks
  2. score        byte   %8.0g                Disability Score
  3. score_2      int    %9.0g                Disability Score Squared
  4. score_3      int    %9.0g                Disability Score Cubed

. lintrend better score, groups(12) plot(log) xlab ylab

The proportion and log odds of better by categories of score

  (Note: 12 score categories of equal sample size;
     Uses mean score value for each category)

    score       min        max      d    total  better  logodds
      0.4         0          1    151      170    0.89     2.07
      2.5         2          3    100      136    0.74     1.02
      4.6         4          5     69      105    0.66     0.65
      6.5         6          7     67      119    0.56     0.25
      8.5         8          9     75      139    0.54     0.16
     10.5        10         11     60      128    0.47    -0.13
     12.5        12         13     57      118    0.48    -0.07
     15.1        14         16     96      197    0.49    -0.05
     17.0        17         17     33       79    0.42    -0.33
     18.5        18         19     49      147    0.33    -0.69
     20.5        20         21     40      121    0.33    -0.71
     22.5        22         23     24       93    0.26    -1.06
```

(graph appears, see Figure 7)

Patients with lower baseline disability scores were much more likely to have recovered by eight weeks (86 percent in the lowest category) than were patients with high scores (24 percent in the highest category). The graph shows this decreasing trend. However, there is a very sharp drop in recovery as disability scores increase from 0 to about 10, a leveling off with scores from about 10 to 15, and then another sharp drop as scores increase above 15. A shape such as this might suggest a polynomial transformation of `score`. The following logistic regression model, which allows the probability of recovery to be a cubic function of disability level (`score_2 = score*score`; `score_3 = score_2*score`), confirms the relationship suggested by the graph.

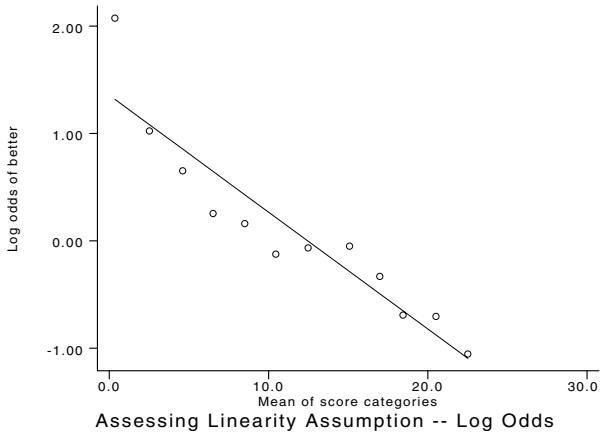

Figure 7

```
. logistic better score score_2 score_3
Logit Estimates                                   Number of obs =   1552
                                                  chi2(3)       = 207.17
                                                  Prob > chi2   = 0.0000
Log Likelihood = -969.56991                       Pseudo R2     = 0.0965
------------------------------------------------------------------------------
  better | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   score |   .6429993   .0497393    -5.709   0.000       .5525425    .7482649
 score_2 |   1.032048    .008033     4.053   0.000       1.016423    1.047913
 score_3 |   .9991876   .0002237    -3.631   0.000       .9987493    .9996261
------------------------------------------------------------------------------

. test score_2 score_3
 ( 1)  score_2 = 0.0
 ( 2)  score_3 = 0.0
           chi2(  2) =   20.36
         Prob > chi2 =    0.0000
```

References

Carey T. S., J. M. Garrett, A. Jackman, C. McLaughlin, J. Fryer. 1995. Outcomes of care in acute low back pain: Effects of initial practitioner choice. *New England Journal of Medicine* 333: 913–917.

Harrell F. E. and L. L. Lee. 1985. The practical value of logistic regression. *Proceedings: 10th Annual SAS Users Group International Conference* 1031–1036.

Hosmer D. W. and S. Lemeshow. 1989. *Applied Logistic Regression.* New York: John Wiley & Sons.

James S. A., N. L. Keenan, D. S. Strogatz, S. R. Browning, and J. M. Garrett. 1992. Socioeconomic status, John Henryism, and blood pressure in black adults: The Pitt County Study. *American Journal of Epidemiology* 135: 59–67.

snp10	Nonparametric regression: Kernel, WARP and k-NN estimators

Isaías Hazarmabeth Salgado-Ugarte, Makoto Shimizu and Toru Taniuchi,
University of Tokyo, Faculty of Agriculture, Department of Fisheries, Japan
FAX (011) 81 3 3812 0529, EMAIL fes01@tzetzal.dcaa.unam.mx

Regression is undoubtedly the most used statistical procedure (Scott 1992). Moreover, many variations on linear least-squares regression have been developed to model relationships that violate, in one or more ways, the assumptions of the classical model. Stata provides a particularly complete set of regression-based tools: the `regress` and `fit` commands (with their associated diagnostic commands), commands for quantile (median) regression, robust regression, logistic regression, and more. Moreover, the addition of commands to estimate generalized linear models (Hilbe 1993, 1994a; Royston 1994a) extends Stata's capabilities to this modern and flexible regression-based approach (Royston 1994b, Hilbe 1994b).

One technique missing from Stata's collection is kernel regression, and it is the purpose of this insert to correct that oversight. Among the nonparametric procedures, kernel estimators are recognized as important exploratory and analytical tools. Härdle (1990) notes, for instance, that kernel estimators possess two characteristics that distinguish them from other nonparametric estimators: they are straightforward to implement and understandable on an intuitive level. In previous inserts, we presented some Stata programs to calculate kernel estimates for univariate data (Salgado-Ugarte et al. 1993, 1995a). This time we extend this approach to bivariate data by presenting programs that calculate nonparametric regression estimators.

This insert presents three different commands for nonparametric regression:

1. `kernreg`: a kernel regression command that uses an interpolation algorithm;
2. `warpreg`: a command that approximates kernel regression using the WARP method; and
3. `knnreg`: a command to calculate k-nearest neighbor (k-NN) estimates.

In addition, we present `gwarpreg`, a program that aids in selecting the appropriate bandwidth for kernel regression.

Background

According to Silverman (1985), regression analysis aids in exploring and displaying bivariate relationships, provides predictions, and enables one to uncover interesting properties of the resulting equation. In this context, a nonparametric estimator is desirable because it does not force the model into a rigidly defined class, but, rather, lets the data "speak for themselves" in choosing the form of the model. In some instances, the nonparametric estimate may suggest a suitable parametric model (such as linear regression); in others, it may be the way to discover a very complex nonlinear mean function.

To illustrate the usefulness of nonparametric regression techniques, we examine three example data sets. The first contains the familiar data on the Old Faithful geyser (Weisberg 1980, Silverman 1985, Härdle 1991). Figure 1 displays a scatterplot of the waiting time until the next eruption against the duration of the previous eruption. Superimposed is the estimated, nonlinear relationship calculated by our kernel regression program. At this stage, we note only the striking two-level pattern suggested by the estimates, and we leave a fuller discussion until later in this insert.

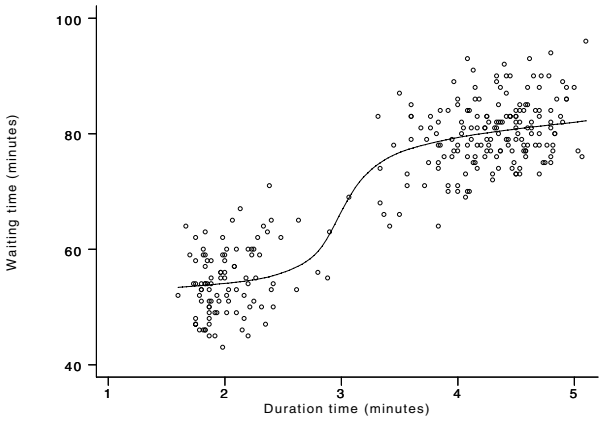

Figure 1. The geyser data with a kernel regression fit

The second example data set comes from an experiment on the efficacy of crash helmets during a simulated motorcycle crash and presents a more complex modeling challenge (Silverman 1985). Figure 2 displays a scatterplot of accelerometer readings against time, again with a smooth curve estimated by kernel regression. The estimated relationship gives a clear indication of the general tendency of the data, although it does not follow the first bend accurately enough and it displays perhaps too much variability at longer time values.

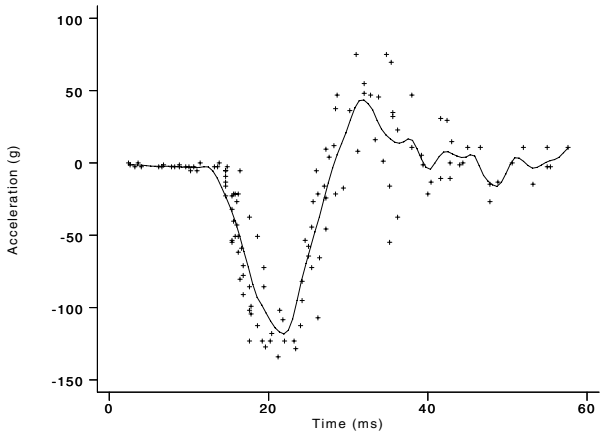

Figure 2. The motorcycle data with a kernel regression fit

Finally, we consider a simulated data set that characterizes the growth cycle of fishes. We generated simulated data that follow a modified von Bertalanffy growth function (VBGF) with error. This function, described by Pauly et al. (1992), uses a combination of periodic functions to account for the seasonal cessation of growth in fishes. In other words, while fish grow larger with age, the rate of growth ebbs seasonally. In our simulations, we used parameter values estimated by Pauly et al. from data on the growth of the Atlantic salmon (*Salmo salar*) in a Scottish stream (Egglishaw 1970). The simulation ado file is available from the first author upon request.

Figure 3 displays a scatterplot of fish length against age for these simulated data along with a smooth line that depicts the true VBGF for these data and a dashed line that displays the kernel regression estimate of the VBGF. The kernel estimate is very close to the functional response, although it deviates where the X values are scarce and where there are influential Y values.

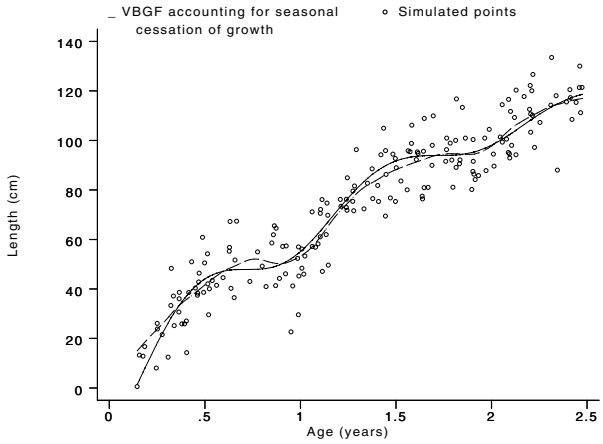

Figure 3. Simulated fish growth data with a kernel regression fit

The traditional approach

To motivate the nonparametric regression estimator, we apply the classical regression model to the geyser data. We begin by reviewing some familiar features of the classical model.

The regression of Y_i on X_i can be written as

$$Y_i = m(X_i) + \epsilon_i \tag{1}$$

where $m()$ is the unknown regression function and ϵ_i is the unobserved, random disturbance that causes Y_i to deviate from $m(X_i)$ (Härdle 1990, 1991; Scott 1992). Under the usual assumptions, the regression function is the conditional mean of Y_i, that is,

$$m(x) = E[Y|X = x]$$

This conditional mean curve may be represented by

$$E[Y|X = x] = \frac{\int yf(x,y)dy}{f(x)} \tag{2}$$

where $f(x,y)$ is the joint density of (X,Y) and $f(x)$ is the marginal density of X. Of course, when the joint distribution is Gaussian, the regression curve is linear (Härdle 1991).

In Figure 1, the scatterplot of the geyser data, we can readily discern not only the two clusters of points, but also a proportional relationship between the waiting time until the next eruption and the duration of the previous eruption. We can use classical regression to quantify this relationship. Following Cook and Weisberg (1982), Silverman (1985) and Härdle (1991), we start with the linear model

$$Y_i = \mu + \alpha X_i + \epsilon_i \tag{3}$$

```
. use geyser, clear
. describe
Contains data from geyser.dta
  Obs:   272 (max= 71392)
 Vars:     2 (max=    99)
Width:     8 (max=   200)
  1. dura         float  %9.0g
  2. wait         float  %9.0g
Sorted by:
r; t=0.06 12:27:33
. regress wait dura
  Source |       SS       df       MS                  Number of obs =     272
---------+------------------------------               F(  1,   270) = 1162.17
   Model |   40644.437     1   40644.437               Prob > F      =  0.0000
Residual |  9442.68066   270  34.9728913               R-squared     =  0.8115
---------+------------------------------               Adj R-squared =  0.8108
   Total |  50087.1176   271  184.823312               Root MSE      =  5.9138
------------------------------------------------------------------------------
    wait |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    dura |   10.72958    .314737     34.091   0.000       10.10993    11.34923
   _cons |   33.47437   1.154822     28.987   0.000       31.20077    35.74798
------------------------------------------------------------------------------
```

This listing reveals a significant regression with a high coefficient of determination. Figure 4 redisplays the data with the fit from this regression shown as the long straight line. Cook and Weisberg noted some inconsistencies with this simple model (as indicated by some curvature in a residual plot) and suggested that at large values of duration the predicted waiting times might be too large. The fit from the kernel regression, displayed in Figure 1, clearly captures this feature and, in addition, suggests an alternative parametric model that takes into consideration the observed clusters of points. This alternative model fits the two clusters by separate regression lines. This model can be written as

$$Y_i = \mu + \alpha X_i I(X_i \leq 3.3) + \beta X_i I(X > 3.3) + \gamma I(X_i > 3.3) + \epsilon_i \tag{4}$$

In this model, the clusters are separated by a partition of the x-axis at a duration of 3.3 minutes. In this parameterization, μ is the intercept and α is the slope of the relationship for the left-hand cluster, while $\mu + \gamma$ is the intercept and β is the slope for the right-hand cluster.

(*Graph on next page*)

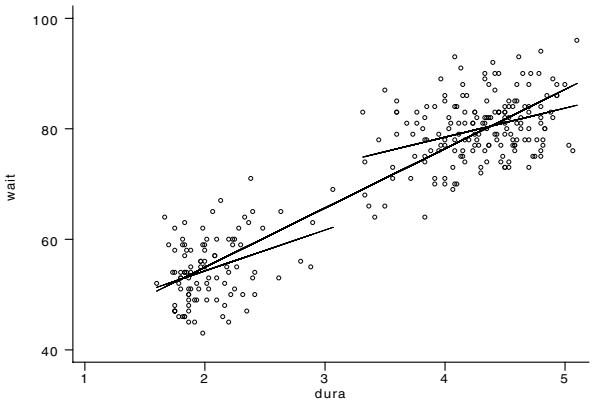

Figure 4. The geyser data with two linear fits

To estimate this model in Stata, we create a dummy variable to mark each cluster and we divide the duration time variable into two variables, one for each cluster.

```
. generate duralo = 0
. generate durahi = 0
. generate high = 0
. replace duralo = dura if dura <= 3.3
(98 real changes made)
. replace durahi = dura if dura > 3.3
(174 real changes made)
. replace high = 1 if dura > 3.3
(174 real changes made)
. regress wait duralo durahi high
  Source |       SS       df       MS                  Number of obs =     272
---------+------------------------------                F(  3,   268) =  442.34
   Model |  41671.3234     3  13890.4411                Prob > F      =  0.0000
Residual |  8415.79429   268  31.4022175                R-squared     =  0.8320
---------+------------------------------                Adj R-squared =  0.8301
   Total |  50087.1176   271  184.823312                Root MSE      =  5.6038
------------------------------------------------------------------------------
    wait |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  duralo |   7.384168    1.99572      3.700   0.000       3.454885    11.31345
  durahi |   5.247466   1.061537      4.943   0.000       3.157454    7.337478
    high |    17.9808   6.167388      2.915   0.004       5.838112     30.1235
   _cons |   39.51536    4.12751      9.574   0.000       31.38889    47.64183
------------------------------------------------------------------------------
```

Again, we have a highly significant regression, although the slopes of the separate regression lines are both shallower than the slope of the regression line in the simpler model. The two, shorter lines in Figure 4 display these separate linear fits.

Since this alternative model nests the original model, it is straightforward to test the restrictions implied by the original model. In this case, these restrictions specify that the intercepts and slopes of the two regression lines are equal. We use Stata's `test` command to calculate the F test.

```
. test high
 ( 1)  high = 0.0
       F(  1,   268) =    8.50
            Prob > F =    0.0039
```

```
. test duralo=durahi, accum
 ( 1)  high = 0.0
 ( 2)  duralo - durahi = 0.0
       F(  2,   268) =   16.35
            Prob > F =    0.0000

. test duralo=durahi
 ( 1)  duralo - durahi = 0.0
       F(  1,   268) =    0.89
            Prob > F =    0.3454
```

These tests indicate that the joint restrictions on the slopes and intercepts are overwhelmingly rejected by the data. The restriction of equal intercepts is also rejected at the 1 percent level, as can be seen directly by examining the t statistic of `high` in the regression output. However, the null hypothesis that the slopes of the two lines are equal is not rejected.

Silverman (1985) emphasizes that a nonparametric curve can be an important exploratory step towards the final model choice. For the geyser data, the kernel regression fit clearly indicated the type of segmented model that the data appear to favor. However, as the motorcycle and simulated growth data show, the leap from a sound nonparametric fit to an improved parametric model is not always this easy.

Kernel regression: the Nadaraya–Watson estimator

The nonparametric approach to regression is related to the nonparametric approach to density estimation. In estimating densities at a given x value, the points in a small neighborhood around x are considered (the neighborhood is defined by the bin width of the histogram or the bandwidth h of the kernel density estimator). In estimating a regression, the response variable Y is weighted in a neighborhood of x. Using a simplified notation, the general form of the nonparametric regression can be written as

$$\widehat{m}_h(x) = n^{-1} \sum_{i=1}^{n} W_{hi}(x) Y_i \tag{5}$$

This expression, applicable to most of the nonparametric regression methods, can be understood as a weighted average of the response variable Y_i where the weights, $W_{hi}(x)$, depend on the particular procedure, on the distance between x and X_i and on the smoothing parameter, h.

Recall that the usual expression for the regression function is

$$m(x) = E[Y|X = x] = \frac{\int y f(x, y) dy}{f(x)} \tag{6}$$

The denominator of the expression on the right is a density which can be estimated nonparametrically by means of a kernel density estimator. Similarly, the joint density in the numerator can be estimated using the multiplicative kernel (see Härdle 1991 for details). If we use the same bandwidth, h, for both estimates, we can substitute the kernel estimates for the densities. Thus we arrive at the following general expression, proposed independently by Nadaraya (1964) and Watson (1964).

$$\widehat{m}_h(x) = n^{-1} \frac{\sum_{i=1}^{n} K_h(x - X_i) Y_i}{n^{-1} \sum_{j=1}^{n} K_h(x - X_j)} \tag{7}$$

Härdle (1991) notes several points about this estimator:

1. the weights depend on the entire X sample through the kernel density estimate $f_h(x)$;

2. observations on Y_i are more influential where X_i are sparse;

3. when the denominator is zero, so is the numerator, and the estimate is defined to be zero as well;

4. as $h \to 0$, X_i converges to Y_i, producing an interpolation of the data;

5. as $h \to \infty$, the weighting function converge to 1 for all values of x, and the estimate converges to the mean of Y;

6. as in kernel density estimation, h determines the smoothness of the estimate.

Calculating kernel regressions in Stata

In previous inserts (Salgado-Ugarte et al. 1993, 1995a, 1995b), we introduced some ado-files to estimate univariate densities using kernel functions. It was straightforward to modify these programs to estimate the numerator and denominator of equation (7). Our implementation is called `kernreg.ado`. The syntax for this command is

`kernreg` *yvar xvar* [`if` *exp*] [`in` *range*] , `bwidth(#) kercode(#) npoint(#)`

[`gen(`*fitvar gridvar*`) nograph` *graph-options*]

There are three required options. `bwidth(#)` specifies h, the smoothing parameter. `npoint(#)` specifies the number of equally spaced points (the grid) to use for calculating the kernel estimates. `kercode(#)` specifies the kernel according to the following codes:

Code	Kernel
1	uniform
2	triangular
3	Epanechnikov
4	quartic (biweight)
5	triweight
6	Gaussian
7	Cosinus

The option `gen(`*fitvar gridvar*`)` allows the user to create two new variables containing the fitted values from the kernel regression and their corresponding grid points, respectively. The `nograph` option suppresses the graph, produced by default, of the fit against the x values. The appearance of this graph can be modified using the options of the `graph` command.

The kernel fits displayed in Figures 1 through 3 were calculated by `kernreg`. In each case, the quartic kernel was used with a grid of 100 points. The bandwidths used were 0.8 for the geyser data, 2.4 for the motorcycle data, and 0.22 for the simulated growth data.

Figures 5 through 7 display the effect on the fits of changes in the bandwidth. Figure 5 displays the geyser data with three kernel regression fits. The line composed of long dashes is the fit calculated with $h = 0.8$, the line composed of short dashes is the fit calculated with $h = 0.4$, and the solid line is the fit calculated with $h = 0.1$. Just as in univariate density estimation, increases in the smoothing parameter reduce the noise in the kernel regression estimates. In this example, setting $h = 0.1$ generates a very rough estimate with one missing value, indicated by the discontinuity in the graph. Nevertheless, all three curves give the same qualitative message—a slightly increasing average response within each cluster of data.

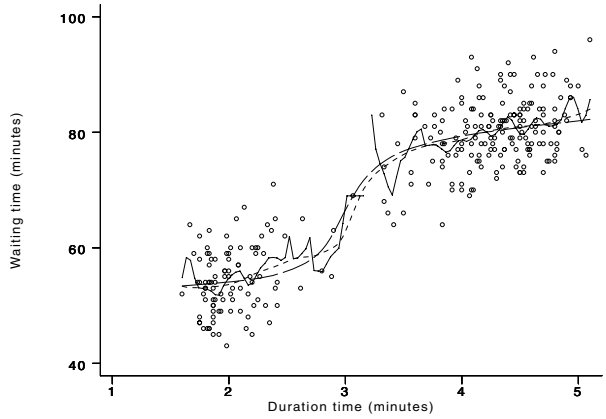

Figure 5. The geyser data with three kernel fits

It is worth remembering that the error increases as the grid becomes coarser. Therefore, the analyst must take care to employ a grid that is fine enough. The suggested value of 100 grid points should be adequate in many instances, but if the range of x values is very large, it may be necessary to try other values for `npoint()`. Linearly interpolating between fitted points also produces more accurate results. (See Jones 1989 for a full discussion of this topic in the context of univariate density estimation.)

Figure 6 displays the motorcycle data with three kernel fits ($h = 1, 2.4, 5$). In this instance, $h = 2.4$ produces a reasonable fit, while smaller and larger bandwidths produce noisy and oversmoothed curves, respectively.

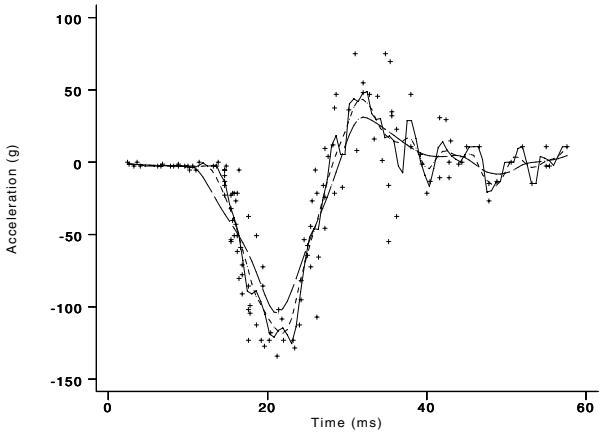

Figure 6. The motorcycle data with three kernel fits

Figure 7 displays the simulated growth data with three kernel fits ($h = 0.1, 0.2, 0.5$). The fit with the highest bandwidth is very close to the true, seasonally adjusted VBGF, while the fit with $h = 0.22$ exhibits cycles similar to those of the common VBGF. The algorithm that calculates the adjusted VBGF is itself quite long and complex, thus it is notable that these relatively simple kernel fits do such a good job of recovering the underlying function.

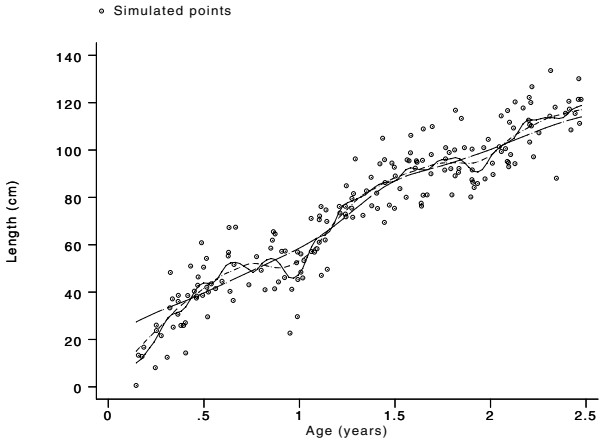

Figure 7. The simulated growth data with three kernel fits

Approximating kernel regression with WARP regression

As we noted in our earlier inserts, the direct calculation of kernel estimates is computer-intensive. However, the WARP technique provides an excellent approximation to kernel estimates with a substantial savings in computer time. Since the Nadaraya–Watson regression estimator is a combination of kernel univariate densities, the gains from using the WARP are substantial.

The outlines of applying the WARP approach to Nadaraya–Watson regression are straightforward. (The interested reader is directed to Härdle 1991 and Scott 1992 for details.) Partition the x-axis into bins and define an index function that returns the index of the bin to which each x belongs:

$$l(x) = j \Leftrightarrow x \in [(j - 1/2)\delta, (j + 1/2)\delta)$$

The sum of the observations on the response variable corresponding to X_i in bin B_z is

$$Y_{\bullet z} = \sum_{i=1}^{n} Y_i I(X_i \in B_z)$$

Thus, for $x \in B_{l(x)}$

$$\widehat{m}_{M,K}(x) = \frac{\sum_{l=1-M}^{M-1} K\left(\frac{l}{M}\right) Y_{\bullet l(x)+l}}{\sum_{l=1-M}^{M-1} K\left(\frac{l}{M}\right) n_{l(x)+l}} \tag{8}$$

where M is the number of shifted histograms to average (see Salgado-Ugarte et al. 1995a for details).

We have implemented a WARP approach to kernel regression in a manner similar to the way we applied WARP to kernel density estimation in *snp6.1*. As before, our programs are based on the algorithms and programs described in Härdle (1991) and Scott (1992). We also consulted revised programs in C based on Härdle and obtained by EMAIL from Statlib. Also as before, our Stata program, `warpreg`, embeds a system call to an executable Turbo Pascal program, `warpregf.exe`. As a consequence, `warpreg` is currently available only for DOS-based platforms. However, the Pascal source code is supplied on the STB distribution diskette to enable interested readers to adapt `warpreg` for use on other platforms.

The syntax of `warpreg` is

`warpreg` *yvar xvar* [`if` *exp*] [`in` *range*] , `bwidth(#) kercode(#) mval(#)`

[`gen(`*fitvar midvar*`) nograph sort` *graph-options*]

There are three required options. `bwidth(#)` specifies h, the smoothing parameter. `mval(#)` specifies M, the number of shifted histograms averaged in calculating the density estimates. `kercode(#)` specifies the kernel using the same codes as `kernreg`, although code 7 (cosinus) is not supported for `warpreg`.

The option `gen(`*fitvar midvar*`)` allows the user the create two new variables containing the fitted values from the WARP regression and the corresponding midpoints from the bins, respectively. The `sort` option informs `warpreg` that the data are already sorted by *xvar* and *yvar*. The graph, produced by default, can be modified with graph options or suppressed by the `nograph` option.

Figure 8 displays the graph `warpreg` produces for the geyser data. In this example, $h = 0.4$, the `mval` (M) is 10, and the quartic kernel is used. the qualitative appearance is similar to that produced by `kernreg`, although the details differ. The points are connected by linear interpolation (the polygon version). Since the corresponding kernel estimate is displayed as a continuous function, the polygon version is preferable (Härdle 1991). It has been shown that the interpolated version displayed by `warpreg` is more accurate than a discretized, step representation (Jones 1989).

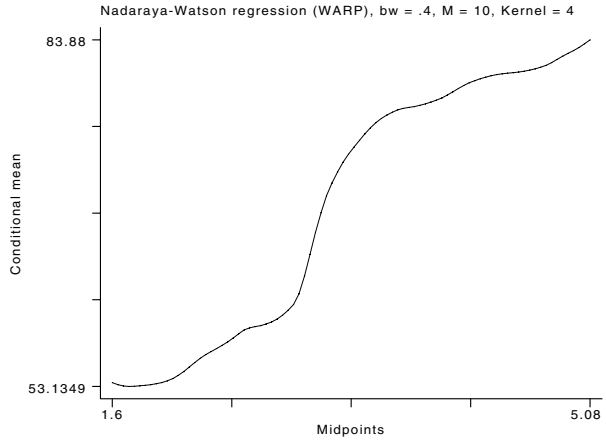

Figure 8. A WARP fit to the geyser data

As in the WARP approach to density estimation, the WARP approximation to the kernel estimate improves as M, the number of average shifted histograms, increases. Figure 9 displays three different WARP approximations ($M = 1, 5, 10$) along with the kernel regression. For the geyser data and $M \geq 5$, there is no appreciable difference between the WARP and the kernel estimates. However, the WARP approximations are calculated much more quickly than the kernel estimate.

(*Figure 9 on next page*)

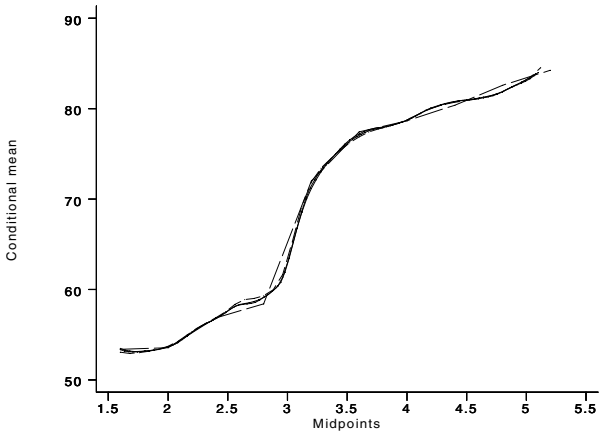

Figure 9. Kernel and WARP fits to the geyser data

Table 1 displays some relative timings for kernel regression and WARP regression. The first column of the table indicates which data set was used. The second column indicates the smoothing parameter used. The final two columns report the execution times, in seconds, for `kernreg` and `warpreg`, respectively. In these examples, `warpreg` is one to two orders of magnitude faster than `kernreg`.

Table 1. Timing comparisons of kernel and WARP regression programs

Data set	h	`kernreg`	`warpreg`
Geyser	0.40	388	11
	0.65	410	11
	0.80	420	11
	1.75	475	10
Motorcycle	1.00	174	9
	2.40	179	7
	5.00	188	6

The speed of `warpreg` is particularly impressive since `warpreg` writes and reads several temporary files and calls an external program for most of the calculations. Note that the execution time of `kernreg` increases with h, while `warpreg` tends to speed up as h rises.

Bandwidth selection

A key step in nonparametric regression is the selection of a value for h, the smoothing parameter. Choosing this parameter is closely related to the problems of selecting the degree of a polynomial regression or of selecting the variables to include in a multiple regression. In all these cases, there is a trade-off between bias and variance; a good compromise must avoid both a curve that follows every data point (overfitting) and a curve that misses important features of the data (oversmoothing) (Altman 1992).

As in the case of density estimation, several methods for choosing the bandwidth for kernel regression have been proposed. The simplest is the subjective approach: the analyst tries several values for h and chooses the one that yields the most interesting estimates. This procedure is highly flexible and is used with beginning students, but objective methods are preferred in automated estimation systems. In addition, objective methods promote consistency of results among investigators (Altman 1992). Many different objective methods have been suggested. Here, we present `gwarpreg`, a Stata program that combines penalty functions with cross-validation to select a bandwidth. This program incorporates algorithms that are modifications of those in Härdle (1991).

Before we demonstrate `gwarpreg`, it may be useful to review some of the motivation for this approach to bandwidth selection. In density estimation, the basic approach is to consider a sequence of bandwidths and to choose the one that minimizes the asymptotic mean integrated squared error (A-MISE) of the estimate. In regression, though, there are other potential measures of the discrepancy between the estimator, $\widehat{m}(X_i)$, and the true regression curve, $m(X_i)$. Among these are the average squared error, the integrated squared error, the conditioned square error, and the mean integrated squared error. It can be shown (Härdle

and Marron 1986) that all of these measures lead asymptotically to the same bandwidth estimator. Thus, for convenience, it is recommended to use the most easily-computed measure, the averaged squared error ASE (Härdle, 1990, 1991). However, due to difficulties in minimizing this function, the technique usually employed is a combination of bias-correction terms and the leave-one-out (cross-validation) estimate, $\widehat{m}_{hi}(X_i)$. This technique employs one of the penalty functions listed in Table 2.

Table 2. Penalty functions used in bandwidth selection
(adapted from Härdle 1991)

Penalty function	General form
Shibata's model selector (Shibata 1981)	$S(u) = 1 + 2u$
Generalized cross validation (Craven and Wahba 1979) (Li 1985)	$GCV(u) = \frac{1}{(1-u)^2}$
Akaike's information criterion (Akaike 1970)	$AIC(u) = e^{2u}$
Finite prediction error (Akaike 1974)	$FPE(u) = \frac{1+u}{1-u}$
Rice's T (Rice 1984)	$T(u) = \frac{1}{1-2u}$

The purpose of the penalty function is to avoid overfitting by penalizing bandwidths that are too small. The adjusted prediction error, $G(h)$, is calculated as the sum of the prediction error (derived from the ASE) and a correction term (obtained from the penalty function). Each penalty function gives different weights to the bias and variance of $\widehat{m}_h$. For instance, Shibata's S places greater weight on reducing the bias, while Rice's T emphasizes reducing the variance. Despite these differences, in practice each of these penalty functions leads to substantially the same choice of optimal bandwidth. Convergence to the optimum is known to be slow with this method, however, it is not possible to derive a more efficient approach, and the selected bandwidth often works well even for moderate sample sizes (Härdle et al. 1988, Härdle 1990, Altman 1992).

An algorithm to calculate $G(h)$ directly is easy to implement, but with the great disadvantage that the number of steps for a specific h are quadratic in the number of observations, that is, the algorithm is $O(n^2)$ (Härdle 1991). Fortunately, the WARP approximation can be applied to this problem as well, and it provides a much more efficient algorithm for calculating h.

In the WARP setting, choosing the h that minimizes the ASE is equivalent to choosing the optimal M, that is, the optimal number of shifted histograms to average. To do this, it is necessary to assume a tolerance, δ, that measures the accuracy of the data.

Defining $W_{Mi}[z]$ and $\widehat{m}_M[z]$ as the values of $W_{Mi}[x]$ and $\widehat{m}_M[x]$ in bin z, and assuming that the weight function $w(\bullet)$ is constant within each bin, we obtain

$$Y^2_{\bullet z} = \sum_{i=1}^{n} Y_i^2 I(X_i \in B_z) \tag{9}$$

The general expression for $G(M)$, the penalty function for WARP regression, is (Härdle, 1991)

$$G(M) = n^{-1} \sum_{z \in Z} \left\{ Y^2_{\bullet z} - 2\widehat{m}_M[z] Y_{\bullet z} + \widehat{m}^2_M[z] n_z \right\} \Xi(n^{-1} W_M[z]) w[z] \tag{10}$$

As in the cross-validation programs for optimal bandwidth selection for density estimation (Salgado-Ugarte et al. 1995b), the algorithm contains the typical steps of the WARP method slightly modified: (1) bin the data, (2) generate the weights, and (3) weight the bins. The main difference is that, after binning the data, there is an additional loop over a range of potential values for M (δ is held constant). This loop includes the generation of weights and a modified version of weighting the nonempty bins and their nonempty neighbors. This procedure, including the extra loop, is linear in the number observations, in contrast to the direct algorithm which is quadratic in n (Härdle 1991).

`gwarpreg`, our program for carrying out this procedure for bandwidth selection, is a modification of the algorithm and programs of Härdle (1991). As in `warpreg`, a system call to an executable Turbo Pascal program is embedded in `gwarpreg`. As a consequence, `gwarpreg` is currently available only for DOS-based platforms. (See the discussion of this issue in the section above on `warpreg`.) The syntax of `gwarpreg` is

`gwarpreg` *yvar xvar* [`if` *exp*] [`in` *range*] , `delta(#)` `kercode(#)` `select(#)`

[`bound(#)` `gen(`*score M bandwidth*`)` `mstart(#)` `mend(#)`]

There are three required options. `delta(#)` specifies the tolerance for estimation. `kercode(#)` specifies the kernel using the same codes as `warpreg`. `select(#)` specifies the penalty function according to the following codes

Code	Penalty function
1	Shibata's S
2	Generalized cross validation
3	Akaike's information criterion
4	Finite prediction error
5	Rice's T

`bound(#)` specifies the proportion of the data to be trimmed prior to the calculations. The default is `bound(.1)`, that is, the most extreme 10 percent of the data is trimmed to reduce the influence of extreme observations. According to Härdle (1990), adjusting this option generally has negligible results on the estimated bandwidth.

`mstart(#)` and `mend(#)` define the range of M values to try. By default, `mstart` is 5 and `mend` is the integer part of $0.3(\max x - \min x)/\delta$.

The `gen()` option allows the user to store the score $G(M)$, M, and the corresponding bandwidth in three new variables. This option is useful in creating customized graphs of the results.

`gwarpreg` produces two graphs. The first graph displays the score, $G(M)$, as a function of M. The second graph displays the score as a function of h, the bandwidth. These graphs allow the user to find the interval that contains the minimum score (and, hence, the optimal bandwidth) by trial and error. After the graphs are displayed, `gwarpreg` lists the five smallest values of the score along with the corresponding values of M and h. By varying the value of δ and the range for M, users can satisfy themselves they have located the global minimum of the score.

As an example, we apply `gwarpreg` to the motorcycle data using the Shibata S penalty function and the quartic kernel.

```
. gwarpreg accel time, delta(0.2) select(1) kercode(4) mend(20)
```

(two graphs appear, not shown)

```
------------------------------------------------------------------------
Adjusted Prediction error G(M) for WARPing Nadaraya-Watson regression
------------------------------------------------------------------------
M-value = 8        Score value = 534.06671       Bandwidth =   1.6000
M-value = 9        Score value = 537.53998       Bandwidth =   1.8000
M-value = 7        Score value = 539.06793       Bandwidth =   1.4000
M-value = 10       Score value = 542.87994       Bandwidth =   2.0000
M-value = 11       Score value = 548.75659       Bandwidth =   2.2000
```

This example suggests setting $h = 1.6$. As a check on this result, we reran `gwarpreg` with each penalty function. Table 3 summarizes the results. Aside from δ, all other inputs were held constant.

Table 3. Bandwidths suggested by different penalty functions

Penalty function	δ	h
Shibata's S	0.2	1.6
Generalized cross validation	0.3	2.4
Akaike's information criterion	0.2	1.8
Finite prediction error	0.3	2.1
Rice's T	0.3	2.7

Figure 10 displays the score as a function of M for these data for each penalty function. The curves show similar trends with one well defined minimum, except for Rice's T, which suggests the existence of a local minimum.

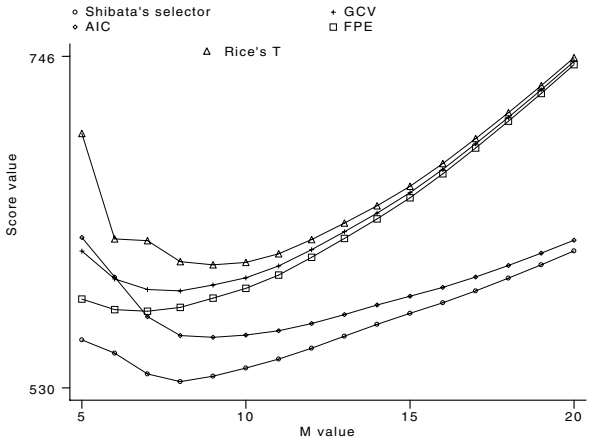

Figure 10. Score versus M for the motorcycle data

We also applied `gwarpreg` to the geyser data and the simulated growth data. For the geyser data, `gwarpreg` identified two local minima in the score function associated with bandwidths of 0.65 and 1.75. This result is consistent with findings of Härdle for these data.

The results for the growth simulation data turned out to depend sensitively on the choice of penalty function. Shibata's S and Akaike's information criterion did not identify a clear minimum in the score function. The other three penalty functions indicated a minimum score with an associated bandwidth between 0.20 and 0.22.

In all three examples, the bandwidth chosen by `gwarpreg` seems to be a good choice, producing a smooth regression curve that captures the main features of the original function. This estimated bandwidth can also be used as a reference for comparison with noisier (smaller bandwidth) or smoother (larger bandwidth) alternatives.

Other methods: k-nearest neighbor estimates

Kernel regression is not the only type of nonparametric regression. Another approach is k-nearest neighbor (k-NN) regression. Kernel regression calculates local averages of the observations Y_i within a fixed neighborhood (bandwidth) of x. In contrast, k-NN regression holds the sample size (span) of each local average constant and allows the bandwidth to fluctuate.

The k-NN regression estimate is defined as

$$\widehat{m}_k(x) = n^{-1} \sum_{i=1}^{n} W_{ki}(x) Y_i \tag{11}$$

The weight sequence is defined, using Härdle's (1991) notation, through the set of indices

$$\{W_{ki}(x)\}_{i=1}^{n} = J_x = \{i | X_i \text{ is one of the } k \text{ nearest observations to } x\} \tag{12}$$

and the k-NN weight sequence defined from this set of neighboring observations is

$$W_{ki}(x) = \begin{cases} n/k, & \text{if } i \in J_x; \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

In this procedure the smoothing parameter is k, the number of neighbor points and, as in kernel regression, the choice of k involves a tradeoff between bias and variance. A larger k produces smoother estimates but at the cost of an increase in bias. There are two interesting extreme cases. When $k = n$, the estimate is simply the average of the response variable. When $k = 1$, the estimate is a step function that matches every response value.

It is convenient to define symmetric k-NN estimates, that is, to use the averages of the $k/2$ observations to the left and the $k/2$ observations to the right of each Y_i. If the data are sorted in the order of increasing X, the k-NN estimate can be computed recursively using the following formula (Härdle 1990, 1991)

$$\widehat{m}_k(X_{i+1}) = m_k(X_i) + k^{-1} \left(Y_{i+[k/2]+1} - Y_{i-[k/2]} \right) \tag{14}$$

for $i = [k/2] + 1, \ldots, n - [k/2]$, where $[u]$ denotes the greatest integer less than or equal to u. This algorithm is linear in the number of observations. This algorithm can also be used to update local polynomial fits. Therefore, it can be used to implement locally weighted scatter plot smooths such as LOWESS at a very modest computational cost (Härdle 1990, Cook and Weisberg 1994).

`knnreg` is a simple implementation of the k-NN estimator that incorporates the suggestion to sort the data on X and Y (XploRe Systems 1993). The syntax of `knnreg` is

`knnreg` *yvar xvar* [`if` *exp*] [`in` *range*] , `knum(#)` [`gen(`*mkvar*`)` `nograph` *graph-options*]

where `knum(#)` specifies the number of neighbors to use. The `gen(`*mkvar*`)` option creates a new variable that contains the estimates. The new variable contains missing values corresponding to both extremes of the sorted response values, as it is based on differences of lagged values for calculation, without endpoint adjustment.

By default, `knnreg` displays a scatterplot of the data along with the estimated regression. To illustrate `knnreg`, we present the results obtained for each of the three example data sets using two different values for k. Figures 11–16 display these results.

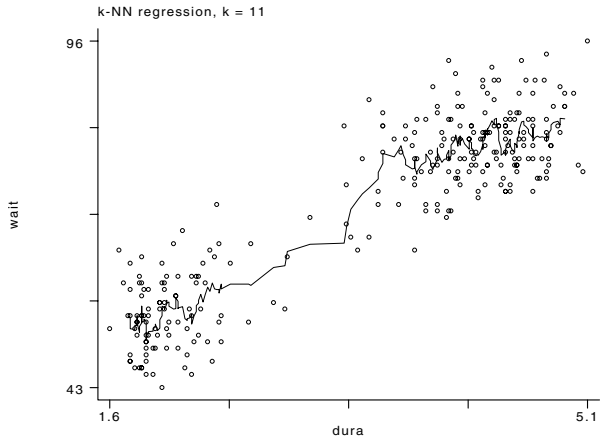

Figure 11. Geyser data, $k = 11$

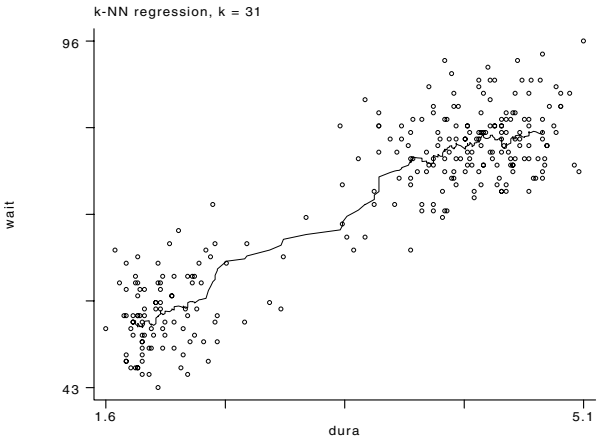

Figure 12. Geyser data, $k = 31$

Figures 11 and 12 display the geyser data along with k-NN regressions for $k = 11$ and $k = 31$, respectively. These estimates are more ragged than the kernel regressions. Both these estimates indicate the existence of two groups with a slightly increasing slope.

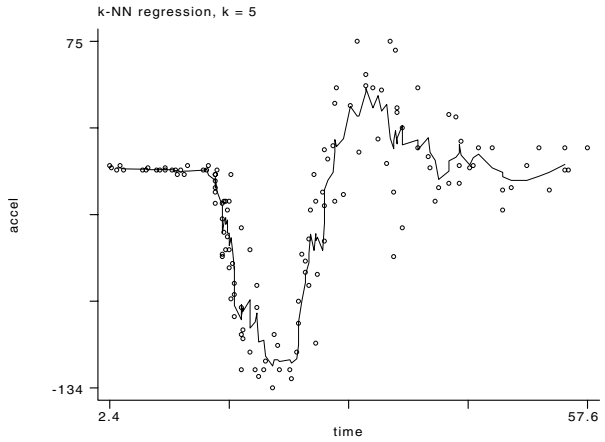

Figure 13. Motorcycle data, $k = 5$

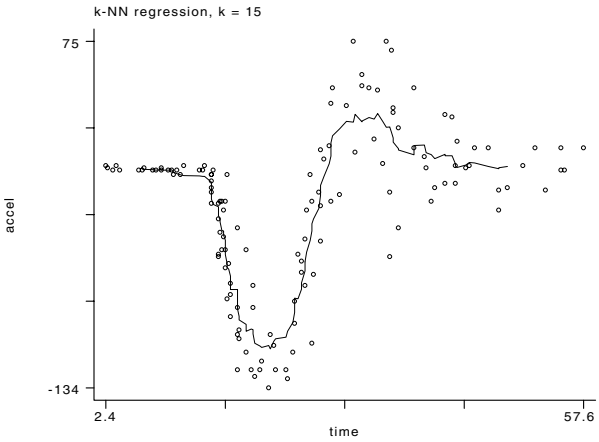

Figure 14. Motorcycle data, $k = 15$

The motorcycle data are displayed in Figures 13 and 14 along with k-NN regressions for $k = 5$ and $k = 15$, respectively. In this instance, the k-NN regressions perform somewhat better than the kernel regressions, especially at the beginning of the series and at the bends in the data.

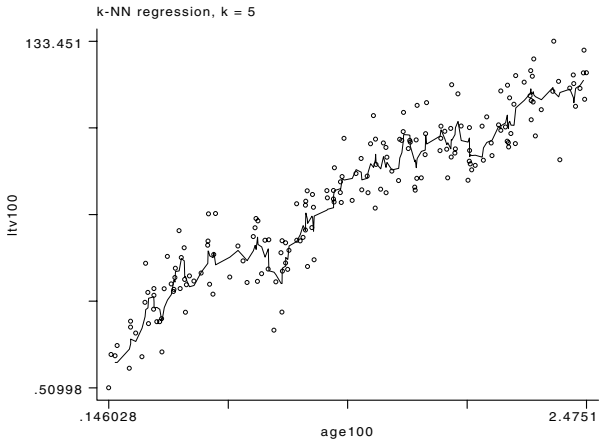

Figure 15. Simulated growth data, $k = 5$

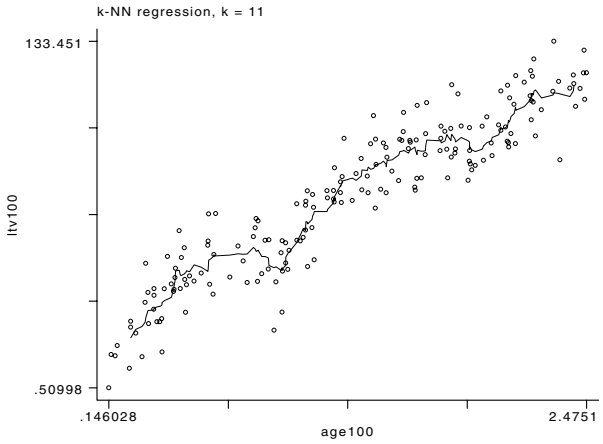

Figure 16. Simulated growth data, $k = 11$

The simulated growth data are displayed in Figures 15 and 16 along with k-NN regressions for $k = 5$ and $k = 11$, respectively. For these data, the k-NN regressions do a good job of revealing the seasonal pattern of the original function, with a close correspondence to the true number of cycles (three). One weakness of these estimates is the implication of intermittent body shrinkage, an unrealistic implication for finfish or shellfish with a hard skeleton. Nevertheless, the estimates are quite good at recovering important features of the simulated points.

These nonparametric regression estimators, both the kernel and k-NN versions, are important tools for the statistician. Their flexibility make them useful for exploring complex bivariate relationships. Following estimation, analytic features of the regression—such as derivatives, extrema, and roots—can be calculated. Moreover, nonparametric regressions can be used in combination with parametric models in the iterative process of model building and checking (Altman 1992). In their role as smoothers, nonparametric regressions can be used in the estimation of generalized additive models (Hastie and Tibshirani 1990) in high-dimensional regression problems (Scott 1992, Cook and Weisberg 1994).

Additional nonparametric estimators

Stata provides additional nonparametric regression estimators.

Local location estimators. The simplest nonparametric regression procedures are local versions of location estimators. Stata's `graph` command allows you to connect cross-medians within fixed-width bands of scatterplots. The `ksm` command without options (other than the bandwidth) calculates running mean smooths, while `ksm` with the weight option calculates running mean smooths with tricube weighting.

Spline smoothing. A spline is a linear smoother, closely related to kernel regression. Stata's `graph` command connects cross-medians with a cubic spline. In addition, the STB has published at least two spline commands (type `lookup spline` for more information).

Local regression smoothing. Stata's `ksm` command calculates locally unweighted and weighted (LOWESS) regression estimators.

Resistant nonlinear smoothers. Stata's `smooth` command calculates a variety of resistant, compound nonlinear smoothers.

For details on these commands, consult the Stata Reference Manual. The user is invited to compare these commands and graphing options to the programs presented in this insert.

Some final notes

`kernreg` and `knnreg` can be used by all Stata users. As noted above, though, `warpreg` and `gwarpreg` can be used only on DOS platforms because of these programs' reliance on executable Pascal programs. Unix users can recompile the Pascal source or convert the programs to C. The Windows versions of the ado files presented are available from the first author upon request.

Härdle's (1991) algorithms and programs formed the basis of all the Pascal routines provided with this insert. As implemented, these routines can handle 1000 observations and small bins.

When questions about the precise definitions of the algorithms arose, we used listings of S functions and C source code obtained from Statlib as a guide.

The results of our programs were verified by comparison with the graphs and tables in Härdle (1991), the results from the

S functions implemented on the mainframe of the University of Tokyo (those not requiring dynamic loading) and with listings of the results of the S functions that require dynamic loading that were provided by Dr. Brian Ripley. We also made some comparisons with the results obtained with XploRe (version 3.1).

We would be grateful to hear of any problems users may have in using our programs.

Acknowledgments

The complete set of S functions and C source code from Härdle (1991) used to implement the WARP procedures in our ado files were obtained by EMAIL from Statlib. Brian Ripley kindly provided listings of the results of these functions for comparison and validation. With Dr. Joseph Hilbe's assistance, Dr. Härdle made available one copy of his program XploRe, which was used to perform additional result checking. The first author is grateful to the Ministry of Education, Science and Culture of Japan and to the Universidad Nacional Autónoma de México (FES Zaragoza and DGAPA) for their support.

References

Akaike, H. 1970. Statistical predictor information. *Annals of the Institute of Statistical Mathematics* 22: 203–217.

——. 1974. A new look at the statistical model identification. IEEE Transactions of Automatic Control AC, 19: 716–723.

Altman, N. S. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46: 175–185.

Cook, R. D., and S. Weisberg. 1982. *Residuals and influence in regression.* London: Chapman and Hall.

——. 1994. *An Introduction to Regression Graphics.* John Wiley & Sons. New York.

Craven, P. and G. Wahba. 1979. Smoothing noisy data with spline functions. *Numerical Mathematics* 31: 377–359.

Egglishaw, H. J. 1970. Production of salmon and trout in a stream in Scotland. *Journal of Fish Biology* 1: 117–136.

Härdle, W. 1990. *Applied nonparametric regression.* Econometric Society monograph series no. 19. New York: Cambridge University Press.

——. 1991. *Smoothing techniques. With implementation in S.* New York: Springer-Verlag. New York

Härdle, W., and J. S. Marron. 1986. Random approximations to an error criterion of nonparametric statistics. *Journal of Multivariate Analysis* 20: 91–113.

Härdle, W., P. Hall, and J. S. Marron. 1988. How far are automatically chosen regression smoothing parameters from their optimum. *Journal of the American Statistical Association* 83: 86–95.

Hastie, T. J. and R. J. Tibshirani. 1990. *Generalized additive models.* London: Chapman and Hall.

Hilbe, J. 1993. sg16: Generalized linear models. *Stata Technical Bulletin* 11: 20–28.

——. 1994a. sg16.5: Negative binomial regression. *Stata Technical Bulletin* 18: 2–5.

——. 1994b. sg22.1: Comment on Royston's revision of `glm`. *Stata Technical Bulletin* 18: 11–13.

Jones, M. C. 1989. Discretized and interpolated kernel density estimates. *Journal of the American Statistical Association* 84: 733–741.

Li, K-Ch. 1985. From Stein's unbiased risk estimates to the method of generalized cross validation. *Annals of Statistics* 13: 1352–1377.

Nadaraya, E. A. 1964. On estimating regression. *Theory of Probability and its Applications* 10: 186–190.

Pauly, D., M. Soriano-Bartz, J. Moreau, and A. Jarre-Teichmann. 1992. A new model accounting for seasonal cessation of growth in fishes. *Australian Journal of Marine and Freshwater Resources* 43: 1151–1156.

Rice, J. A. 1984. Bandwidth choice for non-parametric regression. *Annals of Statistics* 12: 1215–1230.

Royston, P. 1994a. sg22: Generalized linear models: revision of `glm`. *Stata Technical Bulletin* 18: 6–11.

——. 1994b. sg22.3: Generalized linear models: revision of `glm`. Rejoinder. *Stata Technical Bulletin* 19: 17.

Salgado-Ugarte, I. H., M. Shimizu, and T. Taniuchi. 1993. snp6: Exploring the shape of univariate data using kernel density estimators. *Stata Technical Bulletin* 16: 8–19.

——. 1995a. snp6.1: ASH, WARPing, and kernel density estimation for univariate data. *Stata Technical Bulletin* 26: 23–31.

——. 1995b. snp6.2: Practical rules for bandwidth selection in univariate density estimation. *Stata Technical Bulletin* 27: 5–19.

Scott, D. W. 1992. *Multivariate density estimation: Theory, practice, and visualization.* New York: John Wiley & Sons.

Shibata, R. 1981. An optimal selection of regression variables. *Biometrika* 68: 45–54.

Silverman, B. W. 1985. Some aspects of the spline smoothing approach to nonparametric regression curve fitting (with discussion). *Journal of the Royal Statistical Society, Series B* 47: 1–52.

Watson, G. S. 1964. Smooth regression analysis. *Sankhyä*, Series A 26: 359–372.

Weisberg, S. 1980. Applied linear regression. New York: John Wiley & Sons.

XploRe Systems. 1993. *XploRe—A Computing Environment for eXploratory Regression and Data Analysis.* Version 3.1. Institut für Statistik und Ökonometrie. Berlin.

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.

International Stata Distributors

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

Company:	Applied Statistics & Systems Consultants
Address:	14/26 Yizreel St., P.O. Box 1169 Nazerath-Ellit 17100, Israel
Phone:	+972 6554254
Fax:	+972 6554254
Email:	sasconsl@actcom.co.il
Countries served:	Israel

Company:	Dittrich & Partner Consulting
Address:	Prinzenstrasse 2 D-42697 Solingen Germany
Phone:	+49 212-3390 99
Fax:	+49 212-3390 90
Email:	available soon
Countries served:	Austria, Germany, Italy

Company:	Metrika Consulting
Address:	Roslagsgatan 15 113 55 Stockholm Sweden
Phone:	+46-708-163128
Fax:	+46-8-6122383
Email:	available soon
Countries served:	Baltic States, Denmark, Finland, Iceland, Norway, Sweden

Company:	Ritme Informatique
Address:	34 boulevard Haussmann 75009 Paris France
Phone:	+33 1 42 46 00 42
Fax:	+33 1 42 46 00 33
Email:	ritme.inf@applelink.apple.com
Countries served:	Belgium, France, Luxembourg, Switzerland

Company:	Smit Consult
Address:	Scheidingstraat 1 Postbox 220 5150 AE Drunen Netherlands
Phone:	+31 416-378 125
Fax:	+31 416-378 385
Email:	j.a.c.m.smit@smitcon.nl
Countries served:	Netherlands

Company:	Timberlake Consultants
Address:	47 Hartfield Crescent West Wickham Kent BR4 9DW, U.K.
Phone:	+44 181 462 0495
Fax:	+44 181 462 0493
Email:	100412.2603@compuserve.com
Countries served:	Ireland, U.K.

Company:	Timberlake Consultants Satellite Office
Address:	Rua Julião Quintinha, No. 5°–7° Esq. 1500 Lisbon Portugal
Phone:	+351 1 7140009
Fax:	+351 1 7140009
Email:	100412.2603@compuserve.com
Countries served:	Portugal