STATA
TECHNICAL
BULLETIN

November 1995
STB-28

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

crc41	New lfit, lroc, and lstat commands

The following new features have been added to the `lfit`, `lroc`, and `lstat` set of post-`logistic` commands:

1. `lfit` with the `group(#)` option no longer has any arbitrariness when dividing observations into quantiles for the Hosmer–Lemeshow goodness-of-fit test.
2. `lfit` with `group(#)` has a new option, `equal`, which prevents anomalies that could occur if there are large numbers of ties at the quantile boundaries.
3. `lfit` with `group(#)` now works when `fweight`s are used with `logistic`.
4. `lfit` with the `table` option produces more output; it displays predicted probabilities, observed and expected counts for both outcomes, and totals for each group.
5. The `rules` and `asif` options can be used to include observations dropped by `logistic` because their covariates determined success or failure perfectly.
6. The commands are no longer restricted to the estimation sample; statistics can be computed for any set of observations. This makes it possible to compute calibration and discrimination statistics for validation samples; see *sbe12* (Tilford et al. 1995) in this issue.
7. A vector of coefficients can be used to specify the model rather than the last model estimated by `logistic`. A goodness-of-fit test can be computed for a set of published coefficients applied to another data set.
8. A new command `lsens` produces a graph of sensitivity and specificity versus probability cutoff and optionally creates new variables containing this data.
9. The output of `lstat` has been improved.

Syntax

```
lfit  [depvar] [weight] [if exp] [in range] [, group(#) equal table
                                               all rules asif beta(matname) ]

lroc  [depvar] [weight] [if exp] [in range] [, nograph graph_options
                                               all rules asif beta(matname) ]

lstat [depvar] [weight] [if exp] [in range] [, cutoff(#)
                                               all rules asif beta(matname) ]

lsens [depvar] [weight] [if exp] [in range] [, nograph graph_options genprob(varname)
                                               gensens(varname) genspec(varname) replace
                                               all rules asif beta(matname) ]
```

Remarks

If you use these new commands the same way you used the old commands, you will get the same results except for `lfit, group(#)` which uses a slightly different grouping scheme; see the following discussion for details.

The estimation sample is always used by default. When weights, `if`, or `in` are used with `logistic`, it is not necessary to repeat them with these commands when you want statistics computed for the estimation sample (although there is no harm in doing so—the result is the same). Use `if`, `in`, or the `all` option only when you want statistics computed for a set of observations other than the estimation sample. Only specify weights (only `fweight`s are allowed) when you want to use a different set of weights.

depvar may only be specified when using the `beta()` option; see *Options* below.

Options

`all` requests that the statistic be computed for all observations in the data set ignoring any `if` or `in` restriction specified with `logistic`. Note: Any observations dropped by `logistic` because their covariates determined success or failure perfectly are excluded (as are any other observations in the data set with the same values as the ones dropped); to include these observations, use the `rules` or `asif` options.

`rules` requests that the predicted probabilities be set to 0 or 1 according to the "rules" determined during estimation for those observations dropped by `logistic` because their covariates determined success or failure perfectly. See [5s] logit in the *Stata Reference Manual* for a description of the `rules` option for `predict`.

`asif` requests that the predicted probabilities be computed using the estimated coefficients ignoring any exclusion criteria or "rules" determined during estimation for those observations dropped by `logistic` because their covariates determined success or failure perfectly. See [5s] logit in the *Stata Reference Manual* for a description of the `asif` option for `predict`.

`beta(`*matname*`)` specifies a row vector containing coefficients for a logistic model. The columns of the row vector must be labeled with the corresponding names of the independent variables in the data set. The dependent variable *depvar* is specified immediately after the command name.

`group(#)` specifies the number of quantiles to be used to group the data for the Hosmer–Lemeshow goodness-of-fit test. `group(10)` is typically specified. If this option is not given, the Pearson goodness-of-fit test is computed using the covariate patterns in the data as groups.

`equal` can only be specified when the `group()` option is used. This option determines how observations whose predicted probabilities are exactly equal to a quantile boundary are grouped. If `equal` is not specified, observations lying on a quantile boundary are grouped with the lower adjacent quantile. If `equal` is specified, observations lying on a quantile boundary are averaged and placed into the lower and upper adjacent quantiles in the proportions required to make the number of observations in each quantile exactly equal; see *Methods and Formulas* below.

`table` displays a table of the groups used for the Hosmer–Lemeshow or Pearson goodness-of-fit test with predicted probabilities, observed and expected counts for both outcomes, and totals for each group.

`nograph` suppresses graphical output.

graph_options are any of the options allowed with `graph, twoway`; see [3] twoway.

`cutoff(#)` specifies the value for determining whether an observation has a predicted positive outcome. Default is 0.5.

`genprob(`*varname*`)`, `gensens(`*varname*`)`, and `genspec(`*varname*`)` specify the names of new variables created to contain, respectively, the probability cutoffs and corresponding sensitivity and specificity.

`replace` requests that, if existing variables are specified for `genprob()`, `gensens()`, or `genspec()`, they should be overwritten.

The new lfit

If you run the `logistic` command and type `lfit` (or `lfit, group(#)`), you will get the Pearson (or Hosmer–Lemeshow) goodness-of-fit test performed on the estimation sample just as with the old `lfit` command. We do this below using data from Hosmer and Lemeshow (1989), omitting those with race classified as "other".

```
. logistic low lwd race1 smoke ptd ht if race2~=1
Logit Estimates                                     Number of obs =    122
                                                    chi2(5)       =  21.77
                                                    Prob > chi2   = 0.0006
Log Likelihood = -61.304392                         Pseudo R2     = 0.1508
------------------------------------------------------------------------------
         low | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
         lwd |   1.182983   .6924617      0.287   0.774       .3756005      3.7259
       race1 |   3.274937   1.748904      2.221   0.026       1.149843    9.327546
       smoke |   3.866022   1.976902      2.644   0.008       1.419057    10.53243
         ptd |    3.54029   2.001356      2.236   0.025       1.169088    10.72088
          ht |   2.976229   2.365802      1.372   0.170       .6266815    14.13467
------------------------------------------------------------------------------

. lfit, group(10) table
Logistic model for low, goodness-of-fit test
(Table collapsed on quantiles of estimated probabilities)
Note: Because of ties, there are only 7 distinct quantiles.
_Group     _Prob     _Obs_1     _Exp_1     _Obs_0     _Exp_0     _Total
     3    0.0845          3        3.3         36       35.7         39
     4    0.2322          4        3.0         10       11.0         14
     6    0.2630          6        6.8         20       19.2         26
```

```
 7    0.2969          3        3.8         10        9.2         13
 8    0.5151          3        3.0          3        3.0          6
 9    0.5582         10        8.7          6        7.3         16
10    0.8054          5        5.4          3        2.6          8

      number of observations =       122
            number of groups =         7
     Hosmer-Lemeshow chi2(5) =      1.35
                Prob > chi2 =    0.9299
```

The new `lfit` with the `group()` option produces a different grouping than the old `lfit`. There are 39 observations each having a predicted probability of 0.0845 in this data set. The new `lfit` keeps them together, so the first group is comprised of these 39 observations. The old `lfit`, on the other hand, aimed to make the first group contain exactly 12 observations, so it randomly selected 12 of the 39 to put in the first group. If you issued the command again, it would randomly select another 12 of the 39. Hence, you could get slightly different answers for the same logistic model each time you ran the old `lfit` with the `group()` option. Although this behavior is undesirable from the perspective of reproducibility, it is a valid statistical procedure.

The new `lfit` with the `group()` option acts deterministically; it puts all observations with the same predicted probabilities into the same group. If, as in this example, we request 10 groups, the groups that the new `lfit` makes are $[p_0, p_{10}]$, $(p_{10}, p_{20}]$, $(p_{20}, p_{30}]$, $\ldots$, $(p_{90}, p_{100}]$, where p_k is the k-th percentile of the predicted probabilities, with p_0 the minimum and p_{100} the maximum. For this case, we have $p_0 = p_{10} = p_{20} = p_{30}$ and $p_{50} = p_{60}$, so we only get seven groups.

This procedure, however, has its own drawbacks. If there are large numbers of ties at the quantile boundaries (as will frequently happen if all independent variables are categorical and there are only a few of them), the sizes of the groups will be uneven. If the totals in some of the groups are small, the χ^2 statistic is unreliable.

The `equal` option attempts to prevent this last problem by summing the observed and expected counts for all observations tied at a quantile boundary. It then apportions the summed observed counts and summed expected counts into the quantiles on either side of the boundary in such a way as to make the total number of observations in each group exactly equal. Here's an example:

```
. lfit, group(5) table equal

Logistic model for low, goodness-of-fit test
(Table collapsed on quantiles of estimated probabilities)

_Group     _Prob     _Obs_1     _Exp_1     _Obs_0     _Exp_0     _Total
     1    0.0845        1.9        2.1       22.5       22.3       24.4
     2    0.2322        4.0        3.2       20.4       21.2       24.4
     3    0.2630        5.7        6.3       18.7       18.1       24.4
     4    0.5151        7.2        8.2       17.1       16.2       24.4
     5    0.8054       15.2       14.3        9.2       10.1       24.4

      number of observations =       122
            number of groups =         5
     Hosmer-Lemeshow chi2(3) =      0.57
                Prob > chi2 =    0.9024
```

We requested only 5 groups since we did not want the group totals to be appreciably smaller than the number of observations tied at a particular predicted probability.

See the article by Tilford et al. in this issue (*sbe 12*) for examples of the use of `if` with `lfit` and `lroc` (statistics for out-of-sample data) and the use of the `beta()` option (specifying the logistic model with a vector of coefficients).

Graphing sensitivity and specificity versus probability cutoff

The new command `lsens` produces a plot of sensitivity and specificity versus probability cutoff. The graph is equivalent to what you would get from `lstat` if you varied the cutoff probability from 0 to 1.

```
. lsens
```

(graph appears, see Figure 1)

`lsens` will optionally create new variables containing the probability cutoff, sensitivity, and specificity:

```
. lsens, genprob(p) gensens(sens) genspec(spec) nograph
```

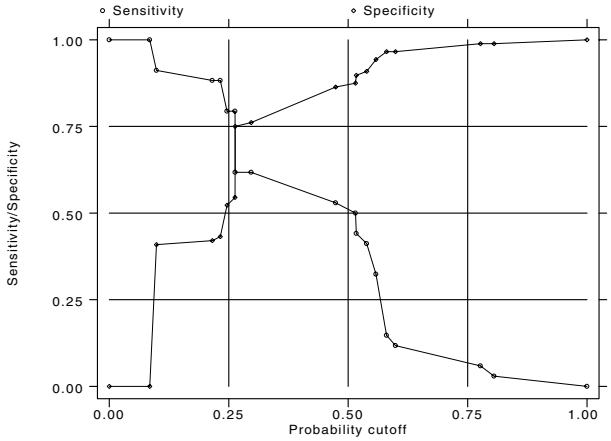

Figure 1. Example of `lsens` output

Methods and Formulas

The following formulas describe how `lfit` with the `group()` option divides the observations into groups. Let $p_1 < p_2 < \cdots < p_m$ be the unique values of the predicted probabilities, and let w_i be the number of observations having predicted probability p_i. Let o_i be the sum of observed successes (the dependent variable not equal to zero) for those observations having predicted probability p_i. The expected number of successes for those observations having predicted probability p_i is simply $e_i = w_i p_i$. Let $N = \sum w_i$ be the total number of observations. Let $G = \#$ be the number of groups requested with `group(#)`. Then the smallest index $q(j)$ such that

$$W_{q(j)} = \sum_{i=1}^{q(j)} w_i \geq \frac{Nj}{G}$$

gives $p_{q(j)}$ as the upper boundary of the j-th quantile, ($j = 1, 2, \ldots, G$). Let $q(0) = 1$ denote the first index.

If the `equal` option is not specified, the groups are

$$[\,p_{q(0)}, p_{q(1)}], (\,p_{q(1)}, p_{q(2)}], \ldots, (\,p_{q(G-1)}, p_{q(G)}]$$

If the `table` option is given, the upper boundaries $p_{q(1)}, \ldots, p_{q(G)}$ of the groups appear next to the group number on the output. The number of observations in the j-th group is

$$n_j = w_{q(j-1)+1} + w_{q(j-1)+2} + \cdots + w_{q(j)}$$

The number of observed successes in the j-th group is

$$O_j = o_{q(j-1)+1} + o_{q(j-1)+2} + \cdots + o_{q(j)}$$

A similar formula gives the expected successes.

If the `equal` option is specified, groups are constructed so that there are exactly $n = N/G$ observations in each group. Note that, in most cases, n will not be an integer. If an observation has predicted probability p_i with $p_{q(j-1)} < p_i < p_{q(j)}$, then it belongs to the j-th group. Those observations with $p_i = p_{q(j)}$ are apportioned into the j-th and $(j+1)$-th groups. With this apportionment scheme, the number of observed successes in the j-th group is

$$O_j = \frac{W_{q(j-1)} - n(j-1)}{w_{q(j-1)}}\, o_{q(j-1)} + o_{q(j-1)+1} + o_{q(j-1)+2} + \cdots + o_{q(j)-1} + \frac{nj - W_{q(j)-1}}{w_{q(j)}}\, o_{q(j)}$$

Substituting e_i for o_i in the formula gives the expected successes. If w_i is substituted into the formula, it is easy to see that one gets $n_j = n$ as we claim.

Note: If $W_{q(j)}$ is equal to Nj/G, then the quantile boundary that appears on the output is $[p_{q(j)} + p_{q(j+1)}]/2$, which is the standard formula for computing percentiles. In other words, the predicted probabilities that are displayed by `lfit` are exactly those from a standard percentile computation.

References

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

Tilford, J. M., P. K. Roberson, and D. H. Fiser. 1995. sbe12: Using lfit and lroc to evaluate mortality prediction models. *Stata Technical Bulletin* 28: 14–18.

crc42	Improvements to the heckman command

The `heckman` command has been improved. It now converges in fewer iterations and converges in some cases where the old `heckman` did not. The new `heckman` also deals with missing values in a more sophisticated fashion, and it displays more output when the `trace` option is specified.

The improvement in the convergence of `heckman` is due to a change in the parameterization of the correlation ρ. The old parameterization was $\theta = \tan(\pi\rho/2)$. The new parameterization uses an inverse hyperbolic tangent transformation:

$$\theta = \operatorname{atanh} \rho = \frac{1}{2} \log\left(\frac{1+\rho}{1-\rho}\right)$$

The atanh transformation and its inverse, tanh, (used each iteration to recover ρ) are numerically smoother than tan and atan. As a result, the new version of `heckman` converges in fewer iterations and converges for large values of $|\rho|$ when the old `heckman` would not. In tests we have run where the old `heckman` converged, the new `heckman` produces the same answers as the old `heckman` to a tolerance of $< 10^{-5}$.

The old `heckman` would omit observations that contained any missing values for any exogenous variable from both the probit and regression estimations. The new `heckman` is smarter. It doesn't care if exogenous variables that are only part of the regression equation (and not part of the probit equation) have missing values when the dependent variable of the regression equation is also missing. Observations are retained as part of the sample for probit estimation as long as their values for all exogenous variables of the probit equation are nonmissing.

When the `trace` option is specified, the new `heckman` first displays the initial probit and linear regression that give the initial Mills' ratio coefficient estimate and then gives the details of the maximum likelihood iterations.

Example

We redo the deliberately nonsensical example given at the end of [5s] heckman in the *Stata Reference Manual.*

```
. use auto
(1978 Automobile Data)
. replace price = . if foreign
(22 real changes made, 22 to missing)
. eq price mpg weight
. eq probit: displ
. heckman price probit, trace
Probit Estimates                                  Number of obs =     74
                                                  chi2(1)       =  44.06
                                                  Prob > chi2   = 0.0000
Log Likelihood = -23.004215                       Pseudo R2     = 0.4892
------------------------------------------------------------------------------
__000470 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   displ |   .0231829   .0059231      3.914   0.000        .011574    .0347919
   _cons |  -2.999473    .787286     -3.810   0.000      -4.542525   -1.456421
------------------------------------------------------------------------------
Note: 0 failures and 3 successes completely determined.
Mills' ratio coefficient estimate from regression
  Source |       SS       df       MS                 Number of obs =      52
---------+------------------------------              F(  3,    48) =   18.17
   Model |   260157662     3  86719220.6              Prob > F      =  0.0000
Residual |   229037139    48  4771607.06              R-squared     =  0.5318
---------+------------------------------              Adj R-squared =  0.5025
   Total |   489194801    51  9592054.92              Root MSE      =  2184.4
```

```
------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
__00047Q |   2557.997   1143.039      2.238   0.030       259.7625    4856.231
     mpg |   162.7514   137.8043      1.181   0.243      -114.3226    439.8254
  weight |   5.120607   .9647999      5.307   0.000       3.180747    7.060467
   _cons |  -14778.32   5545.562     -2.665   0.010      -25928.42   -3628.217
------------------------------------------------------------------------------

Coefficient of Mills' ratio  2557.997
Initial estimate of rho      1.171029
Initial rho set to           0.99
Heckman model may be inappropriate since initial |rho| > 0.9

ML computation begins ...
   162.75137    5.120607  -14778.316   .02318294  -2.9994732   2.6466524
   7.6890969
Iteration 0:  Log Likelihood = -494.74219
(nonconcave function encountered)

(output omitted)

Iteration 12:  Log Likelihood = -490.83493
    105.5897   4.1791758   -10449.21    .0262556  -3.3591755   2.2028686
   7.7478327
Iteration 13:  Log Likelihood = -490.83493

Heckman selection model                           Number of obs    =      74
                                                  Model chi2(4)    =   58.71
                                                  Prob > chi2      =  0.0000
Log Likelihood =   -490.8349341

------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
price    |
     mpg |   105.5897   77.37173      1.365   0.172       -46.0561    257.2355
  weight |   4.179176   .6291039      6.643   0.000       2.946155    5.412197
   _cons |  -10449.21    3376.82     -3.094   0.002      -17067.66   -3830.765
---------+--------------------------------------------------------------------
probit   |
   displ |   .0262556   .0067118      3.912   0.000       .0131007    .0394105
   _cons |  -3.359176    .855216     -3.928   0.000      -5.035368   -1.682983
---------+--------------------------------------------------------------------
_athrho  |
   _cons |   2.202869    .892738      2.468   0.014       .4531342    3.952603
---------+--------------------------------------------------------------------
_lnsigma |
   _cons |   7.747833   .1010999     76.635   0.000       7.549681    7.945985
------------------------------------------------------------------------------
     rho      0.97588               [_athrho]_cons  = atanh(rho)
   sigma    2316.5463               [_lnsigma]_cons = ln(sigma)
  lambda    2260.6717   269.7547
```

Note the large value of `rho`. The old `heckman` would not converge when given this example. Although this behavior may be desirable for stupid models such as this, we replace it with a `heckman` that is numerically sounder.

dm35 | A utility for surveying Stata-format data sets

Timothy J. Schmidt, Federal Reserve Bank of Kansas City, FAX 816-881-2199

`dtainfo` is a utility program that reads all Stata-format data sets in the current directory and reports information about their contents. `dtainfo` can be used to report the location, type, and other content information for a specific variable and content information for each Stata data set and its variables.

Syntax

To obtain information about a specific variable, the syntax is

`dtainfo` *varname*

where *varname* is the name of the variable you seek. To obtain summary information about all Stata data sets in the current directory, the syntax is

```
dtainfo [, sort(fname|obs|var|double|float|long|integer|byte|miss) ]
```

Options

`sort(fname|obs|var|double|float|long|integer|byte|miss)` specifies the sort order of the report as follows:

argument	sorts report by
`fname`	filename
`obs`	number of observations
`var`	number of variables
`double`	number of type `double` variables
`float`	number of type `float` variables
`long`	number of type `long` variables
`integer`	number of type `integer` variables
`byte`	number of type `byte` variables
`miss`	number of missing observations

Discussion

In conducting a research project, many researchers construct and accumulate numerous Stata data sets in a disk directory devoted to the project. As a result, two problems commonly arise. First, it can become difficult to distinguish between data sets and to identify how they are similar or different. Second, it can be difficult to remember where a particular variable is stored. `dtainfo` is designed to help alleviate these problems.

For quick comparisons of a large number of data sets, `dtainfo` provides an overview of the contents and structure of all Stata-format data sets (those ending in a `.dta` file extension) in the current disk directory. `dtainfo` offers several advantages over current alternatives in Stata. First, `dtainfo` provides more information than `describe` ([5d] describe) by displaying the date the file was last saved, the number of missing observations in the data set, and a summary of the number of variables by type. Second, `dtainfo` is more convenient to use in examining a large number of data sets. In one listing, `dtainfo` provides a "snapshot" of every Stata-format data set in the current directory. With `describe` or `cf` ([5d] cf), the user can examine or compare only one file at a time, a tedious task when the number of files is large. In addition, the user may specify that `dtainfo` report the summary information sorted on any one of the reported fields. With the sorting option, the user can quickly identify similar or dissimilar data sets by their most important features.

A second problem in working with many data sets is remembering where a particular variable is stored. To locate a variable, the user would probably resort to describing each data set likely to contain it. If the number of data sets is large, this process could be tedious and time consuming. With `dtainfo`, however, the user can rapidly locate all occurrences of a particular variable name in the current directory's Stata-format data sets. For each occurrence of a variable name, `dtainfo` reports the data set in which it is located, its variable type, the number of observations, and the number of missing observations. This information allows the user to quickly locate where a variable is stored and to identify differences in any multiple occurrences of a variable. Thus, the information from `dtainfo` can complement Stata's `codebook` ([5d] codebook), `compare` ([5d] compare), and `inspect` ([5d] inspect) commands.

Example

The following example illustrates the use of `dtainfo`. Suppose the user is conducting research on foreign exchange rates. In a separate disk directory, the user has created a number of Stata-format data sets to store a large amount of data. After being away from the project for a time, the user may find it useful to have a quick survey of all the Stata-format data sets.

```
. dtainfo
Data set        Date     Time    Obs  Vars  Doub  Floa  Long   Int  Byte   Miss
JAPAN.DTA     10-29-92  11:14    118    27     0    27     0     0     0    557
FXRATE.DTA     9-07-95  13:30   3660     8     0     5     0     0     3    705
FORDIFF.DTA    9-07-95  13:31    171    70     0    68     0     1     1    785
```

```
EDAILY.DTA     9-07-95  13:33   3687     4     0     1     0     1     2    140
EASTGERM.DTA   9-07-95  13:39     32     5     0     2     0     1     1      5
REALFXMO.DTA   9-07-95  13:55    284    18     0    16     0     1     1    114
UKMON.DTA     12-10-92  11:21     79     3     0     3     0     0     0      0
FXRATE1.DTA    1-08-93  13:47   1827     6     0     6     0     0     0    219
BFXIGEJA.DTA   6-23-94  12:36    124    19     0    19     0     0     0    138
FXRATE2.DTA    1-08-93  13:49   1833     6     0     6     0     0     0    204
USACANCU.DTA  11-23-93  15:24     23     7     0     7     0     0     0      0
FORINDIC.DTA   3-12-93  13:39    183    32     0    32     0     0     0    193
FORCPIFX.DTA   3-15-93  15:02    170    18     0    18     0     0     0     13
```

For each data set in the current directory, `dtainfo` provides the user with a lot of information neatly summarized on one line. For instance, `EASTGERM.DTA` was last saved on September 7, 1995, at 1:39 p.m. It contains five variables: two of type `float`, one of type `int`, and one of type `byte`. Note that since Stata has six basic datatypes and `dtainfo` categorizes variables by the five numeric datatypes, the other variable in `EASTGERM.DTA` must be a string. `dtainfo` also reports that `EASTGERM.DTA` has 32 observations on each variable and a total of five missing values in the data set.

`dtainfo` can also provide the same information sorted on any one of its fields (except date and time). For instance, if the user suspects that data sets with similar names contain similar data, it may be useful to sort by filename.

```
. dtainfo, sort(fname)
Data set        Date     Time    Obs  Vars  Doub  Floa  Long   Int  Byte   Miss
BFXIGEJA.DTA   6-23-94  12:36    124    19     0    19     0     0     0    138
EASTGERM.DTA   9-07-95  13:39     32     5     0     2     0     1     1      5
EDAILY.DTA     9-07-95  13:33   3687     4     0     1     0     1     2    140
FORCPIFX.DTA   3-15-93  15:02    170    18     0    18     0     0     0     13
FORDIFF.DTA    9-07-95  13:31    171    70     0    68     0     1     1    785
FORINDIC.DTA   3-12-93  13:39    183    32     0    32     0     0     0    193
FXRATE.DTA     9-07-95  13:30   3660     8     0     5     0     0     3    705
FXRATE1.DTA    1-08-93  13:47   1827     6     0     6     0     0     0    219
FXRATE2.DTA    1-08-93  13:49   1833     6     0     6     0     0     0    204
JAPAN.DTA     10-29-92  11:14    118    27     0    27     0     0     0    557
REALFXMO.DTA   9-07-95  13:55    284    18     0    16     0     1     1    114
UKMON.DTA     12-10-92  11:21     79     3     0     3     0     0     0      0
USACANCU.DTA  11-23-93  15:24     23     7     0     7     0     0     0      0
```

`dtainfo` is also useful in locating specific variables. For example, the user may remember the U.S. dollar/Canadian dollar exchange rate variable is called `ecan`, but locating that variable among all of the data sets could be tedious. This is a job for `dtainfo`.

```
. dtainfo ecan
EDAILY.DTA   contains ecan :  float     3687 observations      140 missing
```

`dtainfo` reports `ecan` is stored in the data set `EDAILY.DTA` as a type float variable with 3687 observations, 140 of which are missing.

With a little creativity, `dtainfo` can assist the user in many other ways. For instance, a user who wanted to know which data sets likely contain daily data could search for all occurrences of the variable `day`.

```
. dtainfo day
FXRATE.DTA   contains day :  byte      3660 observations        0 missing
EDAILY.DTA   contains day :  byte      3687 observations        0 missing
FXRATE1.DTA  contains day :  float     1827 observations        0 missing
BFXIGEJA.DTA contains day :  float      124 observations        0 missing
FXRATE2.DTA  contains day :  float     1833 observations        0 missing
```

A final note

The heart of `dtainfo` is an executable file compiled from source code I wrote in C. Unfortunately, I do not have access to a C compiler for the Apple Macintosh. Therefore, I am currently unable to provide a version of `dtainfo` for users of Stata on the Apple Macintosh. However, if there is enough interest in `dtainfo` by users of the Mac version of Stata, I will be happy to seek out a Mac and release a Mac-compatible version of `dtainfo` in a future issue of the STB. Alternatively, ambitious Mac users with a C compiler can try to port the source code for `dtainfo`, which is included on the STB diskette.

dm36	Comparing two Stata data sets

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

Stata provides the command `cf` ([5d] cf) for comparing a list of variables in memory with equally named variables in a Stata-format data set stored on disk. However, the information provided by the `cf` command is often inadequate for the task at hand. Consider the following situations:

1. You have collected a large and important data set. It is essential that the data be entered into Stata error-free, so you mimic the practice of double-entry bookkeeping—two typists independently enter the data into text files or Stata's data editor. Afterward, you face the task of resolving the discrepancies (if any) between two supposedly identical Stata data sets.
2. You have data that were collected in three cities (say, Denver, Houston, and Omaha). You find it convenient to keep the data in separate files (`denver.dta`, `houston.dta`, `omaha.dta`), as well as in a single, combined file (`cities.dta`). After a time, you recall that some errors have been corrected in the Omaha data, but you're uncertain about just where the corrections were made. You'd like to compare `omaha.dta` with the Omaha observations in `cities.dta` to identify any discrepant data values.
3. You have two versions of what is nominally the same data set. You know or suspect that there are many places where corresponding data values in the two versions are not *exactly* the same—perhaps because the two versions are based on slightly different computational algorithms, or because they were obtained from not-quite-interchangeable sources. You'd like to meander through the two data sets, examining corresponding observations to discern any patterns in the discrepancies that exist.

This insert presents `compdta`, a command that offers several advantages over `cf` in such situations. The syntax of `compdta` is

`compdta` [*varlist*] [`if` *exp*] [`in` *range*] `using` *filename* [, `keep(max|min)` `list` `noisily` `sort`]

`compdta` compares the *varlist* from the data set in memory (the *master* data set) with like-named variables in the Stata-format data set specified by *filename* (the *using* data set). If *varlist* is absent, the comparison is made for all variables in the master data set. An `if` or `in` clause restricts the comparison to observations that satisfy those conditions; otherwise, all observations in the master and using data sets are compared. Neither the master nor the using data set is altered by `compdta`. (The options are explained in the text below.)

When the master and using data sets are in perfect agreement, `compdta` and `cf` give the same response (that is, silence), and `compdta`'s `noisily` option produces output similar to that of `cf`'s `verbose` option: a list of the variables compared, with an indication that no differences were found. Under other circumstances, the responses of `compdta` and `cf` differ in several useful ways.

Example 1

Consider situation (1) above. The principal task is to resolve any mismatches by editing one or both of the values entered by the typists. `cf` can only count the number of mismatches for each variable. `compdta` can, in addition, list the mismatched pairs of data values and their observation number in side-by-side fashion; the `list` option produces such a display for each *varlist* variable whenever disagreements are found.

To simulate situation (1), make a few changes in the 1980 Census housing data `hsng.dta` that is distributed with Stata. Then, save the altered version as `temp.dta`:

```
. use hsng
(1980 Census housing data)
. replace state = upper(state) in 2/4
(3 real changes made)
. replace pop = pop + 2.5 in -2/-1
pop was long now double
(2 real changes made)
. save temp
file temp.dta saved
```

Now, use `compdta` to compare some variables in the two data sets. The default response of `compdta` resembles that of `cf`:

```
. use hsng
(1980 Census housing data)
```

```
. compdta state-pop using temp
3 mismatches in state
2 mismatches in pop
```

But the `list` option allows `compdta` to display the information necessary to resolve discrepancies:

```
. compdta state-pop using temp, list
3 mismatches in state
             state         state_
  2.        Alaska         ALASKA
  3.       Arizona        ARIZONA
  4.      Arkansas       ARKANSAS
2 mismatches in pop
            pop        pop_
 49.    4705767   4705769.5
 50.     469557    469559.5
```

In this display, the variable `state` is from the master data, the variable `state_` is the variable named `state` in the using data, etc. See Example 3 for details of `compdta`'s naming rules.

Example 2

Now consider situation (2) above. The problem is to compare two data sets, one of which purports to be a (proper) subset of the other. Because `cf` does not allow `if` or `in` clauses, it cannot make such a comparison; `compdta` can. To simulate this situation, extract from `hsng.dta` the 13 observations that are from region 4. Alter a single data value and save the data in `region4.dta`. Then, use `compdta` to verify that the region 4 subset of `hsng.dta` is identical to `region4.dta`, except for the single alteration:

```
. use hsng
(1980 Census housing data)
. keep if region==4
(37 observations deleted)
. replace state = "" in 10
(1 real change made)
. save region4
file region4.dta saved
. compdta if region==4 using hsng, l
1 mismatches in state
          state      state_
 10.                 Oregon
```

Example 3

In situation (3) above, the two variants of the `keep` option can be quite helpful. `compdta` uses Stata's `merge` command to merge observations from the master and using data. If *varlist* specifies p variables and the `if` and `in` clauses specify n observations, then `compdta` forms a data set with $2p$ variables and n observations. By default, `compdta` replaces this merged data set with the master data at exit; `keep(max)` and `keep(min)` each leave a part of the merged data in memory. In addition, both variants of `keep` create a variable named `_compdta` that records the number of mismatched data values for each observation.

`keep(max)` retains the largest relevant portion of the merged data: all n observations on the *varlist* variables from the master data, plus their counterparts from the using data when discrepancies are found. If mismatches are found in p' variables, $p + p'$ variables are retained, ordered so that paired variables are adjacent to each other.

To illustrate, reconsider Example 1, but this time add the `keep(max)` option, and then use `describe`:

```
. use hsng
(1980 Census housing data)
. compdta state-pop using temp, keep(max)
3 mismatches in state
2 mismatches in pop
```

```
. describe
Contains data
  Obs:    50 (max= 30465)                    1980 Census housing data
 Vars:     7 (max=    99)
Width:    41 (max=   200)
  1. state        str13  %13s                State
  2. state_       str13  %13s                state in temp.dta
  3. division     byte   %8.0g       division Census division
  4. region       byte   %8.0g       region   Census region
  5. pop          long   %10.0g              Population in 1980
  6. pop_         double %10.0g              pop in temp.dta
  7. _compdta     byte   %8.0g               No. of mismatches
Sorted by:
Note:  Data has changed since last save
```

The variables `state_` and `pop_` are the variables named `state` and `pop` in the using data—as their labels indicate. `compdta` must assign new names to variables in the using data when they are merged with the master data. Its primary strategy is to append an underscore to variable names in the using data (as in the example above). If a variable name is 8 characters long, `compdta` begins at the end of the name and attempts to substitute an underscore or to toggle the case of the eighth character. If this fails to produce an unused name, it moves one character to the left and repeats the process as necessary. (This strategy is not guaranteed to succeed, but failures to produce an acceptable renaming should be extremely rare.)

The data set retained by `keep(max)` makes it easy to look for patterns in the mismatched data values, especially if the spreadsheet data editor launched by the `browse` and `edit` commands is available. Moreover, it may be possible to quickly produce a "correct" data set by (repeatedly) editing one version of a variable and then dropping its counterpart.

The `keep(min)` option might be preferred if you plan instead to reload the master or using data and make changes there. `keep(min)` retains the smallest relevant portion of the merged data: Only those variables and those observations where mismatched data values were found. In addition, `keep(min)` creates a variable named `_id` that records the original observation number, and sets to missing (or blank, for string variables) all pairs of retained data values that agree. This makes discrepant data values easy to locate. To illustrate, repeat the last example, this time using `keep(min)`, followed by `list` to display the entire contents of the retained data set:

```
. use hsng, replace
(1980 Census housing data)
. compdta state - pop using temp, k(min)
3 mismatches in state
2 mismatches in pop
. list
        state     state_         pop        pop_  _compdta         _id
  1.    Alaska     ALASKA           .           .         1           2
  2.   Arizona    ARIZONA           .           .         1           3
  3.  Arkansas   ARKANSAS           .           .         1           4
  4.                          4705767   4705769.5         1          49
  5.                           469557    469559.5         1          50
```

Example 4

By default, `compdta` makes the same assumption as `cf` about the ordering of observations—namely, that the master and using data sets are to be compared using their current sort order. Thus, two identical data sets can yield many mismatches if their sort orders differ. `compdta`'s `sort` option attempts to coerce the using data into the same sort order as the master data before making comparisons.

To illustrate, copy `hsng.dta` to `temp.dta` changing only the sort order. Then, by default, `compdta` (like `cf`) finds many mismatches in the two data sets:

```
. use hsng, replace
(1980 Census housing data)
. sort popden
. save temp, replace
file temp.dta saved
. use hsng
(1980 Census housing data)
```

```
. compdta using temp
49 mismatches in state
45 mismatches in division
37 mismatches in region
     (etc.)
```

But,

```
. compdta using temp, sort
```

finds no mismatches in the two data sets.

Note that `compdta`'s strategy for overcoming differences in sort order can fail if there are ties among the values of the variables defining the sort order. (In the example above, sorting `hsng.dta` on `region` creates this situation.) This problem can be avoided by sorting the master data set on a variable that creates a unique ordering of the observations (e.g., `state` in `hsng.dta`).

Additional remarks

As noted earlier, the `noisily` option forces a verbose display of the comparison of *varlist* in the master and using data. `noisily` also causes `compdta` to display notes about its progress as it prepares the merged data set described in Example 3. For example, `compdta` compresses both the using and master data before merging. Ordinarily, this is done silently; `noisily` displays the effects of the `compress` command. Further, `noisily` issues a warning when the master and using data sets differ in sort order, and shows the minimum values of the `maxvar` and `width` parameters required by the merge operation.

Finally, `compdta` is usually faster than `cf`. This is because `cf` merges the using data one variable at a time, whereas `compdta` performs a single merge of all the *varlist* variables from the using data. The disadvantage, of course, is that `compdta` needs room for about twice as many variables as does `cf`. If the `maxvar` and `width` parameters cannot be set large enough, it will be necessary to subdivide *varlist* and perform the comparison in stages.

ip10 | Finding an observation number

Sean Becketti, Stata Technical Bulletin, stb@stata.com

Stata provides the `if` and `in` clauses to restrict the operation of a command to a subset of the data. These clauses make it easy to locate interactively the observations that satisfy certain conditions. For example, if you `use` the familiar automobile data and type

```
. list if foreign
```

Stata will display all the observations that contain information on foreign cars.

What if you want to find just the first observation on a foreign car? As it happens, these data contain all the domestic cars first, followed by all the foreign cars. Thus, one way to solve this problem is

```
. list if foreign & !foreign[_n-1]
```

This command lists observation 53, the first foreign car in the data set. In fact, by poking around with the `list` command, it is straightforward to locate any observation interactively, even when you don't have any prior knowledge of the structure of the data set.

This problem is more complicated inside a Stata program. It is relatively easy to use `if` and `in` to restrict an operation to a particular observation. Sometimes, though, you really need to store the observation number of a target observation, perhaps to set the boundaries of a `while` loop.

`findobs` solves this problem. `findobs` displays the observation number of the i-th observation that satisfies specified conditions. The observation number is also stored in the `S_1` macro, so it can be used later in the program. The syntax of `findobs` is

`findobs` [`if` *exp*] [`in` *range*] [, `instance(#)`]

The `instance()` option specifies which occurrence of the target condition is to be located. For example, `instance(3)` requests the observation number of the third occurrence of the specified conditions. The default is `instance(1)`. Typing `instance(0)` returns the number of the observation *before* the first occurrence. Negative numbers indicate that occurrences are to be counted starting from the end of the data set. Thus, `instance(-1)` requests the observation number of the *last* occurrence, `instance(-2)` requests the next-to-last occurrence, and so on. `instance(-0)` requests the number of the observation *following*

the last occurrence. If the specified occurrence is not found, if the indicated observation number is less than one, or if it is greater than _N, findobs displays nothing and stores a missing value in S_1.

Examples

The following session log illustrates the operation of findobs

```
. drop _all
. set obs 10
obs was 0, now 10
. generate i = _n
. findobs if i==3
3
. display "$S_1"
3
. findobs if i==2*int(i/2)                  /* Find the first even number */
2
. findobs if i==2*int(i/2), instance(3)     /* Find the third even number */
6
. findobs if i==2*int(i/2), instance(0)
1
. findobs if i==2*int(i/2), instance(-2)    /* Next-to-last even number */
8
. findobs if i==3, instance(-0)
4
. findobs if i==2*int(i/2), instance(-0)
```

If you have installed the Stata time series library in your ado-path (Becketti 1995), findobs will detect it and display the date associated with the target observation as well as the observation number. To illustrate this feature, we continue the example.

```
. generate int month = i
. generate year = 1995
. labmonth
. period 12
12 (monthly)
. datevars year month
. findobs if i==3
March, 1995 (3)
. display "$S_1"
3
. display "$S_2"
March, 1995
```

Reference

Becketti, S. 1995. sts7.6: A library of time series programs for Stata (Update). *Stata Technical Bulletin* 24: 30–35.

sbe12	Using lfit and lroc to evaluate mortality prediction models

John M. Tilford, Paula K. Roberson, and Debra H. Fiser
University of Arkansas for Medical Sciences
EMAIL mtilfor@care.ach.uams.edu

Recent developments in illness severity scoring systems permit medical researchers to estimate a patient's probability of death. Information on probability of death can be useful for a number of applications including patient prognosis, quality assessment, and clinical research.

Accuracy of the severity scoring system is crucial for these types of analyses. Lemeshow and LeGall (1994) review several systems that estimate the probability of death for adult intensive care patients and document methods for evaluating the performance of these systems. Evaluation of severity scoring systems is accomplished through assessment of calibration and discrimination statistics in developmental and validation samples (Hadorn et al. 1993).

Calibration and discrimination

Calibration measures the correspondence between estimated probabilities of death predicted by the severity scoring system and actual mortality. Formal testing of calibration uses Hosmer–Lemeshow goodness-of-fit tests (`lfit` following a `logistic` regression model). Discrimination measures the ability of the model to distinguish survivors from nonsurvivors and is measured by the area under the receiver operating characteristic (ROC) curve (`lroc` following `logistic`).

Assessment of model performance based solely on calibration and discrimination statistics from developmental samples can be misleading. Developmental sample statistics tend to overstate performance because the model was calculated to provide the best fit to these specific data. Clearly, to be useful, a model must perform well on the data from which it was developed. Evaluation on an independent (validation) sample, however, provides a more comprehensive picture of how well the model will perform in predicting future observations.

Previous versions of `lfit` and `lroc` permitted the calculation of calibration and discrimination statistics only on developmental samples. With the new versions of `lfit` and `lroc` (see *crc41* in this issue), these statistics are now available to Stata users on both developmental and validation samples. Furthermore, the `lfit` procedure now includes probabilities of mortality (or other outcomes of interest) in the output and both procedures permit researchers to substitute their own vector of coefficients in place of the estimated coefficients from the `logistic` command.

Example

We present a simple example to calculate the Hosmer–Lemeshow goodness-of-fit statistic and the ROC curve in developmental and validation data sets using estimated coefficients from a logistic regression and using coefficients from the published literature. The example uses data from several large pediatric intensive care units (PICUs). Patient probabilities of mortality are predicted using the Pediatric Risk of Mortality (PRISM) scoring system (Pollack et al. 1989). This scoring system assigns values to physiologic variables routinely collected upon PICU admission and forms a PRISM score by summing the values. The PRISM score and other variables are multiplied by coefficients from a published logistic regression equation to form the logit value $\mathbf{X}\beta$. A probability of mortality is calculated by transforming the logit value by the equation

$$P(\text{PICU mortality}) = \frac{e^{\mathbf{X}\beta}}{1 + e^{\mathbf{X}\beta}}$$

In this example, we estimate our own regression coefficients to illustrate Stata's new capabilities and then compare calibration and discrimination statistics based on our estimated coefficients and the published coefficients. The example uses preliminary data and is for illustrative purposes only.

Part 1: Calibration and discrimination in the developmental sample

The first step involves randomly splitting 11,365 observations in half. The first half of the observations will be the developmental sample and the second half will be the validation sample.

```
. use picu
. gen u = uniform()
. gen ndata = (u<=.5)
```

Second, we use the developmental sample to estimate a logistic regression on PICU mortality using variables included in the original paper: operative status, age (in months), and the PRISM scores.

```
. logistic mort opstat icuprism agem if ndata==1
Logit Estimates                                   Number of obs =   5660
                                                  chi2(3)       = 829.13
                                                  Prob > chi2   = 0.0000
Log Likelihood = -655.41765                       Pseudo R2     = 0.3874
------------------------------------------------------------------------------
    mort | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  opstat |   .6122442   .1161535     -2.586   0.010       .4221208    .8879993
icuprism |   1.216156   .0108302     21.975   0.000       1.195113    1.237569
    agem |    1.00173   .0011885      1.457   0.145       .9994037    1.004063
------------------------------------------------------------------------------

. logit
Logit Estimates                                   Number of obs =   5660
                                                  chi2(3)       = 829.13
                                                  Prob > chi2   = 0.0000
Log Likelihood = -655.41765                       Pseudo R2     = 0.3874
------------------------------------------------------------------------------
    mort |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  opstat |   -.490624   .1897176     -2.586   0.010      -.8624636   -.1187843
icuprism |   .1956949   .0089052     21.975   0.000       .1782409    .2131488
    agem |   .0017289   .0011864      1.457   0.145      -.0005965    .0040543
   _cons |  -5.114916   .1767704    -28.935   0.000       -5.46138   -4.768453
------------------------------------------------------------------------------
```

The results of this regression are displayed showing the underlying coefficients so that they can be compared to the published coefficients. In general, the coefficients from this regression are similar to the published coefficients:

$$\text{logit} = -.433\,\texttt{opstat} + .207\,\texttt{icuprism} - .005\,\texttt{agem} - 4.782$$

To test calibration in the developmental sample, the Hosmer–Lemeshow goodness-of-fit test is calculated by `lfit`. Note that, by default, `lfit` uses the same data that the `logistic` command used. We also requested the test to be calculated on ten "deciles of risk" and displayed in a table.

```
. lfit, group(10) table
Logistic model for mort, goodness-of-fit test
(Table collapsed on quantiles of estimated probabilities)
_Group     _Prob     _Obs_1     _Exp_1     _Obs_0     _Exp_0     _Total
     1    0.0046          3        2.3        563      563.7        566
     2    0.0062          4        3.2        562      562.8        566
     3    0.0075          2        3.8        566      564.2        568
     4    0.0090          3        4.6        561      559.4        564
     5    0.0131          4        6.3        562      559.7        566
     6    0.0160          3        8.1        563      557.9        566
     7    0.0217          9       10.4        557      555.6        566
     8    0.0350         21       15.7        545      550.3        566
     9    0.0861         37       30.8        529      535.2        566
    10    0.9987        179      179.8        387      386.2        566
      number of observations =      5660
            number of groups =        10
     Hosmer-Lemeshow chi2(8) =      9.38
                Prob > chi2 =      0.3112
```

The output gives the ten groups, their cutoff probability, the number of observed deaths, the number of expected deaths, the number of observed survivors, the number of expected survivors, and the total observations in each group. In this test, a lower value of the χ^2 statistic or a higher p-value indicates a better fitting model. The p-value of 0.31 indicates good calibration in the developmental sample as does the close association between observed and expected mortality in the 10 probability groups.

The area under the ROC curve provides an estimate of the model's discrimination:

```
. lroc, nograph
Logistic model for mort
number of observations =     5660
area under ROC curve   =   0.8992
```

System developers typically require discrimination statistics around 0.70. Examination of the output from the `lroc` procedure indicates the model has good discrimination in the developmental sample.

Part 2: Calibration and discrimination in the validation sample

We now examine whether the model performs as well in the validation sample based on calibration and discrimination statistics. To test calibration in the validation sample, we used the `lfit` procedure and specifically requested the test to be run on the second half of our data that was not included in the logistic regression.

```
. lfit if ndata==0, group(10) table

Logistic model for mort, goodness-of-fit test
(Table collapsed on quantiles of estimated probabilities)

_Group     _Prob     _Obs_1     _Exp_1     _Obs_0     _Exp_0     _Total
     1    0.0047          1        2.3        570      568.7        571
     2    0.0063          4        3.3        566      566.7        570
     3    0.0077          5        3.9        566      567.1        571
     4    0.0094          4        4.8        566      565.2        570
     5    0.0131          5        6.5        566      564.5        571
     6    0.0159          3        8.2        567      561.8        570
     7    0.0215         14       10.4        557      560.6        571
     8    0.0359         16       15.8        554      554.2        570
     9    0.0784         39       31.0        532      540.0        571
    10    0.9987        174      184.1        396      385.9        570

       number of observations =      5705
             number of groups =        10
      Hosmer-Lemeshow chi2(8) =         9.34
                  Prob > chi2 =         0.3143
```

The p-value for the Hosmer–Lemeshow test in the validation sample was almost exactly the same as the p-value for the developmental sample.

```
. lroc if ndata==0, nograph

Logistic model for mort

number of observations =     5705
area under ROC curve   =   0.8885
```

The discrimination test statistic (area under ROC curve = 0.8885) in the validation sample was only slightly lower than the test statistic from the developmental sample (area under ROC curve = 0.8992). These statistics indicate the PRISM scores in our data have excellent discrimination and calibration in both developmental and validation data sets.

Part 3: Calibration and discrimination using published coefficients

Finally, we compute calibration and discrimination statistics using the coefficients published by the developers of the PRISM scoring system (Pollack et al. 1989). For comparison, we calculate these statistics on our developmental data set, but they could be calculated on either the validation or developmental data set or on the entire data set.

First, we assign the published PRISM coefficients to a row vector named `b`. Then we label the columns of `b` with the corresponding names of our independent variables.

```
. mat b = (-.433,.207,-.005,-4.782)
. mat colnames b =  opstat icuprism agem _cons
```

We ask for calibration and discriminations statistics using the dependent variable `mort` (independent variables are automatically determined from the column names of the row vector `b`).

```
. lfit mort if ndata==1, beta(b) group(10) table

Logistic model for mort, goodness-of-fit test
(Table collapsed on quantiles of estimated probabilities)

_Group     _Prob     _Obs_1     _Exp_1     _Obs_0     _Exp_0     _Total
     1    0.0039          3        1.7        563      564.3        566
     2    0.0053          2        2.7        571      570.3        573
     3    0.0071          0        3.4        559      555.6        559
     4    0.0089          3        4.5        563      561.5        566
     5    0.0124          6        6.1        560      559.9        566
     6    0.0178          4        8.7        562      557.3        566
     7    0.0250          9       11.5        557      554.5        566
     8    0.0415         19       18.1        547      547.9        566
     9    0.0966         45       35.7        521      530.3        566
    10    0.9995        174      198.9        392      367.1        566

       number of observations =      5660
             number of groups =        10
      Hosmer-Lemeshow chi2(8) =        15.68
                  Prob > chi2 =         0.0471
```

```
. lroc mort if ndata==1, beta(b) nograph
Logistic model for mort
number of observations =     5660
area under ROC curve   =   0.9017
```

The Hosmer–Lemeshow χ^2 statistic, not surprisingly, indicates a poorer fit using the published coefficients than coefficients estimated from our own data. The area under the ROC curve, however, is similar to our previous estimates.

This information is useful for quality of care analyses based on this type of data. Quality of care analyses often use adjusted mortality rates where the adjustments derive from severity models like the PRISM. Severity models that perform poorly in terms of discrimination or calibration in either developmental or validation samples could cause such analyses to be misleading.

References

Hadorn D. C., E. B. Keeler, W. H. Rogers, and R. H. Brook. 1993. Assessing the performance of mortality prediction models. Santa Monica, CA: RAND.

Lemeshow, S. and J. R. Le Gall. 1994. Modeling the severity of illness of ICU patients: a systems update. *Journal of the American Medical Association* 272: 1049–1055.

Pollack M., U. E. Ruttimann, and P. R. Getson. 1988. Pediatric risk of mortality (PRISM) score. *Critical Care Medicine* 16: 1110–1116.

sg43	Modified t statistics

Richard Goldstein, Qualitas, Inc., EMAIL richgold@netcom.com

While it is widely known that t statistics are somewhat robust, it is not usually appreciated how narrow this robustness is. t tests generally are not robust to skewed data or to comparisons where the variance of one group is larger than the variance of the other or to comparisons where there is heterogeneity of effect (different observations are affected in different ways). It is particularly important to note that the rank-sum test is also not robust in many of the same situations. This insert presents two ado-files that provide alternative tests that perform better than the classical t test under these circumstances.

O'Brien's generalized t test

I have written two modified versions of the two-sample t test which is helpful in some cases. The first is `obrien`, an ado-file that implements O'Brien's generalized t test and generalized rank-sum test (O'Brien 1988). The syntax is

`obrien` *var* [`if` *exp*] [`in` *range*] , `by`(*groupvar*)

where *var* is the response variable and *groupvar*, which is required, is a two-category grouping variable.

O-Brien's test indicates whether there may be a problem with a t test or a ranksum test (heterogeneity of effect). A significant result for the quadratic term implies nonlinearity of effect and suggests the use of the generalized rather than the classical test. It is recommended that a p-value of 0.25 be used to determine whether there is nonlinearity. A significant result for the joint test means there is a significant difference in the location of the groups. `obrien` presents both an adjusted version of O'Brien's test and the original, unadjusted version. The simple adjustment used here is to multiply the original p-value by 1.45. This adjustment is discussed and extensive tables are given in Blair (1991). The last row of the display presents the standard results as per O'Brien's suggestion. Note that the p-value for the ranksum test will be slightly different than the value reported by Stata's `ranksum` command due to a slightly different treatment of ties. Note also that O'Brien's article always presents one-sided tests, while I always present two-sided tests.

The following example uses the familiar automobile data to illustrate `obrien`:

```
. use auto
(1978 Automobile Data)

. obrien price, by(foreign)
p-values for generalized-t and generalized ranksum
                      t:by OLS    t:by logistic   ranksum
quadratic term:         0.0602       0.0762        0.9785
generalize (no adj.):   0.1553       0.1648        0.5880
generalize (adj.):      0.2251       0.2389        0.8526
standard:               0.6802                     0.3012

. obrien mpg, by(foreign)
p-values for generalized-t and generalized ranksum
                      t:by OLS    t:by logistic   ranksum
quadratic term:         0.8189       0.9282        0.2108
generalize (no adj.):   0.0025       0.0095        0.0030
generalize (adj.):      0.0036       0.0138        0.0044
standard:               0.0005                     0.0015
```

Another modified t test

`modt` provides another modified t test (Brownie et al. 1990). The syntax of `modt` is

`modt` *var1* `=` *var2* [`if` *exp*] [`in` *range*]

— or —

`modt` *var* [`if` *exp*] [`in` *range*] , `by`(*groupvar*)

There is also an immediate version of this command:

`modti` *n1 mean1 sd1 n2 mean2*

`modt` is primarily useful for comparing the means of two groups where one group is a control group, and it is expected that the variance of the treatment group is larger than the variance of the control group. The control should be entered as *var1* in the first syntax. In the second syntax, the control group should have the lower value of *groupvar*; for instance, if the categories of the group variable are coded as zero and one, then zero should be the control group.

For the immediate version, first type the sample size, the mean, and standard deviation for the control group, then type the sample size and mean for the treatment group. There is no reason to enter the standard deviation for the treatment group.

This modification of the t test uses only the variance of the control group, though the means and numbers of observations from both groups are used.

Again we use the automobile data to illustrate this command.

```
. modt mpg, by(foreign)
modified t-statistic:  -4.100     df: 51
          2-sided p-value: 0.0001
          1-sided p-value: 0.0000

. modt price, by(foreign)
modified t-statistic:  -0.396     df: 51
          2-sided p-value: 0.6934
          1-sided p-value: 0.3467
```

References

Blair, R. C. 1991. New critical values for the generalized t and generalized rank-sum procedures. *Communications in Statistics—Simulation* 20: 981–994.

Brownie, C., D. D. Boos, J. and Hughes-Oliver. 1990. Modifying the t and ANOVA F tests when treatment is expected to increase variability relative to controls. *Biometrics* 46: 259–266.

O'Brien, P. C. 1988. Comparing two samples: extensions of the t, rank-sum, and log-rank tests. *Journal of the American Statistical Association* 83: 52–61.

sg44	Random number generators

Joseph Hilbe, Department of Sociology, Arizona State University, EMAIL atjmh@asuvm.inre.asu.edu
Walter Linde-Zwirble, Health Outcomes Technologies, EMAIL walterl22@aol.com

This insert presents ado-files we have written to implement random number generators (RNGs). These ado-files allow the user to generate synthetic data from a variety of distributions. Many of these generators employ the rejection method using a Lorenzean cover function.

Two different types of RNGs have been provided:

1. Standard random number generators, where the first term following the command is the desired number of observations and additional terms specify the parameters of the distribution.

An example of this first type of RNG is `rndbin`, which generates binomial random numbers. To generate a variable called `xb` that contains 20,000 pseudo-random draws from the binomial distribution with mean = .5, we type

```
. rndbin 20000 .5 1
( Generating . )
Variable xb created.
. summarize
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+--------------------------------------------------------
      xb |   20000      .50115   .5000112          0          1
```

2. Generators that rely on previously determined data and parameters. For this second type of RNG, a linear predictor is calculated which, using the inverse canonical link function of the distribution, determines a value of μ, the fitted value. This value of μ is then used to generate a random deviate appropriate to both the distribution and the data/parameters.

An example of this second type of RNG is `rndpoix`, which generates Poisson responses. For instance, define the linear predictor $y = 1 + .5|x_1| - .25|x_2|$ where x_1, x_2 are independent standard normal deviates. We generate 50,000 random Poisson responses for this case as follows:

```
. drop _all
. set obs 50000
obs was 0, now 50000
. generate x1 = abs(invnorm(uniform()))
. generate x2 = abs(invnorm(uniform()))
. generate lp = 1 + .5*x1 - .25*x2       /* create the linear predictor */
. generate mu = exp(lp)                  /* Poisson inverse link  */
. rndpoix mu
( Generating ............................ )
Variable xp created.
. glm xp x1 x2, f(poi) nolog
Residual df  =     49997                              No. of obs =     50000
Pearson X2   =  49351.31                              Deviance   =  53822.24
Dispersion   =  .9870854                              Dispersion =  1.076509
Poisson distribution, log link
------------------------------------------------------------------------------
      xp |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      x1 |   .5013978   .0034082    147.116   0.000       .4947179    .5080777
      x2 |  -.2563966   .0042539    -60.274   0.000      -.2647341   -.2480592
   _cons |   1.004081   .0051795    193.855   0.000       .9939293    1.014233
------------------------------------------------------------------------------
```

Note how similar the estimates are to the coefficients we specified in constructing the linear predictor.

Example: Constructing a probit model

The following example uses the binomial RNG to construct a probit model.

```
. drop _all
. set obs 20000
obs was 0, now 20000
```

```
. generate x1 = abs(invnorm(uniform()))
. generate x2 = abs(invnorm(uniform()))
. generate lp = .5 + .3*x1 - .6*x2        /* create the linear predictor */
. generate den = 1                        /* denominator = 1 (Bernoulli) */
. generate mu = normprob(lp)              /* Probit inverse link  */
. rndbinx mu den
( Generating . )
Variable bnlx created.
. glm bnlx x1 x2, f(bin den) l(probit) nolog
Residual df  =     19997                              No. of obs =     20000
Pearson X2   =  19997.04                              Deviance   =  25166.22
Dispersion   =  1.000002                              Dispersion =    1.2585
Binomial (N=den) distribution, probit link
------------------------------------------------------------------------------
    bnlx |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      x1 |   .2863205   .0159422     17.960   0.000       .2550743    .3175666
      x2 |   -.599287   .0158705    -37.761   0.000      -.6303926   -.5681814
   _cons |   .5106917   .0196748     25.957   0.000       .4721299    .5492536
------------------------------------------------------------------------------
```

Once again, the estimates are consistent with the coefficients we specified in the linear predictor. Moreover, note the χ^2 dispersion. Well-specified binomial models have a χ^2 dispersion approximating 1.0.

List of RNGs

Generators whose commands end in '`x`' provide the capability to model a complete synthetic data set. RNGs of this type are supplied for the following distributions: Poisson, binomial, gamma, and inverse Gaussian. An important caveat when working with any RNG: care must be taken that parameters are sensible for the distribution in question.

The table below lists the RNGs now available in Stata. The first two (uniform and normal) are part of Stata proper. The remainder are provided on the STB-28 distribution diskette. In the syntax diagrams, *obs* is the number of observations desired. The other parameters should be self-explanatory.

Table 1. List of random number generators

Distribution	Syntax
uniform	`uniform()`
normal	`invnorm(uniform())`
Student's t	`rndt` *obs df*
χ^2	`rndchi` *obs df*
F	`rndf` *obs df_n df_d*
log normal	`rndlgn` *obs mean var*
Poisson	`rndpoi` *obs mean* `rndpoix` [*mu*]
binomial	`rndbin` *obs prob numb* `rndbinx` [*prob*] *den*
gamma	`rndgam` *obs shape scale* `rndgamx` [*mu*] `, s(#)`
inverse Gaussian	`rndivg` *obs mean sigma* `rndivgx` [*mu*]`, s(#)`
exponential	`rndexp` *obs shape*
Weibull	`rndwei` *obs shape scale*
beta binomial	`rndbb` *obs denom prob k*

References

Evans, M. et al. 1993. *Statistical Distributions.* 2d ed. New York: John Wiley & Sons.

sg45	Maximum-likelihood ridge regression

Robert L. Obenchain, Eli Lilly and Company, Indianapolis, 317-276-3150

Ridge regression is a graphically oriented methodology for analysis of ill-conditioned (multicollinear) regression models. Ridge methods tend to be computationally intensive, especially when normal-theory maximum-likelihood estimation techniques are incorporated to provide objective information about the most appropriate form and extent of shrinkage. This insert presents an overview of ridge concepts along with five Stata programs to monitor the effects of shrinkage.

Ill-conditioning and ridge regression

Fitting of models to ill-conditioned data collected retrospectively poses serious obstacles to multiple regression practitioners, particularly in such fields as economics where interest can focus on the relative sizes of estimated coefficients. Consider the classical multiple regression model

$$y = 1\mu + X\beta + \epsilon \tag{1}$$

where y is an $n \times 1$ vector of observations on the response variable, μ is the unknown intercept, X is an $n \times p$ matrix containing coordinates for $p \geq 2$ nonconstant predictor variables, β is a $p \times 1$ vector of unknown coefficients, and ϵ is an $n \times 1$ vector of unobserved, normally-distributed disturbance terms

$$\epsilon \sim N(0, \sigma^2 I) \tag{2}$$

If the predictor variables are centered by subtracting off their observed means and the resulting X matrix of explanatory variables is of full column rank, then the maximum-likelihood estimate of β is the least-squares solution

$$\widehat{\beta} = (X'X)^{-1}X'y \tag{3}$$

It is straightforward to show that

$$\widehat{\beta} \sim N(\beta, \sigma^2(X'X)^{-1}) \tag{4}$$

Problems arise when X is ill-conditioned. Numerical ill-conditioning occurs when exact linear relationships exist between, say, the i-th and j-th explanatory variables. That is,

$$x_i = a + bx_j, \tag{5}$$

where a and b are constants. In this case, $X'X$ is singular, and $\widehat{\beta}$ is not uniquely determined.

More commonly, two or more X variables are highly correlated, and $X'X$ approaches singularity. In this situation, $\widehat{\beta}$ is unique but is imprecisely estimated. In other words, the relative magnitudes of the elements of $\widehat{\beta}$ may be distorted (they may even have "wrong" numerical signs), because the fitted coefficients are also highly correlated. As a result of this statistical ill-conditioning, elements of $\widehat{\beta}$ or certain linear combinations may be insignificant primarily because their variances are relatively large.

The topic of ill-conditioned regression models is one of the most thoroughly researched problems in statistics, and *ridge regression* is one approach that has been proposed to treat the symptoms of ill-conditioning. Ridge estimators shrink the estimated coefficient vector, $\widehat{\beta}$, and thus provide biased estimates of β. But variance is also reduced by shrinking, so ridge estimators can achieve lower mean squared error (MSE) risk than least squares.

An intuitive way to treat ill-conditioning is to increase the diagonal elements of $X'X$ before attempting to invert this inner products matrix and to form $\widehat{\beta}$ via equation (3) (Piegorsch and Casella 1989). Indeed, the original ridge estimator of Hoerl (1962) was

$$\beta^* = (X'X + kI)^{-1}X'y \tag{6}$$

where k is a small, positive constant.

Interest in ridge regression was sparked by Hoerl and Kennard (1970a, 1970b) when they suggested plotting the p elements of β^* as a function of k in a graphical display called the *ridge trace*. They observed that the relative magnitudes of the elements of β^* tend to stabilize as k increases and over-optimistically conjectured that it is easy to pick an extent of shrinkage yielding lower MSE than least squares. For more than twenty years now, a storm of criticisms, alternative proposals for choice of k, and ridge simulation studies have appeared in the statistical literature. If any sort of consensus has emerged, it may well be that (a) ridge methods tend to shrink much too much to be anywhere close to being minimax rules (that is, you can end up either

winning big by reducing MSE or else losing big by increasing MSE) and (b) the *generalized cross validation* method of Golub, Heath and Wahba (1979) for picking an appropriate extent of shrinkage is a consistent high-performer in simulation studies.

Classical, normal-theory maximum-likelihood estimation in generalized ridge regression has been a research interest of mine since 1973. In my first published ridge paper, Obenchain (1975), I derived general equations for *likelihood monitoring* that generated little interest, apparently due to their complexity. However, Gibbons (1981) did evaluate this *O-method* and found that it out-performed generalized cross validation in her "favorable case" MSE calculations. In Obenchain (1981), I restricted interest to a specific two-parameter family of generalized ridge estimators, given in equation (12) below, and derived a closed form expression for the extent of shrinkage along a given ridge path that is most likely to achieve minimum MSE risk (see equations (15) and (16) below). This maximum-likelihood approach to shrinkage is fairly conservative in the sense that it reduces the MSE risk by only about 50 percent even when its $\delta^{MSE} = 0$ (see equation (13) below), but this conservatism also means that the maximum-likelihood approach can increase MSE by at most 25 percent in the least favorable cases, usually somewhere in the $\delta^{MSE} = 0.8$ to $\delta^{MSE} = 0.9$ range. Ridge methods that shrink more aggressively than maximum likelihood tend either to do a little better or else much, much worse on MSE, depending upon whether the application is either favorable or unfavorable to shrinkage, respectively.

Other ridge research efforts of mine (Obenchain 1978, 1984) led to greater understanding of a variety of *multivariate risk* (matrix valued MSE) characteristics of shrinkage estimators, along with corresponding normal-theory maximum-likelihood estimates. Like ridge coefficients, these risk estimates can also be plotted in traces to display the effects of shrinkage and to help ridge practitioners decide whether to start shrinking in the first place and, once they start shrinking, where to stop.

Principal components and generalized ridge regression

This subsection contains technical details of generalized ridge estimation that may be skipped over on first reading. Here we show (i) how to decompose least squares estimates into uncorrelated components and principal correlations, (ii) why regression on principal components is a special case of generalized ridge regression, and (iii) how ridge estimators shrink least-squares coefficients along the principal axes of the given X coordinates.

Even in cases where X is numerically ill-conditioned—$\mathrm{rank}(X) = r < min(p, n-1)$—the singular value decomposition of X can be written as $X = H\Lambda^{1/2}G'$. In this decomposition, H is an $n \times r$ semi-orthogonal matrix of standardized *principal coordinates*, G is a $p \times r$ semi-orthogonal matrix of *principal axis direction-cosines*, and $\Lambda^{1/2}$ is an $r \times r$ diagonal matrix of ordered *singular values*, $\lambda_1^{1/2} \geq \cdots \geq \lambda_r^{1/2} > 0$.

Although the least-squares solution is not uniquely determined when $r < p$, the shortest least-squares coefficient vector is $\widehat{\beta} = G\Lambda^{-1/2}H'y \equiv Gc$, where c is the $r \times 1$ vector of *uncorrelated components* of $\widehat{\beta}$. Note that

$$c \sim N(\gamma, \sigma^2\Lambda^{-1}) \tag{7}$$

where $\gamma \equiv G'\beta$ are the r unknown *true components* of β. The structure of these uncorrelated components, $c = \Lambda^{-1/2}H'y$, provides key insights into the nature of statistical ill-conditioning:

$$c_i = r_i^o\sqrt{\frac{y'y}{\lambda_i}} \tag{8}$$

where r_i^o is the *principal correlation* between y and the i-th column of H, the familiar R-squared statistic is $R^2 = r_1^{o2} + \cdots + r_r^{o2}$, and the t-statistic for testing $\gamma_i = 0$ is

$$t_i = \frac{c_i}{\widehat{\sigma}\lambda_i^{-1/2}} = r_i^o\sqrt{\frac{n-r-1}{(1-R^2)}} \tag{9}$$

Thus the i-th principal correlation, r_i^o, determines whether the i-th component is statistically significant, and yet c_i can be large numerically simply because its λ_i is relatively small rather than because its r_i^o is relatively large.

Linear generalized ridge estimates are of the form

$$\beta^* = G\Delta c = \sum g_i\delta_i c_i \tag{10}$$

where Δ is an $r \times r$ diagonal matrix of *non-stochastic shrinkage factors*, $\delta_1, \cdots, \delta_r$, and g_i is the i-th column of G. Each shrinkage factor lies in the closed interval from zero to one, $0 \leq \delta_i \leq 1$, and the total extent of shrinkage is measured by

$$m = r - \delta_1 - \cdots - \delta_r = \mathrm{rank}(X) - \mathrm{trace}(\Delta) \tag{11}$$

This m is called the *multicollinearity allowance* ridge parameter, introduced and discussed in Obenchain and Vinod (1974), Vinod (1976) and Obenchain (1981). Ridge coincides with least squares at $m = 0$ ($\delta_1 = \cdots = \delta_r = 1$), and all ridge coefficients approach zero as m approaches its upper limit of $m = p$ ($\delta_1 = \cdots = \delta_r = 0$).

Regression on principal components is the special case of equation (10) in which each δ_i is either 0 or 1. Standard methods for deciding which δ_i to set equal to zero are: (a) the components with the smallest singular values, $\lambda_i^{1/2}$, or (b) the components with the smallest absolute principal correlations, $|r_i^o|$.

Our primary focus will be on the *two-parameter ridge family* in which the shrinkage factor applied to the i-th uncorrelated component of the least-squares solution is of the general form

$$\delta_i = \lambda_i / (\lambda_i + k\lambda_i^q), \tag{12}$$

where k is nonnegative and q is a finite power that determines the shape (or curvature) of the ridge path through p-dimensional space (Goldstein and Smith 1974). The "ordinary" ridge estimators of equation (6) correspond to $q = 0$ in equation (12).

The two-parameter family is quite versatile in the sense that most shrinkage paths considered in ridge regression literature are either special cases or limiting cases of this family. For example, $q = 1$ yields uniform shrinkage, $\delta_1 = \cdots = \delta_p$. In actual ridge practice, ties among eigenvalues are rare (except in designed experiments.) The common situation is $\lambda_1 > \cdots > \lambda_r > 0$ where $r = \mathrm{rank}(X) \leq p$, and the first r shrinkage factors are then all unequal as long as $m > 0$, $m < r$ and $q \neq 1$ in equation (12). Note that $q > 1$ would focus initial shrinkage upon major principal axes, $\delta_1 < \cdots < \delta_r$, while $q < 1$ focuses initial shrinkage along minor axes, $\delta_1 > \cdots > \delta_r$. These $q < 1$ (declining δ) shrinkage patterns, when favored by the y data, have much greater potential for reduction in MSE risk via variance-bias trade-offs than do the $q > 1$ patterns.

The limit as q approaches $+\infty$ is optimal for the Gibbons (1981) "unfavorable case" where the true β vector lies along the eigenvector corresponding to the smallest regressor eigenvalue, λ_r. And the limit as q approaches $-\infty$ is essentially what Marquardt (1970) called "assigned-rank" regression. In both of these limiting cases, the shrinkage path travels along a series of "edges" of the principal-components regression hyper-rectangle. I have found that $q = \pm 5$ is usually adequate to roughly approximate these $q = \pm\infty$ limiting cases.

It is easily shown that the unknown extent of shrinkage that minimizes the MSE risk of $\delta_i c_i$ as a linear estimate of γ_i is

$$\delta_i^{MSE} = \frac{\gamma_i^2}{\gamma_i^2 + (\sigma^2/\lambda_i)} = \frac{\lambda_i}{\lambda_i + (\sigma^2/\gamma_i^2)} \tag{13}$$

These equations can be solved to express γ_i and σ as functions of λ_i and of any trial value for the δ_i shrinkage factor to yield

$$\gamma_i = \pm\sigma\sqrt{\frac{\delta_i}{\lambda_i(1-\delta_i)}} = \frac{\pm\sigma}{\sqrt{k\lambda_i^q}} \tag{14}$$

where the last expression follows only for ridge estimators in the two-parameter family of equation (12). The likelihood that any given ridge estimation minimizes MSE risk is then defined by maximizing, by choice of the $\widehat{\sigma}$ estimate and the $\pm$ signs, the likelihood that γ is of the form given in equation (14). The resulting closed-form solution, Obenchain (1981), is

$$\widehat{k} = \widehat{k}(q) = \left(\sum \lambda_j^{(1-q)}\right) \frac{1 - R^2\mathrm{CRL}^2(q)}{nR^2\mathrm{CRL}^2(q)} \tag{15}$$

where the "curlicue" function is

$$\mathrm{CRL}(q) = \frac{\sum |r_j^o| \, \lambda_j^{(1-q)/2}}{\sqrt{\sum r_j^{o2} \sum \lambda_j^{(1-q)}}} \tag{16}$$

Furthermore, the most likely q-shape is the one that maximizes $\mathrm{CRL}(q)$ (Obenchain 1975) AND CAN be found by numerical search. Note that these maximum-likelihood ridge estimators are more versatile than principal components regression in the sense that they use both the r_i^o and the $\lambda_i^{1/2}$ to select shrinkage factors anywhere in the range $0 \leq \delta_i < 1$.

A reasonable way to plot traces for the family of equation (12) is first to decide which p quantities will be plotted vertically, then to fix the value of the shape parameter q, and finally to plot with m of equation (11) on the horizontal axis over the range from $m = 0\,(k = 0)$ to $m = r\,(k = +\infty)$. In the Stata ado files, numerical values for the k parameter are determined implicitly given values for m and q. And all trace displays use m of equation (11) on the horizontal axis instead of k so that traces will not only have finite width but will also be easier to compare for different choices of q-shape.

The Stata ado-files

This insert presents five Stata programs to estimate and evaluate models using maximum-likelihood ridge regression:

`rxrcrlq` determines which q-shape (curvature) for the shrinkage path is most likely to contain the MSE-optimal ridge estimator. `rxrcrlq` searches a user-specified lattice of q-shapes within the range $-5 \leq q \leq +5$.

`rxridge` performs generalized ridge calculations and displays five types of traces of specified q-shape: (i) shrunken coefficients (β_i^* of equation (10)), (ii) estimated scaled (or relative) MSE, (iii) excess MSE eigenvalues (OLS minus ridge), (iv) inferior-direction cosines, and (v) ridge shrinkage δ-factors.

`rxrmaxl` computes three types of likelihood criteria to determine an ideal extent of shrinkage along a path of given q-shape: (i) classical, (ii) random-coefficients, and (iii) empirical Bayes.

`rxrrisk` computes and displays, for specified true model parameters γ and σ and specified path q-shape, five types of traces: (i) expected coefficients, (ii) true, scaled MSE, (iii) true excess MSE eigenvalues, (iv) the true inferior direction, and (v) ridge shrinkage δ-factors.

`rxrsimu` generates, for given model parameters and specified path q-shape, pseudo-random responses and a trace of the resulting true scaled, squared-error losses from shrinkage.

Note that the first three programs—`rxrcrlq`, `rxridge` and `rxrmaxl`—are the ones you should find most useful for data analysis and statistical inference. However, you may use the last two programs—`rxrrisk` and `rxrsimu`—to convince yourself that the estimation methods incorporated in the first three programs can be expected to work well in actual practice; in fact, this is the approach that we will use in this insert.

Our exploration of the Stata programs for likelihood-based ridge regression is organized as follows. First, we show how true expected risks and simulated losses, respectively, can be expected to vary upon shrinkage when the regression parameters, β and σ, have known values. Unfortunately, these two preliminary sections have little to do with the usual analysis/inference situation in which ridge regression is actually applied, that is, when the unknown regression parameters are to be estimated from an observed response variable conditional on given predictor variables. On the other hand, we will see later that traces of maximum-likelihood estimates of unknown, true MSE risks can mimic the most important features of their population analogs. And illustrating this phenomenon certainly enhances the credibility of our graphical/likelihood approach to ridge regression analysis.

The formal syntax diagrams and listings of options are reserved for the final section of the insert.

The Portland Cement data

The remainder of this insert uses the *Portland Cement* data of Hald (1952) to illustrate use of the five Stata programs. This data set is well known and also quite small (only $n = 13$ observations on $p = r = 4$ predictor variables named `v3CA`, `v3CS`, `v4CAF` and `v2CS`.) This is the same numerical example I used in Obenchain (1984) to illustrate that one's data can suggest use of a relatively extreme path shape; here, our motivation for using the $q = -5$ path shape will be postponed until we illustrate the usage of `rxrcrlq.ado`.

Known model parameters: `rxrrisk`

Let us assume that the true values of the uncorrelated components of β are $\gamma = G'\beta = (.646, .0, .323, .108)'$ and the true error standard deviation is $\sigma = .215$. These numerical values are fairly close to their least squares estimates for the `heat` response variable of Hald (1952); namely, $\widehat{\gamma} = (.657, .008, .303, .388)'$ and $\widehat{\sigma} = .163$. The example below uses `rxrrisk` to display expected shrinkage results (Figures 1–4) for the $q = -5$ family of (12). We begin by using the Portland Cement data and loading the matrices `rxgamma` and `rxsigma` with the values assumed for γ and σ, respectively.

```
. use haldcemt
. matrix rxsigma = (.215)
. matrix rxgamma = (.646,.0,.323,.108)
. rxrrisk heat v3CA v3CS v4CAF v2CS, q(-5) m(2) t(0.001)
RXrrisk:  Shrinkage Path has Qshape =-5.00
RXrrisk:  Estimated Sigma = .16259326
RXrrisk:  Estimated Uncorrelated Components...
c[1,4]
           c1         c2         c3         c4
c1  .65695805  .00830862  .30277026  .38803631
RXrrisk:  True Sigma =      .215
RXrrisk:  True Uncorrelated Components...
rxgamma[1,4]
      c1    c2    c3    c4
r1  .646     0  .323  .108
RXrrisk will now rescale the true sigma and components,
preserving all signal/noise ratios, so as to equate the
expected response sum-of-squares to yTy =        12
RXrrisk:  Rescaled True Sigma = .21513909
RXrrisk:  Rescaled True Uncorrelated Components...
rxgamma[1,4]
           c1         c2         c3         c4
r1   .6464179          0  .32320895  .10806987
MCAL =  0.000 ... True  OLS Summed SMSE = 51.858386
MCAL =  0.500 ... True Summed SMSE Risk =  13.43022
MCAL =  1.000 ... True Summed SMSE Risk = .78905158
MCAL =  1.500 ... True Summed SMSE Risk = 1.0183621
MCAL =  2.000 ... True Summed SMSE Risk = 2.5913642
MCAL =  2.500 ... True Summed SMSE Risk = 2.6182079
MCAL =  3.000 ... True Summed SMSE Risk = 3.1409814
MCAL =  3.500 ... True Summed SMSE Risk = 5.5661976
MCAL =  4.000 ... True Summed SMSE Risk = 11.537242
RXrrisk:  Expected Coefficients...
obs was 0, now 9
(Note: file rxrrisk1.dta not found)
file rxrrisk1.dta saved
```

(graph appears, see Figure 1)

```
RXrrisk:  True Scaled MSE Risk...
obs was 0, now 9
(Note: file rxrrisk2.dta not found)
file rxrrisk2.dta saved
```

(graph appears, see Figure 2)

```
RXrrisk:  True Excess Eigenvalues...
obs was 0, now 9
(Note: file rxrrisk3.dta not found)
file rxrrisk3.dta saved
```

(graph appears, see Figure 3)

```
RXrrisk:  True Inferior Direction Cosines...
obs was 0, now 9
(Note: file rxrrisk4.dta not found)
file rxrrisk4.dta saved
```

(graph appears, see Figure 4)

```
RXrrisk:  Shrinkage DELTA Factors...
obs was 0, now 9
(Note: file rxrrisk5.dta not found)
file rxrrisk5.dta saved
```

(graph appears, not shown)

`rxrrisk` begins by echoing the q-shape selected using `q()` option. For this example, we have selected $q = -5$. Next the OLS estimates $\widehat{\sigma}$ and $\widehat{\gamma}$ are displayed, followed by the "true" values we loaded. Next, `rxrrisk` steps through selected values of m (labeled `MCAL`). The option `m(2)` (described later) controls the number of steps. The option `t(0.001)` sets the search convergence criterion.

The results of `rxrrisk` are stored in Stata matrices. `rxrrisk` converts these matrices to Stata data sets (using `rxrmkdta`, a utility routine) and saves these data sets under the names `rxrrisk1.dta`, `rxrrisk2.dta`, ..., `rxrrisk5.dta`. (If data sets with these names already exist, they are replaced.) The statistics stored in these data sets are displayed as Stata graphs, four of

which appear, in slightly edited form, as Figures 1–4. The fifth graph is omitted. All five graphs are also stored on disk under the names `rxrrisk1.gph`, `rxrrisk2.gph`, ..., `rxrrisk5.gph`. (Again, previously stored graphs with these names will be replaced.)

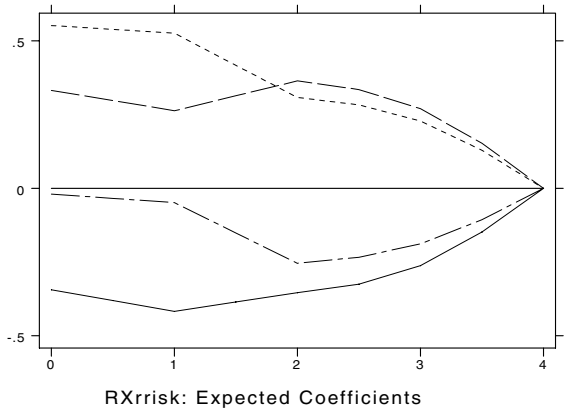

Figure 1.

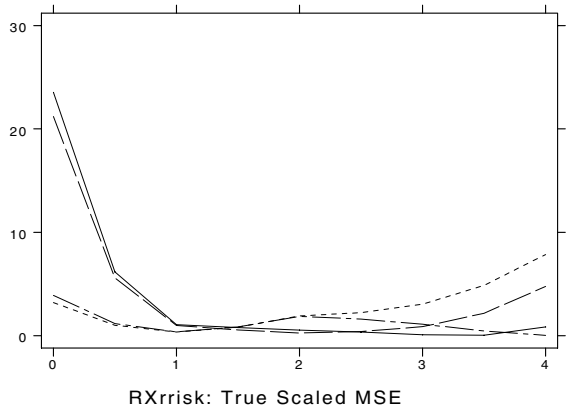

Figure 2.

Figure 1 shows how the expected values of the ridge coefficients change as bias is introduced via the $q = -5$ family. The dashed line that starts at the top left displays the trace for the predictor variable `v3CA` (in this, and in succeeding, figures). The line with the long dashes that also lies above zero displays the trace for `v3CS`. The line with mixed dots and dashes displays the trace for `v4CAF`. The solid line displays the trace for `v2CS`. The true regression coefficient vector, $\beta = G\gamma = (.552, .332, -.020, -.344)'$, is displayed at $m = 0$ in Figure 1. Thus the first two, true coefficients are positive while the last two are negative. In particular, note that bias introduced by shrinkage along the $q = -5$ path can tend to make β_3 somewhat more negative than its true value of $-.020$.

Figure 2 gives the associated scaled (or "relative") MSE risks defined as follows. Risk is expected loss (or mean squared error), and the scaled risk values plotted in Figure 2 are the diagonal elements of the mean squared error matrix each divided by σ^2. Scaled risk values measure the uncertainty in an estimate as a multiple of the variance of a single observation. Scaled risk values also have the advantage of being known values for least-squares estimates even when regression parameters are unknown. For example, the values at the left extreme $(m = 0)$ of Figure 2 are the diagonal elements of $(X'X)^{-1}$, which do not depend upon $\beta = G\gamma$ or σ.

The eigenvalues of the difference in scaled risk matrices (least squares minus ridge) are displayed in Figure 3. As long as these eigenvalues are all nonnegative, no linear combination of least squares coefficients has smaller risk than the corresponding linear combination of ridge coefficients.

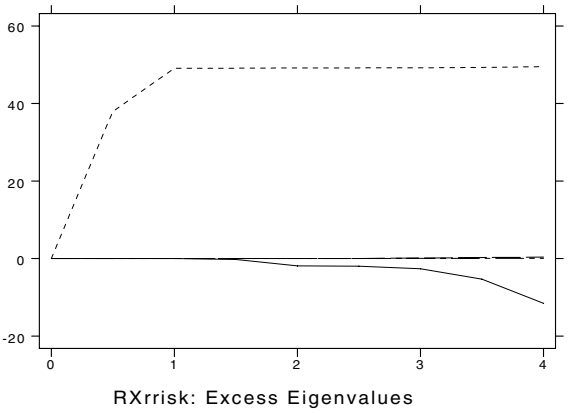

Figure 3.

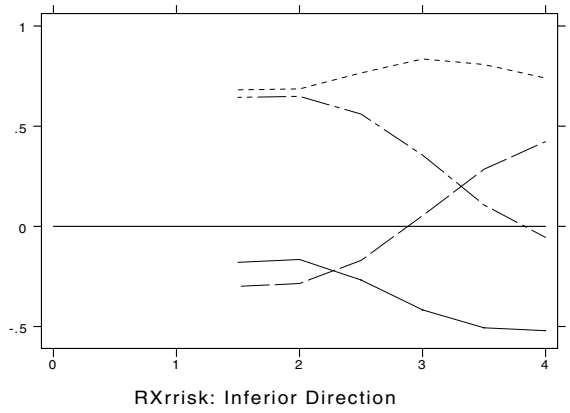

Figure 4.

At most one excess eigenvalue can be negative (Obenchain 1978), and the corresponding normalized eigenvector (Figure 4) points in the inferior-direction of p-dimensional space along which ridge has higher risk than least squares. For example, the first and third direction cosines (for variables `v3ca` and `v4caf`) are nearly equal when an inferior direction appears at $m = 1.5$ in Figure 4. Linear combinations like $\beta_1 - \beta_3$ are thus essentially orthogonal to the inferior direction at this m, so the ridge estimate of $\beta_1 - \beta_3$ probably has lower risk than least squares. But the ridge estimate of $\beta_1 + \beta_3$ near $m = 1.5$ can possibly have higher risk than least squares, because this linear combination has a relatively large projection onto the inferior direction.

Figures 1 through 4 indicate that our numerical example is amenable to ridge shrinkage with $q = -5$ in equation (12). The trace of the scaled risk matrix decreases from 51.8 at $m = 0$ to 0.787 at $m = 1.0$ and then starts increasing again. Thus an m value of about 1 is risk optimal when $q = -5$, and this is like saying that ill-conditioning has effectively reduced the rank of the regressor matrix by one, from four to three. This makes very good sense in this example because the regressor matrix comes from a "mixture" equipment with the four regressors adding (except for a relatively large round-off error) to 100 percent.

Simulated responses: `rxrsimu`

The logical next step in exploring our numerical example is to use pseudo-random numbers to simulate a response vector for this model using procedure `rxrsimu`. The results from a single invocation of `rxrsimu` are displayed in the listing below and in Figure 5.

```
. use haldcemt
. matrix rxsigma = (.215)
. matrix rxgamma = (.646,.0,.323,.108)
. rxrsimu heat v3CA v3CS v4CAF v2CS, q(-5) m(2)
RXrsimu:  Shrinkage Path has Qshape =-5.00
RXrsimu:  Estimated Sigma = .16259326
RXrsimu:  Estimated Uncorrelated Components...
c[1,4]
           r1         r2         r3         r4
c1  .65695805  .00830862  .30277026  .38803631
RXrsimu:  True Sigma = .21513909
RXrsimu:  True Uncorrelated Components...
rxgamma[1,4]
           c1         c2         c3         c4
r1   .6464179          0  .32320895  .10806987
RXrsimu will now rescale the true sigma and components,
preserving all signal/noise ratios, so as to equate the
expected response sum-of-squares to yTy =        12
RXrsimu:  Rescaled True Sigma = .21513909
RXrsimu:  Rescaled True Uncorrelated Components...
rxgamma[1,4]
           c1         c2         c3         c4
r1   .6464179          0  .32320895  .10806987
RXrsimu:  Simulated Sigma = .21311153
RXrsimu:  Simulated Uncorrelated Components...
c[1,4]
            r1          r2          r3          r4
c1   .65260458  -.02292297   .27097722   1.3641871
MCAL =  0.000 ... True OLS Summed SSE Loss = 34.078349
MCAL =  0.500 ... True Summed SSE Loss = 7.1773437
MCAL =  1.000 ... True Summed SSE Loss = .32067964
MCAL =  1.500 ... True Summed SSE Loss = 1.0210648
MCAL =  2.000 ... True Summed SSE Loss = 2.5054672
MCAL =  2.500 ... True Summed SSE Loss = 2.5509614
MCAL =  3.000 ... True Summed SSE Loss = 3.0736594
MCAL =  3.500 ... True Summed SSE Loss = 5.4985368
MCAL =  4.000 ... True Summed SSE Loss =      11.5
RXrsimu:  Simulated Shrinkage Coefficients...
obs was 0, now 9
(Note: file rxrsimu1.dta not found)
file rxrsimu1.dta saved
RXrsimu:  Scaled True Squared Error Losses...
obs was 0, now 9
(Note: file rxrsimu2.dta not found)
file rxrsimu2.dta saved
```

(graph appears, see Figure 5)

```
RXrsimu:  Shrinkage DELTA Factors...
obs was 0, now 9
(Note: file rxrsimu5.dta not found)
file rxrsimu5.dta saved
```

(graph appears, not shown)

```
. list ysim yexp
```

	ysim	yexp
1.	-1.123823	-1.115586
2.	-1.251552	-1.477653
3.	.6228011	.7167799
4.	-.4981516	-.372227
5.	-.0099682	-.0052288
6.	.4211500	.651238
7.	.3680945	.5466103
8.	-1.119887	-1.291865
9.	-.2948058	-.2422897
10.	1.434455	1.351955
11.	-1.182420	-.8971266
12.	1.404445	1.091739
13.	1.229661	1.043654

There are some similarities in the operation of `rxrrisk` and `rxrsimu`. Both commands require that the true values of σ and γ are specified in advance. Some of the initial output is identical in both commands as well. Like `rxrrisk`, the `rxrsimu` commands automatically saves Stata data sets (`rxrsimu1.dta`, `rxrsimu2.dta`, and `rxrsimu5.dta`) and Stata graphs (`rxrsimu2.gph` and `rxrsimu5.gph`). `rxrsimu` adds two variables to the existing data set. `ysim` contains the simulated values of the response, while `yexp` contains the expected values.

Note that minimum overall loss occurs at about $m = 1.0$ in Figure 5. Also, remember that the expected value of the scaled, squared-error loss trace of Figure 5 would be the scaled MSE risk trace of Figure 2.

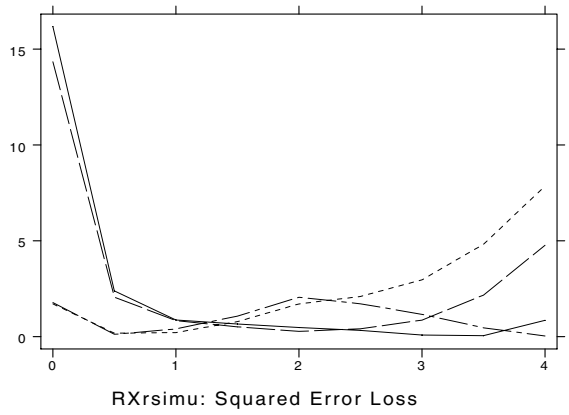

Figure 5.

Data analysis/inference: `rxridge`

Let us now continue our numerical example by analyzing the simulated responses (`ysim`) using procedure `rxridge` as if we had no knowledge of the true regression parameter settings used to generate the data.

```
. rxridge ysim v3CA v3CS v4CAF v2CS, q(-5) m(2) t(0.001)
RXridge:  Shrinkage Path has Qshape =-5.00
RXridge:  Adjusted response sum-of-squares =        12
RXridge:  OLS Residual Variance = .04541652
RXridge:  Variance of Principal Correlations = .00378471
MCAL =  0.000 ... True  OLS Summed SMSE = 51.858386
MCAL =  0.500 ... Estimated Summed SMSE = 13.367138
MCAL =  1.000 ... Estimated Summed SMSE = .53672157
MCAL =  1.500 ... Estimated Summed SMSE = 2.4061363
```

```
MCAL =  2.000 ... Estimated Summed SMSE =  4.819774
MCAL =  2.500 ... Estimated Summed SMSE = 4.8481264
MCAL =  3.000 ... Estimated Summed SMSE =   5.17465
MCAL =  3.500 ... Estimated Summed SMSE = 9.8344084
MCAL =  4.000 ... Estimated Summed SMSE = 18.568804
RXridge:  Shrinkage Coefficients...
obs was 0, now 9
(Note: file rxridge1.dta not found)
file rxridge1.dta saved
```

(graph appears, see Figure 6)

```
RXridge:  Scaled MSE Risk Estimates...
obs was 0, now 9
(Note: file rxridge2.dta not found)
file rxridge2.dta saved
```

(graph appears, see Figure 7)

```
RXridge:  Excess Eigenvalue Estimates...
obs was 0, now 9
(Note: file rxridge3.dta not found)
file rxridge3.dta saved
```

(graph appears, see Figure 8)

```
RXridge:  Inferior Direction Cosine Estimates...
obs was 0, now 9
(Note: file rxridge4.dta not found)
file rxridge4.dta saved
```

(graph appears, see Figure 9)

```
RXridge:  Shrinkage DELTA Factors...
obs was 0, now 9
(Note: file rxridge5.dta not found)
file rxridge5.dta saved
```

(graph appears, not shown)

```
RXridge:  Estimated Sigma = .21311153
RXridge:  Uncorrelated Components...                    Number of obs =      13
------------------------------------------------------------------------------
    ysim |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      c1 |   .6526046   .0411443     15.861   0.000        .5577258    .7474834
      c2 |  -.022923   .0490037     -0.468   0.652       -.1359258    .0900798
      c3 |   .2709772   .1424142      1.903   0.094       -.0574305     .599385
      c4 |   1.364187   1.526713      0.894   0.398       -2.156419    4.884793
------------------------------------------------------------------------------
```

`rxridge` saves Stata data sets and graphs using naming conventions analogous to those used by `rxrrisk` and `rxrsimu`.

Note that only the estimates of the first and possibly third uncorrelated components are statistically significant, but the fourth component is huge numerically. Thus our `rxrsimu`-generated response vector is even more susceptible to ill-conditioning in X than the original `heat` data of Hald (1952).

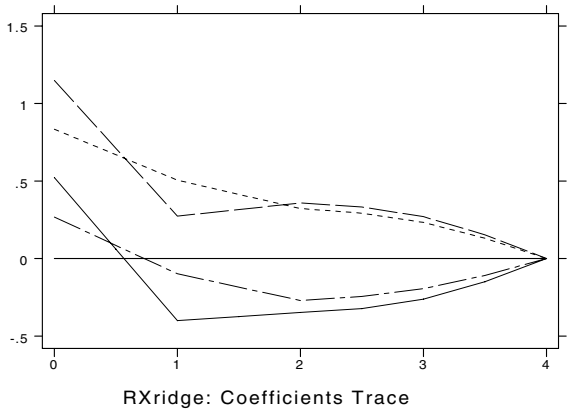

Figure 6.

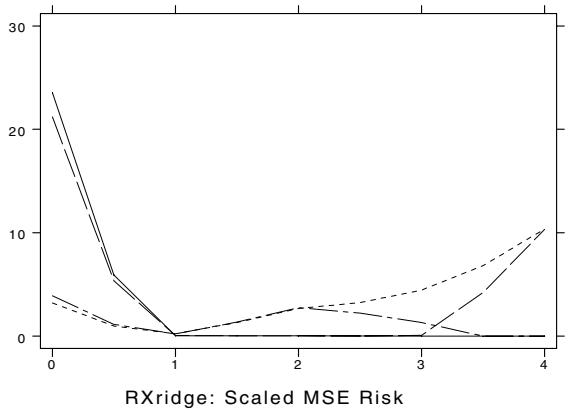

Figure 7.

Note in Figure 6 that all four least-squares estimates for coefficients are positive ($m = 0$) and that shrinkage to at least $m = 1$ is required to produce ridge coefficients with the "right" numerical signs.

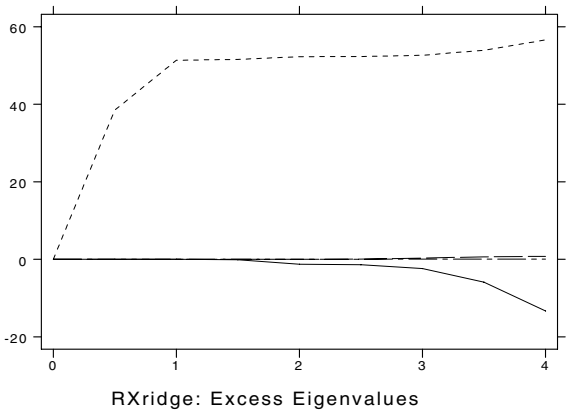

Figure 8.

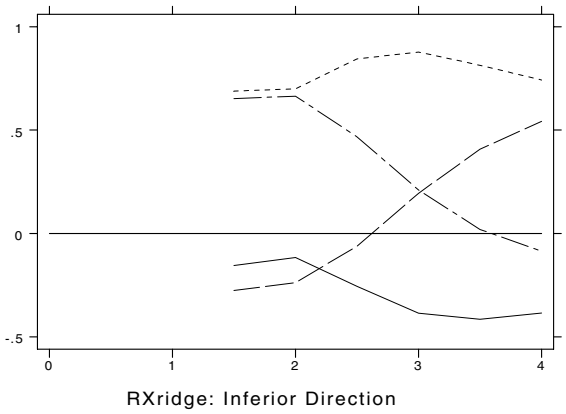

Figure 9.

Figures 7–9 are traces of estimates of scaled risks, excess eigenvalues, and inferior direction cosines, respectively (Obenchain 1978, 1981). These traces are all based upon normal-theory maximum-likelihood, but scaled risk estimates have first been adjusted using known constants that make them unbiased and then truncated, if necessary, so as to have correct range (no scaled risk estimate is given that is below its scaled variance lower limit).

Visual examination of Figures 6–9 suggests that the ridge solution at $m = 1.0$ in the $q = -5$ family has much more desirable risk characteristics than does the least squares solution.

Shrinkage path shape: `rxrcrlq`

So far we have plotted traces using only shape $q = -5$ in (12). This is because $q = -5$ was the MSE-optimal shrinkage path shape for the original Hald (1952) data. And we can use procedure `rxrcrlq` to verify that this is also the best choice for the `rxrsimu` generated data.

```
. rxrcrlq ysim v3CA v3CS v4CAF v2CS
RXrcrlq:  Estimated Sigma = .21311153
RXrcrlq:  Uncorrelated Components...                    Number of obs =      13
------------------------------------------------------------------------------
    ysim |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      c1 |   .6526046   .0411443     15.861   0.000       .5577258    .7474834
      c2 |   -.022923   .0490037     -0.468   0.652      -.1359258    .0900798
      c3 |   .2709772   .1424142      1.903   0.094      -.0574305     .599385
      c4 |   1.364187   1.526713      0.894   0.398      -2.156419    4.884793
------------------------------------------------------------------------------

RXrcrlq: Classical, Normal-Theory, Maximum-Likelihood Choice of Q=>Shape
and MCAL=>Extent of Shrinkage in Generalized Ridge Regression...
------------------------------------------------------------------------
The curlicue function, CRL(Q), is the (nonnegative) Correlation
between the R-vector of absolute values of the principal correlations
of regressors with the response and the L-vector of regressor spread
eigenvalues raised to the power (1-Q)/2.
------------------------------------------------------------------------

qvec[21,5]
        Qshape       MCAL      Konst     CRL(Q)      ChiSq
 r1          5  3.9620787  1.760e+08  .05583205  45.426121
 r2        4.5  3.9620482   24548041  .05585532  45.426088
 r3          4   3.961938  3416311.2  .05593933  45.425969
 r4        3.5  3.9615135  471414.18  .05626175  45.425512
 r5          3  3.9597236  62767.277   .0576044  45.423578
 r6        2.5  3.9511002  7157.8975  .06373354  45.414169
 r7          2  3.8986512  460.94542  .09406287  45.353457
 r8        1.5  3.4958991  10.524657  .24223801  44.703863
```

```
 r9          1  1.4705107  .58134686  .59741232  39.943159
r10         .5  1.4471443   .7457153  .74293599  35.504378
r11          0  1.7167549  2.4715085  .78595994  33.585201
r12        -.5  1.9154949  9.7502148  .81188967  32.212621
r13         -1  2.0447908  40.498435  .83293916   30.94283
r14       -1.5  2.1009248  171.06443  .85209421  29.635774
r15         -2  2.1182805  727.90415  .86989888  28.259285
r16       -2.5  2.1205947  3115.3572  .88627757  26.820271
r17         -3  2.1182927  13417.735  .90110205  25.337028
r18       -3.5  2.1152813   58203.02  .91432178  23.829866
r19         -4  2.1129101  254461.68  .92596664  22.318394
r20       -4.5  2.1116114  1121925.8   .9361225  20.820911
r21         -5  2.1115136  4990888.1  .94490765  19.354315
The most likely Qshape = -5.00 achieves Maximum CRL(Q) and Minimum ChiSq.
This Min ChiSq has degrees-of-freedom =  2 and sig.level =0.000
------------------------------------------------------------------------
In multiple regression models where the Minumum ChiSq is significantly
greater than zero, the 2-parameter generalized ridge family is probably
too restrictive (unlikely to contain the MSE optimal shrinkage factors.)
------------------------------------------------------------------------
```

`rxrcrlq` uses the maximum-likelihood equations (15) and (16) to evaluate alternative q-shapes for the shrinkage path. Here the most likely q-shape is -5 because it achieves maximum $\mathrm{CRL}(q)$ and minimum χ^2 in the listing above. This minimum χ^2 has two degrees of freedom and a significance level that is zero to three decimal places. Thus the two-parameter generalized ridge family is probably too restrictive (unlikely to contain the MSE optimal shrinkage factors) for this example.

Is there a family of shrinkage (δ) factors "less restrictive" than equation (12) that we could consider? Not really; I know of no proposals for, say, a three-parameter family. A full r-parameter solution (in which each δ factor is estimated separately) is possible, but this would impose no "smoothness" requirements whatsoever on shrinkage factors. (In the current example, the data strongly suggest taking δ_2 much smaller than δ_1 or δ_3, but there is very little potential for MSE reduction by such a "greedy" tactic because $\widehat{\gamma}_2 = -.02$ is already tiny, numerically.) Besides, r-parameter estimates are not amenable to visual display using ridge traces.

It is a straightforward task to generate traces for several different values of q and to make a choice (either objective or subjective) of the shape one likes best. These traces can change shape and thus interpretation quite drastically as q changes. Obviously unfavorable choices of q will have minimum SMSE risk either at or very close to $m = 0$ in Figures 2 and 7. Furthermore, a negative excess eigenvalue will not only appear for very small m values in Figures 3 and 8 when q is unfavorable, but this negative eigenvalue will also dominate the most positive eigenvalue in absolute magnitude. Anyway, the most likely q-shape (which is -5 in our current example) usually strikes me as being adequately general even when the `rxrcrlq` χ^2 statistic is significantly greater than zero.

A search on a finer lattice of q values over a wider range than $-5 \leq q \leq +5$ could be considered, of course, but we must remember that the primary purpose of the `rxrcrlq` calculations is simply to interest us in examining the corresponding trace displays.

Shrinkage extent: `rxrmaxl`

Other maximum-likelihood approaches besides the classical, fixed-coefficient approach of Obenchain (1975, 1981) are possible, but they do not yield closed form expressions for the optimal k or m given q. The empirical Bayes approach of Efron and Morris (1977) and the random coefficient method of Golub, Heath, and Wahba (1979) and Shumway (1982) are two maximum-likelihood alternatives implemented in `rxrmaxl`. This program "monitors" all three of the above likelihood criteria on a lattice of m values, referring to them as `CLIK`, `EBAY`, and `RCOF`, respectively.

```
. rxrmaxl heat v3CA v3CS v4CAF v2CS, q(-5) m(2) t(0.001)
RXrmaxl:  Shrinkage Path has Qshape =-5.00
RXrmaxl:  Estimated Sigma = .21311153
RXrmaxl:  Uncorrelated Components...                  Number of obs =      13
------------------------------------------------------------------------------
    ysim |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      c1 |   .6526046   .0411443     15.861   0.000       .5577258    .7474834
      c2 |   -.022923   .0490037     -0.468   0.652      -.1359258    .0900798
      c3 |   .2709772   .1424142      1.903   0.094      -.0574305     .599385
      c4 |   1.364187   1.526713      0.894   0.398      -2.156419    4.884793
------------------------------------------------------------------------------
```

```
RXrmaxl:  3 Normal, Maximum-Likelihood Shrinkage Criteria...
(Classical, Empirical Bayes, and Random Coefficients)
MCAL =  0.500
MCAL =  1.000
MCAL =  1.500
MCAL =  2.000
MCAL =  2.500
MCAL =  3.000
MCAL =  3.500
MCAL =  4.000
RXrmaxl:  Listings of Three Minus-2-Log-Likelihood Ratios...
CLIK[9,3]
           CLIK        EBAY        RCOF
  0  1.000e+100  1.000e+100  1.000e+100
 .5   1.742e+12   83.985422   84.219263
  1   9.889e+11   74.129038   74.567306
1.5   445435.87   31.003878   32.064207
  2   738.64505   19.480671   20.780788
2.5   27.939063    28.33019   21.782698
  3   35.164964   71.501773   31.229391
3.5   39.875945   151.47765   39.478093
  4   45.465477   256.22102   45.465477
------------------------------------------------------------------------
The Maximum Likelihood choices for MCAL=>Extent of Shrinkage are
the ones that Minimize the CLIK, EBAY or RCOF criteria, above.
------------------------------------------------------------------------
```

The maximum-likelihood choices for m-extent of shrinkage are the ones that minimize the `CLIK`, `EBAY` or `RCOF` criteria, above. I used `rxrmaxl` above to display these criteria for the `ysim` variable generated by `rxrsimu` within the $q = -5$ family. `CLIK` is a minus two log likelihood ratio whose minimum has an asymptotic χ^2 distribution with two degrees of freedom. `EBAY` and `RCOF` are normalized so they cannot become negative, but their minima apparently do not have asymptotic χ^2 distributions. Anyway, the `EBAY` and `RCOF` criteria both favor $m = 2$ when $q = -5$ while `CLIK` favors $m = 2.5$. Thus, using the "$2/r$-ths Rule-of-Thumb" of Obenchain (1978) for the extent of shrinkage likely to be good (that is, likely to dominate least-squares in a matrix MSE sense) in this $p = r = 4$ predictor model, we again find that shrinkage to at least $m = 2 \times 2/r = 1$ is highly desirable.

The primary reservation that comes to my mind concerning the `rxrmaxl.ado` calculations is that they are difficult to plot, at least simultaneously. All start at $+\infty$ at $m = 0$, and `EBAY` can also be very large as m approaches p. And, again, minimum values are not comparable. However difficult they may be to produce, plots of these minus two log likelihoods are of interest because one needs to see how flat each is near its minimum.

A summary of the example

The most striking feature of our example is the extent to which the `rxridge` estimates mimic their expected values from `rxrrisk`. I assure the reader that this mimicry is typical rather than an artifact of the single set of simulated responses generated using `rxrsimu`. The overall signal-to-noise ratio here was $\beta'\beta/\sigma^2 = 2.5$ when the diagonal elements of $X'X$ were scaled to equal $(n-1) = 12$, and one might expect much less mimicry with much lower ratios. I suggest that skeptics simply try it for themselves.

Next, I ask you to reexamine the estimated traces of Figures 6 through 9. Isn't it remarkable how much incredibly detailed information is contained in these traces concerning the extent and effects of shrinkage on ill-conditioned estimates of regression coefficients?

Syntax

As promised above, this section presents the syntax diagrams for all of the ridge commands.

`rxrcrlq` *depvar varlist* [`if` *exp*] [`in` *range*] [, `nq(#)` `qmax(#)` `qmin(#)` `rescale(#)` `tol(#)`]

`rxridge` *depvar varlist* [`if` *exp*] [`in` *range*] [, `msteps(#)` `qshape(#)` `rescale(#)` `tol(#)`]

`rxrmaxl` *depvar varlist* [`if` *exp*] [`in` *range*] [, `msteps(#)` `qshape(#)` `omdmin(#)` `rescale(#)` `tol(#)`]

`matrix rxsigma = (#)`

```
matrix rxgamma = (#,...,#)

rxrrisk depvar varlist [ if exp ] [ in range ] [ , msteps(#) qshape(#)
        omdmin(#) rescale(#) tol(#) ]

rxrsimu depvar varlist [ if exp ] [ in range ] [ , msteps(#) qshape(#)
        rescale(#) start(#) tol(#) ]
```

`msteps(#)` specifies the number of steps per unit increase in m, the multicollinearity allowance parameter; the default value is 4. The total number of steps along the generalized shrinkage path from the least squares solution ($m = 0$) to all zero coefficients ($m = r$) will thus be $1 + (\texttt{msteps} \times r)$, where $r = \mathrm{rank}(X)$.

`nq(#)` specifies an integer number of q-shape values to evaluate between `qmin` and `qmax`, inclusive. The default value is 21, and `nq` cannot be reset to any integer value less than 9.

`omdmin(#)` is the strictly positive minimum value to be used for calculation of $(1 - \delta)$ shrinkage factors. The default is $\texttt{omdmin} = 10^{-13}$.

`qmax(#)` specifies the maximum q-shape to evaluate. The default value is $\texttt{qmax} = +5$, and `qmax` cannot be reset to any value less than $+2$.

`qmin(#)` specifies the minimum q-shape to evaluate. The default value is $\texttt{qmin} = -5$, and `qmin` cannot be reset to any value greater than -2.

`qshape(#)` controls the shape (or curvature) of the generalized shrinkage path through likelihood space; the default value is 0, which yields Hoerl–Kennard "ordinary" ridge regression. $\texttt{qshape} = 1$ yields uniform shrinkage, and all `qshape` values between $+5.0$ and -5.0 are allowed.

`rescale(#)` controls the scaling of the response variable and all p predictor variables in the *varlist*. The default value is $\texttt{rescale} = 1$ to scale all centered variables to have sample variance 1 (sample sum-of-squares equal to one fewer than the number of observations). To retain the original scaling of variables in the Stata `.dta` file, specify `rescale(0)`.

`start(#)` controls Stata's uniform random number seed value, and the `rxrsimu` default value is 12345. If you make repeated `rxrsimu` runs without changing this seed, you will get the same pseudo-random values each time. When you do change `start`, make it large, positive, and odd.

`tol(#)` specifies the search convergence criterion and defaults to 0.01.

Restrictions: Several restrictions apply to the ridge programs.

1. The regression models always contain an intercept (constant) term.
2. The number, p, of (nonconstant) predictor variables, X, in the *varlist* must be at least two.
3. If p is greater than 20, the programs will refuse to draw `trace` plots.
4. The dependent variable, y, must be nonconstant.
5. No missing values are allowed.
6. The matrices `rxgamma` and `rxsigma` must be set prior to calling `rxrrisk` and `rxrsimu`.

The ridge programs internally center all variables to have mean zero, thus the fitted (hyper)plane always passes through $\overline{y} = 0$ at $\overline{X} = 0$. The implied y-intercept at the original X origin can, of course, be determined implicitly as the coefficients for the p, nonconstant regressors change, but the y-intercept is not calculated by the ridge programs.

In addition to coding these ridge programs for Stata, I have programmed my maximum-likelihood ridge algorithms in SAS/IML, S-PLUS and GAUSS. Also, Bernhard Walter, of the Technische Universität München, has created splendidly interactive routines for XLisp-Stat. However, the most complete implementation of my algorithms is provided by my stand-alone systems for MS-DOS personal computers: `rxridge.exe`, `rxtraces.exe`, and `pathproj.exe` (Obenchain 1991 and Nash 1992). For example, `rxridge.exe` calculates inference intervals (classical, confidence and Bayes HPD) for shrunken coefficients and performs ridge residual analyses (outlying responses and/or high leverage regressor combinations). I distribute all of the above software systems as freeware.

References

Efron, B. and C. Morris. 1977. Comment on A simulation study of alternatives to ordinary least squares by Dempster, A. P., M. Schatzoff, and N. Wermuth. *Journal of the American Statistical Association* 72: 91–93.

Gibbons, D. G. 1981. A simulation study of some ridge estimators. *Journal of the American Statistical Association* 76: 131–139.

Goldstein M. and A. F. M. Smith. 1974. Ridge-type estimators for regression analysis. *Journal of the Royal Statistical Society B* 36: 284–291

Golub, G. H., M. Heath, and G. Wahba. 1979. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics* 21: 215–223.

Hald, A. 1952. *Statistical Theory and Engineering Applications* New York: John Wiley & Sons. (Portland Cement data, p. 647).

Hoerl, A. E. 1962. Applications of ridge analysis to regression problems. *Chemical Engineering Progress* 58: 54–59.

Hoerl, A. E. and R. W. Kennard. 1970a. Ridge regression: biased estimation for non-orthogonal problems. *Technometrics* 12: 55–67.

——. 1970b. Ridge regression: applications to non-orthogonal problems. *Technometrics* 12: 69-82. (errata: 723.)

Marquardt, D. W. 1970. Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics* 12: 591–612.

Nash, J. C. 1992. Statistical shareware: illustrations from regression techniques. *The American Statistician* 46: 312-F-318.

Obenchain, R. L. 1975. Ridge analysis following a preliminary test of the shrunken hypothesis. *Technometrics* 17: 431–441.

——. 1978. Good and optimal ridge estimators. *Annals of Statistics* 6: 1111–1121.

——. 1981. Maximum likelihood ridge regression and the shrinkage pattern alternatives. unpublished review, 74 pages. *I.M.S. Bulletin* 10: 37; Abstract 81t–23.

——. 1984. Maximum likelihood ridge displays. *Communications in Statistics* A-13: 227–240. (Proceedings of the Fordham Ridge Symposium, ed. H. D. Vinod.)

——. 1991. Ridge regression systems for MS-DOS personal computers. *The American Statistician* 45: 245–246.

Obenchain, R. L. and H. D. Vinod. 1974. Estimates of partial derivatives from ridge regression on ill-conditioned data. *NBER-NSF Seminar on Bayesian Inference in Econometrics*, Ann Arbor, Michigan.

Piegorsch, W. W. and G. Casella. 1989. The early use of matrix diagonal increments in statistical problems. *SIAM Review* 31: 428–434.

Shumway, R. H. 1982. Maximum likelihood estimation of the ridge parameter in linear regression. Technical Report, Division of Statistics, University of California at Davis.

Vinod, H. D. 1976. Application of new ridge methods to a study of Bell System scale economies. *Journal of the American Statistical Association* 71: 835–841.

STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:			
an	announcements	*ip*	instruction on programming
cc	communications & letters	*os*	operating system, hardware, & interprogram communication
dm	data management		
dt	data sets	*qs*	questions and suggestions
gr	graphics	*tt*	teaching
in	instruction	*zz*	not elsewhere classified
Statistical Categories:			
sbe	biostatistics & epidemiology	*srd*	robust methods & statistical diagnostics
sed	exploratory data analysis	*ssa*	survival analysis
sg	general statistics	*ssi*	simulation & random numbers
smv	multivariate analysis	*sss*	social science & psychometrics
snp	nonparametric methods	*sts*	time-series, econometrics
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified

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

International Stata Distributors

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

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

Company:	Howching
Address:	11th Fl. 356 Fu-Shin N. Road Taipei, Taiwan, R.O.C.
Phone:	+886-2-505-0525
Fax:	+886-2-503-1680
Countries served:	Taiwan

Company:	Metrika Consulting
Address:	Ruddammsvagen 21 11421 Stockholm Sweden
Phone:	+46-708-163128
Fax:	+46-8-6122383
Countries served:	Baltic States, Denmark, Finland, Iceland, Norway, Sweden

Company:	Oasis Systems BV
Address:	Lekstraat 4 3433 ZB Nieuwegein The Netherlands
Phone:	+31 30 6066336
Fax:	+31 30 6065844
Countries served:	The Netherlands

Company:	Ritme Informatique
Address:	34 boulevard Haussmann 75009 Paris, France
Phone:	+33 1 42 46 00 42
Fax:	+33 1 42 46 00 33
Countries served:	Belgium, France, Luxembourg, Switzerland

Company:	Timberlake Consultants
Address:	47 Hartfield Crescent West Wickham Kent BR4 9DW, U.K
Phone:	+44 181 462 0495
Fax:	+44 181 462 0493
Countries served:	Eire, Portugal, U.K.