STATA TECHNICAL BULLETIN

July 1995
STB-26

A publication to promote communication among Stata users

Contents of this issue — page

crc39 How to make older ado-files work correctly

Some readers of the STB have called our technical support line to tell us they have problems getting ado-files from early issues of the STB to work correctly. The problem arises with older ado-files that use elements of Stata that have changed since the publication of that program. StataCorp makes every effort to minimize the number of changes to existing features when new versions of Stata are released, but, inevitably, some features must be changed to accommodate requested improvements.

There is a way to make older ado-files function as intended. Stata's `version` command instructs Stata to behave as if it were the specified version of Stata, regardless of the version that is actually being used. For instance, the command

```
. version 3.0
```

tells Stata to interpret and execute subsequent commands exactly as Version 3.0 of Stata would. (Of course, this example works only if you are running Version 3.0 or later of Stata.) You should always include a `version` command as the first line of your own ado-files. This practice protects the programs from changes in later versions of Stata.

The `version` command was introduced in Version 3.0 of Stata. As a consequence, ado-files written under earlier versions of Stata (or published before STB-7) will not contain this command. If you have difficulty running one of these programs, simply add the command "`version 2.1`" to the beginning of the program, and it will execute normally.

crc40 Correcting for ties and zeros in sign and rank tests

The current Stata commands that perform sign and rank tests (`signtest`, `signrank`, and `ranksum`) do not correct for ties and zeros. Included in this issue's `crc` directory are new versions of `signtest`, `signrank`, and `ranksum` that compute the appropriate corrections to the variance when ties or zeros occur.

Assumptions for the sign test and Wilcoxon signed-rank test

Both the sign test and Wilcoxon signed-rank tests test the null hypothesis that the distribution of a random variable X has median zero. The sign test makes no additional assumptions, but the Wilcoxon signed-rank test makes the additional assumption that the distribution of X is symmetric. If $X = Y_1 - Y_2$, where Y_1 and Y_2 have the same distribution, then it follows that the distribution of X is symmetric about zero. Thus, the Wilcoxon signed-rank test is often described as a test of the hypothesis that two distributions are the same; i.e., $Y_1 \sim Y_2$.

The test statistic for the sign test is the number n_+ of observations greater than zero. Assuming that the probability of an observation being equal to zero is exactly zero, then, under the null hypothesis, $n_+ \sim \text{Binomial}(n, p = \frac{1}{2})$, where n is the total number of observations. But what do we do if we have some observations that are zero?

Fisher's Principle of Randomization

We have a ready answer to this question if we view the test from the perspective of Fisher's Principle of Randomization (Fisher 1935). Fisher's idea (stated in a modern way) was to look at a family of transformations of the observed data such that the a priori likelihood (under the null hypothesis) of the transformed data is the same as the likelihood of the observed data. The distribution of the test statistic is then produced by calculating its value for each of the transformed "randomization" data sets, considering each data set equally likely.

For the sign test, the "data" are simply the set of signs of the observations. Under the null hypothesis of the sign test, $P(X_i > 0) = P(X_i < 0)$, so we can transform the observed signs by flipping any number of them and the set of signs will have the same likelihood. The 2^n possible sign changes form the family of randomization data sets. If we have no zeros, this procedure again leads to $n_+ \sim \text{Binomial}(n, p = \frac{1}{2})$.

If we do have zeros, changing their signs leaves them as zeros. So if we observe n_0 zeros, each of the 2^n sign-change data sets will also have n_0 zeros. Hence, the values of n_+ calculated over the sign-change data sets range from 0 to $n - n_0$, and the "randomization" distribution of n_+ is $\text{Binomial}(n - n_0, p = \frac{1}{2})$.

Example

```
. signtest mpg = 22

Sign test

    sign |    observed    expected
---------+------------------------
positive |          26        34.5
negative |          43        34.5
    zero |           5           5
---------+------------------------
     all |          74          74

One-sided tests:
Ho: median of mpg = 22 vs. Ha: median of mpg > 22
    Pr(#positive >= 26)
    = Binomial(n = 69, x >= 26, p = 0.5) =  0.9853

Ho: median of mpg = 22 vs. Ha: median of mpg < 22
    Pr(#negative >= 43)
    = Binomial(n = 69, x >= 43, p = 0.5) =  0.0266

Two-sided test:
Ho: median of mpg = 22 vs. Ha: median of mpg ~= 22
    Pr(#positive >= 43 or #negative >= 43)
    = min(1, 2*Binomial(n = 69, x >= 43, p = 0.5)) =  0.0533
```

Handling zeros and ties in the Wilcoxon signed-rank test

Fisher's Principle of Randomization can also be used to deal with zeros and ties in the Wilcoxon signed-rank test. Here the data consist of the signed ranks; i.e., we first calculate $r_i = \mathrm{sign}(x_i)\,\mathrm{rank}(|x_i|)$ for the each of the original observations. Given our assumption of symmetry, we have $f(x_i) = f(-x_i)$, where f is the distribution of X. Hence, the likelihood is unchanged if we flip signs on the x_i, and thus the family of randomization data sets is the 2^n possible sign changes for the r_i.

When we observe $x_j = 0$, r_j will always be zero in each of the randomization data sets (using $\mathrm{sign}(0) = 0$). When we have ties, we can assign averaged ranks for each group of ties and then treat them the same as the other ranks. The randomization distribution can be expressed as follows: If the observed test statistic is $t = \sum r_i$, the distribution of t is $T = \sum r_i S_i$, where the r_i are the observed signed-ranks (considered fixed) and the S_i are independent random variables with $P(S_i = 1) = \frac{1}{2}$ and $P(S_i = -1) = \frac{1}{2}$. It is easy to see that $E(T) = 0$ and $\mathrm{Var}(T) = \frac{1}{4}\sum r_i^2$. The test statistic for the Wilcoxon signed-rank test is usually expressed as the sum of the positive signed-ranks, but this just differs from T by a constant.

Example

```
. signrank x1 = x2

Wilcoxon signed-rank test

    sign |      obs   sum ranks    expected
---------+---------------------------------
positive |       33        1256      1382.5
negative |       37        1509      1382.5
    zero |        4          10          10
---------+---------------------------------
     all |       74        2775        2775

unadjusted variance    34456.25
adjustment for ties      -58.12
adjustment for zeros      -7.50
                     ----------
adjusted variance      34390.62

Ho: median of x1 = x2
             z =  -0.682
    Prob > |z| =   0.4952
```

The "unadjusted variance" is the variance that the randomization distribution would have had if there had been no ties or zeros; i.e., $\mathrm{Var}_{\mathrm{unadj}}(T) = \frac{1}{4}\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/24$. The adjustment for ties is the change in the variance when the ranks (for nonzero observations) are replaced by averaged ranks. The adjustment for zeros is the change in the variance when the ranks for the zeros are signed to make $r_j = 0$; i.e., the variance is reduced by $\frac{1}{4}\sum_{i=1}^{n_0} i^2 = n_0(n_0+1)(2n_0+1)/24$.

Handling ties in the Wilcoxon rank-sum test (Mann–Whitney test)

For the Wilcoxon rank-sum test, we have two independent random variables X_1 and X_2, and we test the null hypothesis that $X_1 \sim X_2$. We have a sample of size n_1 from X_1 and another of size n_2 from X_2. The data are then ranked without regard to the sample to which they belong. Wilcoxon's test statistic (Wilcoxon 1945) is the sum of the ranks for the observations in the first sample: $T = \sum R_{1i}$. Mann and Whitney's U statistic (Mann and Whitney 1947) is the number of pairs (X_{1i}, X_{2j}) such that $X_{1i} > X_{2j}$. These statistics differ only by a constant: $U = T - n_1(n_1 + 1)/2$.

Again Fisher's Principle of Randomization provides a method for calculating the distribution of the test statistic, ties or not. The randomization distribution consists of the $\binom{n}{n_1}$ ways to choose n_1 ranks from the set of all $n = n_1 + n_2$ ranks and assign them to the first sample. It is a straightforward exercise to verify that $E(T) = n_1(n+1)/2$ and $\mathrm{Var}(T) = n_1 n_2 s^2/n$, where s is the standard deviation of the combined ranks for both groups. This formula for the variance is exact and holds when there are no ties and when there are ties and we use averaged ranks. (Indeed, the variance formula holds for the randomization distribution of choosing n_1 numbers from any set of n numbers.)

Example

```
. ranksum mpg, by(foreign)
Two-sample Wilcoxon rank-sum (Mann-Whitney) test
 foreign |      obs    rank sum    expected
---------+--------------------------------
       0 |       52      1688.5        1950
       1 |       22      1086.5         825
---------+--------------------------------
combined |       74        2775        2775
unadjusted variance     7150.00
adjustment for ties      -36.95
                     ----------
adjusted variance       7113.05
Ho: median mpg(foreign==0) = median mpg(foreign==1)
             z =  -3.101
    Prob > |z| =   0.0019
```

Other methods of handling zeros and ties

Other proposed methods of handling zeros and ties can sometimes lead to anomalies (Pratt and Gibbons 1981). The only other procedure that can be recommended is to decide a priori to break all zeros and ties in such a way as to make the p-value as large as possible. This is, of course, a very conservative procedure and can lead to an appreciable loss in power if the probability of observing zeros and ties is not small.

References

Fisher, R. A. 1935. *Design of Experiments.* Edinburgh: Oliver and Boyd.

Mann, H. B. and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics* 18: 50–60.

Pratt, J. W. and J. D. Gibbons. 1981. *Concepts of Nonparametric Theory.* New York: Springer-Verlag.

Wilcoxon, F. 1945. Individual comparisons by ranking methods. *Biometrics* 1: 80–83.

dm31 Counting missing values: an extension to egen

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

The `rmiss()` function available with Stata's `egen` command counts the number of missing values in a variable list. However, `rmiss()` will not accept string variables in the variable list. I have written a generalization to `rmiss()`, called `rmiss2()`, that follows the same syntax and offers the same functionality as `rmiss()` except that `rmiss2()` handles both numeric and string variables. The following example illustrates the use of `rmiss2()`.

```
. use example
(1978 Automobile Data)
```

```
. list
                 make     price       mpg
  1.                       4099        22
  2.        AMC Pacer      4749        17
  3.       AMC Spirit         .        22
  4.    Buick Century      4816        20
  5.                       7827         .

. egen nmiss = rmiss(price mpg)

. list
                 make     price       mpg      nmiss
  1.                       4099        22          0
  2.        AMC Pacer      4749        17          0
  3.       AMC Spirit         .        22          1
  4.    Buick Century      4816        20          0
  5.                       7827         .          1

. drop nmiss

. egen nmiss = rmiss(make price mpg)
type mismatch
r(109);

. egen nmiss = rmiss2(make price mpg)

. list
                 make     price       mpg      nmiss
  1.                       4099        22          1
  2.        AMC Pacer      4749        17          0
  3.       AMC Spirit         .        22          1
  4.    Buick Century      4816        20          0
  5.                       7827         .          2
```

dm32 Matching names in Stata

Peter Sasieni, Imperial Cancer Research Fund, London, FAX (011)-44-171-269-3429

I was recently faced with the problem of trying to identify individuals who appeared more than once in a data base of approximately 8000 records collected over a nine year period. Although sex and age (in years) might be used to partially eliminate false matches, it was only possible to identify matches by using the names. These had been entered as two strings: **name1** (surname) and **name2** (forename). Exact matches were easy to obtain. The following code provides a separate **id** to each unique name. **id** is missing for names that occur only once.

```
. generate str30 name = name1 + name2

. sort name

. generate long id = (name != name[_n-1])

. replace id = sum(id)

. sort id

. quietly by id: replace id=id[2]
```

The problem with this approach is that it does not permit any variations in spelling of names. We cannot easily identify "BILL" with "WILLIAM" without using a sophisticated program, but it ought to be possible to match "ANN" with "ANNE". Additionally we would like to check for the inversion of surname and forename. The following code tackles the name inversion and partially overcomes the problem of slight variants in one or other name.

```
. generate byte stack=1

. global nobs=_N

. global nfrom=$nobs+1

. expand 2

. replace name1 = name2 in $nfrom/l

. replace name2 = name1[_n-$nobs] in $nfrom/l

. replace stack=2 in $nfrom/l

. sort name1 name2

. generate byte m1=(name2==name2[_n-1] | name2==name2[_n+1])
```

`m1` identifies individuals that are near matches. It is guaranteed to match "SASIENI PETER" with "PETER SASIENI". It has a good chance of matching "SASIENI PETER" with "SASIENIE PETER". This is because `m1` only looks for an exact match of **name2** and, provided there are not lots of different people with the surname "SASIENI", these two records are likely to be adjacent in the sorted data base. Thus "SASIENI PETER" and "SOSIENI PETER" are less likely to be matched since they will be separated by anyone with the name "SMITH". Likewise "SMITH YVONNE" and "SMITH YVETTE" will probably match (when sorted by the forename — stack==2), but "SASIENI JOHN" and "SASIENI JONATHAN" are unlikely to match. (The latter fails because there is not an exact match and because "JOHN" and "JONATHAN are unlikely to be consecutive forenames in a large, alphabetically sorted, data base.)

One way of trying to match such names is to relax the matching criterion, using

```
. generate str2 n2a=substr(name2,1,2)
. generate byte m2=(n2a==n2a[_n-1] | n2a==n2a[_n+1])
```

Another is to create a new variable and to sort the data using it. For example

```
. generate str7 namex=substr(name1,1,3)+substr(name1,-3,3)+substr(name2,1,1)
. sort namex
. generate byte m3=(namex==namex[_n-1] | namex==namex[_n+1])
```

We found the above tricks quite useful for identifying additional matches without creating too many false matches. The difficulty in obtaining a substantially better solution to the problem of matching names in Stata stems from the size of the problem. One could imagine computing a score that would describe the distance between any two names, but since one wishes to compare every record with every other record, there would be _N*(_N-1)/2 (i.e., approximately 32 million when _N=8000) such scores to compute. It would not be difficult to store only the 5 closest names and their scores, but the problem of computing the scores is probably beyond the capabilities of Stata.

A related STB entry by William Gould (*dm13*) is concerned with separating a list of names into prefixes (such as Mr. and Ms.), first names, middle initials, last names, suffixes (such as Jr.) and affiliations (such as BSc. or Esq.). Although there is no overlap between nmatch, the program presented here, and Gould's extrname program, it will be useful to run extrname before attempting to match names whenever the names have been entered in a variety of formats.

A program to match names

We include with this insert an ado-file, nmatch, which uses minlen and replstr from *dm13.1*. This program probably should be adapted to the particular problem at hand. The enclosed version assumes that the names have been entered in uppercase as two string variables. The syntax is

nmatch *namevar1 namevar2*

nmatch produces five additional variables: _m1 identifies observations that share the same values for both *namevar1* and *namevar2* with another observation. Observations share a common integer value of _m1 if and only if they have the same names in *namevar1* and *namevar2*. Observations with unique names have _m1 set to missing.

Similarly _m2 and _m3 are integer-valued variables for observations with identical values in either *namevar1* or *namevar2* and lexicographically adjacent, but non-identical, values in the other variable. Observations with a nonmissing value for _m1 will only be nonmissing for _m2 or _m3 if the relaxed matching criteria identify an additional observation that matches the original, exactly matching names. For example "WILLIAM GOULD", "WILLIAM GOULD" and "WILL GOULD".

_m4 makes use of replstr to search for names that match apart from certain spelling variations. nmatch makes the following substitutions (in the order given) to strings in *namevar1* and *namevar2* before searching for a new match:

Y	→	I
IE	→	I
EI	→	I
OU	→	O
Z	→	S
LL	→	L
NN	→	N
TT	→	T
FF	→	F
PH	→	V
HN	→	N
MAC	→	MC

_m5 looks for matches by interchanging the value of *namevar1* with that of *namevar2*. Only exact matches are counted.

Example

When applied to our data set of 7873 observations, the result was

```
. describe N*
  1. Name1        str15  %15s              surname
  2. Name2        str10  %10s              forename
. nmatch Name1 Name2
. describe _*
  3. _m1          int    %8.0g             Exact match
  4. _m2          int    %8.0g             forename match
  5. _m3          int    %8.0g             surname match
  6. _m4          int    %8.0g             Sound match
  7. _m5          int    %8.0g             Reverse match
. summarize _m*
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
     _m1 |     794    3689.597   2205.015          8       7439
     _m2 |      42    3513.095   2409.536        266       6737
     _m3 |      10      5214.4   1737.218       2602       6975
     _m4 |      19    3730.579     2551.6        464       7321
     _m5 |       2        7654          0       7654       7654
```

Of the 42 observations with nonmissing `_m2` (same first name), all but 7 were subsequently eliminated. Of the 10 identified by `_m3` (same surname), 8 were confirmed, whereas only 2 of the 19 new matches identified by `_m4` had ages that were compatible. The pair identified by `_m5` were observations on the same individual.

I regret that the data set used in this example cannot be made available due to data protection legislation. To help readers gain familiarity with `nmatch`, I have constructed an artificial example using the automobile data supplied with Stata.

I split the `make` variable into two new variables, `Make` and `Model`, and stored just these two variables in an example data set that is provided on the distribution diskette.

```
. use example
(1978 Automobile Data)
. list in f/5
          Make       Model
  1.       AMC     Concord
  2.       AMC       Pacer
  3.       AMC      Spirit
  4.     Buick     Century
  5.     Buick     Electra
```

Applying `nmatch` to these data produces

```
. nmatch Make Model
. summarize
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
    Make |       0
   Model |       0
     _m1 |       0
     _m2 |       0
     _m3 |      14    37.14286   21.31398          2         67
     _m4 |       0
     _m5 |       0
```

There are no exact matches (`_m1` is always missing) because this data set contains exactly one record for each type of automobile. There are fourteen cases where the value of `Make` is identical and the values of `Model` are lexicographically adjacent.

```
. list _m3 Make Model if _m3!=.
             _m3       Make        Model
  8.          54      Buick        Regal
  9.          54      Buick      Riviera
 17.          46      Chev.  Monte Carlo
 18.          46      Chev.        Monza
 31.          45      Merc.      Marquis
 32.          45      Merc.      Monarch
 33.          67      Merc.         XR-7
 34.          67      Merc.       Zephyr
 36.          24       Olds    Cutl Supr
 37.          24       Olds      Cutlass
 56.           2     Datsun          200
 57.           2     Datsun          210
 68.          22     Toyota      Corolla
 69.          22     Toyota       Corona
```

The treatment of the four Mercury models is instructive.

I invite the interested reader to modify this data set and to use it to experiment with `nmatch`.

Reference

Gould, W. 1993a. dm13: Person name extraction. *Stata Technical Bulletin* 13: 6–11.

——. 1993b. dm13.1: String manipulation functions. *Stata Technical Bulletin* 13: 11–13.

dm33 Elapsed days using 30-day months

Ken Heinecke, Century Investment Management Corporation, EMAIL heinecke@itis.com

Financial analysts often find it necessary to calculate cash flows based on a 360-day year. I have written `days360` to calculate the number of days between two given dates based on twelve 30-day months. The syntax of the command is

`days360, begdate(`*date1*`) enddate(`*date2*`)`

where *date1* and *date2* must be entered as *mm/dd/ccyy* or *mm-dd-ccyy*. Note that the centuries are required and that both `begdate()` and `enddate()` must be specified for the program to run. The program calculates the number of elapsed days and saves it in the global macro `days`.

Methodology

I have followed the NASD method for calculating 30-day months. If the beginning date is the 31st of a month, it becomes equal to the 30th of the same month. If the ending date is the 31st of a month and the beginning date is less than the 30th of a month, the ending date is considered to be equal to the 1st of the next month. If the ending date is the 31st and the beginning date is the 30th or 31st, the ending date becomes equal to the 30th day of the same month.

Example

```
. days360, b(3/31/1995) e(2/28/1996)

. macro list days
days:      328

. days360, b(5-15-1990) e(7-1-1992)

. macro list days
days:      766
```

dm34 Constructing axis labels for dates

Sean Becketti, Editor, Stata Technical Bulletin

This insert presents `yrxlab`, a simple utility designed for a single, narrow purpose: constructing attractive axis labels for date variables. I hesitate to publish such a specialized function, however, I use `yrxlab` so frequently and it solves such a nagging problem, that I believe it may be useful to a large number of STB users.

Background

Stata provides several ways of generating axis labels for date variables. When the date variable is stored as a Stata elapsed date with a date format, the `datelab` program can be used (Riley 1995). While date formats are useful, they are a relatively recent addition to Stata. As a consequence, over the years many time series users (including myself) developed other encodings for date variables. For instance, I routinely create a variable that is equal to the last two digits of the year plus the fraction of the year elapsed. Thus, with monthly data, I would type

```
. generate date = (year-1900) + (month-1)/12
```

This variable works nicely as the x-variable in time series graphs.

Even with this natural encoding of the date, it can be difficult to produce an attractive `xlabel()`. In an interactive session, I generally start by typing, say,

```
. graph y date, ylabel xlabel
```

to see what Stata will produce. If I am unhappy with the results, I will explicitly type the desired values in the `xlabel()` option.

This approach will not work when the graph is produced inside a program or do-file. In this situation, I have sometimes hard-coded an `xlabel()` (when the date coverage of the date was predictable and constant) or used `nicenum` to guess at a likely set of values to label (Hardin 1995).

Neither of these approaches completely solves the problem. The difficulty is that the implicit rules for selecting an attractive set of values to label is different when the values are dates. In particular, it is frequently helpful to the reader to label each year even if doing so “crowds” the x-axis a bit. `yrxlab` makes it easy to construct this type of `xlabel()` inside a Stata program.

Constructing a “year” label

`yrxlab` constructs a valid `xlabel` specification for the period covered by a set of variables and stores it in `S_1`. The syntax of `yrxlab` is

`yrxlab` *varlist* [`if` *exp*] [`in` *range*] [`,` `step(`*#*`)` `year(`*year-variable*`)`]

`yrxlab` assumes that some measure of time involving years will be the x-variable in an upcoming graph. `yrxlab` determines the minimum and maximum values of the *year-variable* for which there are nonmissing values in any of the variables in the *varlist*. The *year-variable* actually can be any variable that will be displayed along the x-axis. If no *year-variable* is specified, the program assumes there is a variable in the current data set named `year`. (This default is tailored to my working habits.) Then `yrlab` constructs an `xlabel` option that labels the x-axis at every `step`-th value between the minimum and maximum values. The default `step()` is one.

Example

A simple example will illustrate the use of `yrxlab`. This example uses several programs from the Stata time series library (Becketti 1995), but none of them are needed to use `yrxlab`. They are included only to make the example more realistic.

Suppose a colleague claims that the monthly average of the daily 1-year constant maturity Treasury (CMT) rate is well-modeled by a first-order autoregressive process. You are convinced that this claim is mistaken, and you decide to produce a graph that provides evidence on this question. First you estimate a first-order autoregression for the 1-year CMT. You use a data set that contains monthly data for the 1990s, set the period and date variable markers to let the time series programs know you are working with monthly data, and estimate the model.

```
. use example
(1-year Treasury rates)

. describe

Contains data from example.dta
  Obs:    66 (max= 30478)                       1-year Treasury rates
 Vars:     3 (max=    99)                       10 Jul 1995 17:22
Width:     8 (max=   200)
  1. year         int    %8.0g                  Year
  2. month        int    %8.0g       month      Month
  3. cmt1         float  %9.3f                  1-year CMT yield
Sorted by:  year  month
```

```
. period 12
12 (monthly)

. datevars year month

. tsreg cmt1, lag(1) nomult replace
Monthly data:  February, 1990 to June, 1995    (65 obs)

  Source |       SS       df       MS                  Number of obs =      65
---------+------------------------------               F(  1,    63) = 2395.00
   Model |  167.770367     1  167.770367               Prob > F      =  0.0000
Residual |  4.41316457    63  .070050231               R-squared     =  0.9744
---------+------------------------------               Adj R-squared =  0.9740
   Total |  172.183531    64  2.69036767               Root MSE      =  .26467

------------------------------------------------------------------------------
    cmt1 |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  L.cmt1 |   .9689501   .0197992     48.939   0.000       .9293845    1.008516
   _cons |    .131867   .1112399      1.185   0.240      -.0904283    .3541623
------------------------------------------------------------------------------
              AIC: -2.628
 Schwarz criterion: -2.561
Durbin-Watson test: .916
  seasonal DW test: 1.634
        Q(12) test: 0
       LM(12) test: .03
     ARCH(12) test:  0.15
    normality test: .45
```

Already you detect problems with the model. The Q and LM tests indicate serially correlated residuals, as does the Durbin–Watson test.

Now you increase the size of the current data set to produce forecasts through the end of 1998. You use `filldate` to fill in the missing values of the date variables. Then you use `tspred` to calculate dynamic forecasts from the model. You suspect your colleague's enthusiasm for the first-order autoregressive model may be based on a naive comparison of historical values with static, within-sample forecasts, so you use `predict` to calculate the static forecasts as well.

```
. set obs 108
obs was 66, now 108

. filldate, beg(1995 7)

. tspred forecast

. predict fit
(43 missing values generated)
```

Now you encode a date variable, using the technique described above, and graph the historical values along with the two predictions.

```
. generate date = (year-1900) + (month-1)/12

. graph cmt fit forecast date, c(lll) s(o..) ylabel xlabel
```
(graph appears, see Figure 1)

The inadequacy of the model and the reason for your colleague's enthusiasm are both evident in this graph. However, Stata's choices of "nice" values for the dates don't correspond to the choices you want. You use `yrxlab` to construct a better set.

```
. generate yr = year-1900

. yrxlab cmt forecast, year(yr)

. global xlab "$S_1"

. display "$S_1"
xlab(90,91,92,93,94,95,96,97,98)

. graph cmt fit forecast date, c(lll) s(o..) ylabel $xlab
```
(graph appears, see Figure 2)

Remarks

`yrxlab` is most useful in Stata programs and ado-files, where it is necessary to handle date ranges without knowing in advance how many years will be covered. To prevent "overcrowding" along the x-axis, your programs can set the `step()` value to control the number of values that are labeled. There are (at least) two ways to approach this problem. First, you can take an initial peek at the minimum and maximum date values as a guide to setting the step size. Alternatively, you can leave the step size at its default of one and test the size of the string returned in `S_1` by `yrxlab`. If the string is too long, change the step size to two and try again. If it's still too long, change the step size to five and try again, and so on.

As it stands **yrxlab** is a narrowly specialized program, but its basic function has wider applicability and the program easily can be cannibalized. First, **yrxlab** can construct an **xlabel** for any variable, not just for a date. The **year()** option allows you to specify any variable for the x-axis variable. There is one potential pitfall. **yrxlab** blithely assumes that the minimum and maximum values of the **year()** variable will be nice numbers, like integers. When the x-variable is a floating point variable, this assumption can produce unsightly labels. Logic could be added to **yrxlab** to handle this problem.

Second, **yrxlab** addresses a long-standing problem of axis labels: finding a convenient way to specify stepped values. I have long yearned for a simple way to tell Stata to label every kth value from $x_{\min}$ to $x_{\max}$. **yrxlab** is a partial solution for that problem, at least for the **xlabel**.

Third, **yrxlab** contains a subroutine that, with a little modification, can provide a more general solution to the problem of stepped values in a wide variety of settings. This subroutine is called **steplist** and its syntax is

steplist , **f**rom(*#*) **t**o(*#*) [**s**pace(*str*) **st**ep(*#*)]

bigskip**steplist** stores in **S_1** the values from **from()** to **to()** in increments of size **step()**. The default step size is 1. (If **to()** is less than **from()**, **step()** must be negative.) By default, the values are separated by single spaces, but an alternative separator can be specified with the **space()** option.

yrxlab calls **steplist** to construct the list of values, then wraps it up as an **xlabel**, as follows:

```
steplist, from(`min´) to(`max´) space(",") step(`step´)
global S_1 "xlab($S_1)"
```

Clearly, **steplist** can be used in many other settings.

References

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

Hardin, J. 1995. dm28: Calculate nice numbers for labeling or drawing grid lines. *Stata Technical Bulletin* 25: 2–3.

Riley, A. 1995. dm26: Labeling graphs with date formats. *Stata Technical Bulletin* 24: 4–5.

Figures

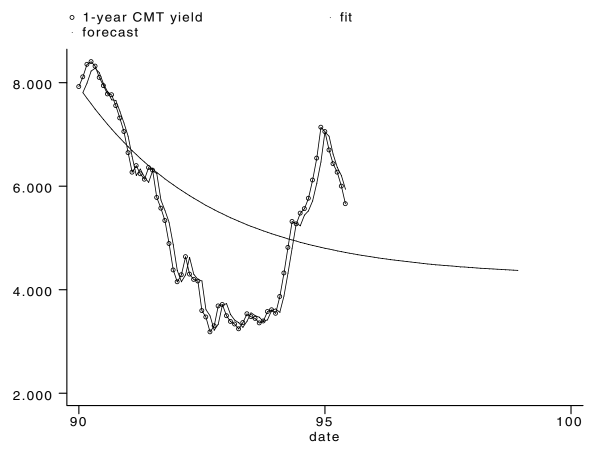

Figure 1

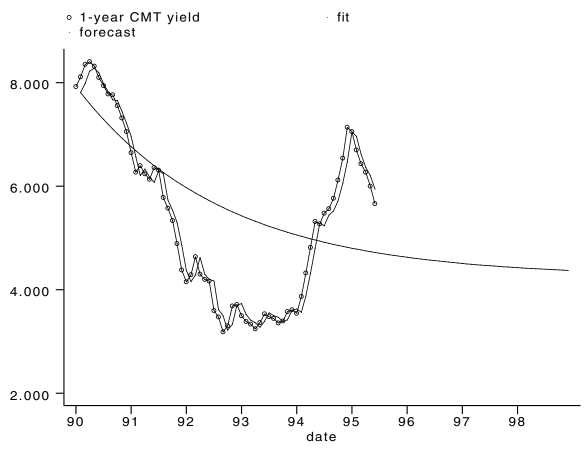

Figure 2

ip8 An enhanced `for` command

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

Stata's `for` command is extremely useful, but it has a limitation that I seem to trip over frequently. `for` replaces the `@` token with whatever the user has typed. Unfortunately, `for` also inserts a space in front of the `@` when doing the substitution. For example, the command

```
. for 10 20, any : generate mpg@ = mpg*@
```

which one might hope would generate two new variables, `mpg10` and `mpg20`, instead fails with an error message saying that `mpg` already exists. The reason is that `for` has expanded the command to

```
. generate mpg 10 = mpg* 10
. generate mpg 20 = mpg* 20
```

which have syntax errors. This insert presents `for2` which remedies the problem.

In addition, `for2` has a new option, `numeric`, that provides list expansion for numeric *forlists*. For example, `1-5` is expanded to `1 2 3 4 5`, whereas `1-5/2` becomes `1 3 5`.

The syntax of `for2` is

`for2` *list* [`, any noheader nostop numeric`] `:` *stata_cmd*

Apart from `numeric`, which is illustrated below, the options are the same as for the standard `for` command.

Example

```
. for2 3-9/3 15, numeric: generate group@ = group+@
```

This command creates four new variables called `group3`, `group6`, `group9` and `group15`, equal to `group+3`, `group+6`, `group+9` and `group+15` respectively.

sg35.1 Robust tests for the equality of variances: correction

Mario A. Cleves, Arkansas Foundation for Medical Care, FAX 501-785-3460

Several astute readers of the STB noticed errors in `robvar.ado`, a program I wrote to calculate robust tests for the equality of variances. There were two problems. First, due to a typo, the reported statistics were incorrect. Second, the syntax diagram for `robvar` mistakenly indicated that `if` and `in` clauses are allowed in `robvar`.

I have corrected the typo and retested `robvar`. I am convinced that `robvar` now reports accurate results. However, `robvar` does not handle `if` and `in`. I leave that extension to a later insert. The corrected program appears on the STB-26 distribution diskette.

I wish to thank the alert readers who first noticed these problems, and I encourage any readers with questions or concerns to contact me at the FAX number listed above.

Reference

Cleves, M. 1995. sg35: Robust tests for the equality of variances. *Stata Technical Bulletin* 25: 13–15.

sg37.1 Orthogonal polynomials: correction

William M. Sribney, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

The `orthpoly` program described in *sg37* for computing orthogonal polynomials gave a syntax error when used with `if`, `in`, or weights. A corrected version is supplied on the STB-26 media.

Reference

Sribney, W. M. 1995. sg37: Orthogonal polynomials. *Stata Technical Bulletin* 25: 17–18.

sg40 Testing for the mean of a skewed variable

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

While the standard one-sample t test is relatively robust, it can have problems if the data are sufficiently non-normal. If the problem is tails that are too heavy, the Wilcoxon test ([5s] signrank) can be used. However, if the data are skewed (even as much as to a lognormal distribution), then both the t test and the Wilcoxon sign-rank test can have problems (as can the sign test). In this case, there is evidence that one-sided tests can be badly off (two-sided tests are not as badly off).

This insert presents `johnson`, an ado-file that implements Johnson's corrected t test for skewed data. Johnson's test loses little power, compared to the t test, when the distribution of the data is normal (Kleijnen et al. 1986). The test is sensitive to the degree of skewness and the sample size. As Sutton (1993) recently showed, the test is slightly conservative when the skew is greater than 1.5 and noticeably conservative when the skew is 2.9 or greater. Sample sizes of less than 20 for skew of about 1.5 or less than 80 for skew of about 3 also can affect the test. In these situations, Sutton recommends a bootstrap correction to Johnson's test that appears to improve the results.

Chen (1995) has proposed a modified, upper-tailed, Johnson test that is better, in terms both of power and of the accuracy of the p-value, than either Johnson's test or Sutton's bootstrap correction when the sample size is less than 20. The difference is not large, however, unless the asymmetry is marked and/or the sample size is less than 10.

The syntax for this test is

`johnson` *varname* `=` { *#* | *varname2* } [`if` *exp*] [`in` *range*]

`johnson` displays the results of the classical t test (as reported by Stata's `ttest` command), of Johnson's modified t test (denoted by `t1`), and of Chen's modification of Johnson's test (denoted by `t2`).

Example

We use the automobile data set supplied with Stata to illustrate `johnson`. We wish to test whether the mean of `mpg` is equal to 20. This variable exhibits a fair amount of skewness.

```
. use \stata\auto
(1978 Automobile Data)
. summarize mpg, detail
                       Mileage (mpg)
-------------------------------------------------------------
      Percentiles      Smallest
 1%           12             12
 5%           14             12
10%           14             14       Obs                  74
25%           18             14       Sum of Wgt.          74
50%           20                      Mean            21.2973
                        Largest       Std. Dev.      5.785503
75%           25             34
90%           29             35       Variance       33.47205
95%           34             35       Skewness       .9487176
99%           41             41       Kurtosis       3.975005
```

We use `signrank` to calculate Wilcoxon's test and `johnson` to calculate, first, the classical t test and, second, Johnson's modified t test.

```
. signrank mpg = 20
Test:  Equality of distributions (Wilcoxon Signed-Ranks)
Result of mpg - (20)
 Sum of Positive Ranks = 1610.5
 Sum of Negative Ranks = 1164.5
 z-statistic   1.20
 Prob > |z|    0.2296
```

```
. johnson mpg = 20
  Variable |      Obs        Mean   Std. Dev.
  ---------+--------------------------------
       mpg |       74     21.2973    5.785503
           Ho:  mean = 20
                   t = 1.93 with 73 d.f.
            Pr > |t| = 0.0576
           Ho:  mean = 20
                  t1 = 1.96 with 73 d.f.
            Pr > |t| = 0.0542
           Ho:  mean = 20
                  t2 = 1.96 with 73 d.f.
            Pr > |t| = 0.0542
```

In the next example, we use new data on mileage, taken from the second example in the description of the `ttest` command ([5s] ttest). We conduct a paired-samples test on these data.

```
. use mpg, clear
(Stata manual, 3: 249)
. ttest mpg1 = mpg2
  Variable |      Obs        Mean   Std. Dev.
  ---------+--------------------------------
      mpg1 |       12          21    2.730301
      mpg2 |       12       22.75    3.250874
  ---------+--------------------------------
     diff. |       12       -1.75     2.70101
           Ho:  diff = 0  (paired data)
                   t = -2.24 with 11 d.f.
            Pr > |t| = 0.0463
. johnson mpg1 = mpg2
  Variable |      Obs        Mean   Std. Dev.
  ---------+--------------------------------
  __000009 |       12       -1.75     2.70101
           Ho:  mean = 0
                   t = -2.24 with 11 d.f.
            Pr > |t| = 0.0463
           Ho:  mean = 0
                  t1 = -2.06 with 11 d.f.
            Pr > |t| = 0.0635
           Ho:  mean = 0
                  t2 = -2.06 with 11 d.f.
            Pr > |t| = 0.0636
```

Instead of testing the equality of the means of the two variables, it is equivalent to test whether the mean difference of the variables is zero.

```
. generate diff = mpg1-mpg2
. ttest diff = 0
  Variable |      Obs        Mean   Std. Dev.
  ---------+--------------------------------
      diff |       12       -1.75     2.70101
           Ho:  mean = 0
                   t = -2.24 with 11 d.f.
            Pr > |t| = 0.0463
. johnson diff = 0
  Variable |      Obs        Mean   Std. Dev.
  ---------+--------------------------------
      diff |       12       -1.75     2.70101
           Ho:  mean = 0
                   t = -2.24 with 11 d.f.
            Pr > |t| = 0.0463
           Ho:  mean = 0
                  t1 = -2.06 with 11 d.f.
            Pr > |t| = 0.0635
           Ho:  mean = 0
                  t2 = -2.06 with 11 d.f.
            Pr > |t| = 0.0636
```

References

Chen, L. 1995. Testing the mean of skewed distributions. *Journal of the American Statistical Association* 90: 767–772.

Johnson, N. J. 1978. Modified t tests and confidence intervals for asymmetrical populations. *Journal of the American Statistical Association* 73: 536–544.

Kleijnen, J. P. C., G. L. J. Kloppenburg, and F. L. Meeuwsen. 1986. Testing the mean of an asymmetric population: Johnson's modified t test revised. *Communications in Statistics, Part B—Simulation and Computation* 15: 715–732.

Sutton, C. D. 1993. Computer-intensive methods for tests about the mean of an asymmetrical distribution. *Journal of the American Statistical Association* 88: 802–810.

sg41	Random-effects probit

William M. Sribney, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

`rfprobit` estimates a random-effects model for cross-sectional time-series probit. The syntax of `rfprobit` is

```
rfprobit depvar [indepvars] [if exp] [in range] [, i(varname) quadrat(#) nochisq
                                                   nolog level(#) maximize_options ]
```

Options

`i(`*varname*`)` specifies the variable corresponding to an independent unit (e.g., a subject id). This variable represents the i in $\mathbf{x}_{it}$. Either this option must be specified or i must be set using the `iis` command; see [5s] xt in the *Reference Manual.*

`quadrat(#)` specifies the number M of points to use for Gaussian–Hermite quadrature. The default is 6. Increasing its value improves accuracy slightly but also increases computation time (see the following discussion).

`nochisq` omits the estimation of the constant-only model. This will reduce computation time at the cost of not being able to calculate the model χ^2 or pseudo R^2.

`nolog` suppresses the display of the likelihood iterations.

`level(#)` specifies the significance level, in percent, for confidence intervals of the coefficients; see [4] estimate.

maximize_options control the maximization process; see [7] maximize. Use the `trace` option to view parameter convergence. The `ltol(#)` option can be used to loosen the convergence criterion (default is `1e-6`) during specification searches.

Maximum-likelihood estimation procedure

`rfprobit` uses the maximum-likelihood estimation procedure outlined in Butler and Moffitt (1982). For independent units $i = 1, 2, \ldots, n$, measured at times $t = 1, 2, \ldots, T_i$, the random-effects probit model is

$$y^*_{it} = \mathbf{x}_{it}\mathcal{B} + \nu_i + \epsilon_{it}$$

$$y_{it} = \begin{cases} 1 & \text{if } y^*_{it} > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\nu_i \sim N(0, \sigma^2_\nu)$, $\epsilon_{it} \sim N(0, \sigma^2_\epsilon)$, with ν_i and ϵ_{it} independent (and also both independent of $\mathbf{x}_i$). The model for the unobserved continuous y^*_{it} is the estimate from Stata's `xtreg` command. In the probit model for y_{it}, however, $\mathcal{B}$, σ_ν, and σ_ϵ are not all identifiable, so we arbitrarily set $\sigma_\epsilon = 1$. Further, rather than using σ_ν, we parameterize the likelihood in terms of the within-subject correlation

$$\rho = \frac{\sigma^2_\nu}{\sigma^2_\nu + \sigma^2_\epsilon}$$

If we condition on ν_i and $\mathbf{x}_i$, the y_{it} are independent, and we have

$$P(\mathbf{y}_i|\mathbf{x}_i, \nu_i) = \prod_{t=1}^{T_i} F(\mathbf{x}_{it}\mathcal{B} + \nu_i), \qquad \text{where } F(x) = \begin{cases} \Phi(x) & \text{if } y_{it} = 1 \\ 1 - \Phi(x) & \text{if } y_{it} = 0 \end{cases}$$

and Φ is the cumulative normal density function. Integrating over the distribution of ν_i gives

$$P(\mathbf{y}_i|\mathbf{x}_i) = \int_{-\infty}^{\infty} \frac{e^{-\nu^2/2\sigma^2_\nu}}{\sqrt{2\pi}\,\sigma_\nu} \left[\prod_{t=1}^{T_i} F(\mathbf{x}_{it}\mathcal{B} + \nu)\right] d\nu$$

The log-likelihood is then $l = \sum_{i=1}^{n} \log P(\mathbf{y}_i|\mathbf{x}_i)$.

The integral for $P(\mathbf{y}_i|\mathbf{x}_i)$ can be approximated using M-point Gaussian–Hermite quadrature:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)\,dx \approx \sum_{m=1}^{M} w_m f(x_m)$$

The weights w_m and points x_m are computed using the algorithm described in Press et al. (1992). The quadrature formula requires that $f(x)$ can be well-approximated by a polynomial. As the time periods T_i become large, $\prod_{t=1}^{T_i} F(\mathbf{x}_{it}\mathcal{B} + \nu_i)$ is no longer well-behaved. Thus, this procedure should only be used for small-to-moderate T_i (Borjas and Sueyoshi 1994). Based on simulations, $\max T_i \leq 50$ appears to be a safe upper bound. Other than this limit, there are no restrictions on T_i; **rfprobit** handles unbalanced data as well as balanced.

Example

Using the NLSY data (Center for Human Resource Research 1989) on women aged 14–24 in 1968, we model union membership (1 if union; 0 if not union) using random-effects probit. Women were surveyed in 16 years during the period 1968–1988, and we restrict our data set to those women who have completed their education. The number of women in this subsample is $n = 4148$ with T_i ranging from 1 to 12 with a median of 4 and $N =$ 19,213 total observations.

```
. rfprobit union age grade black msp c_city not_smsa south, i(idcode)
Constant-only model
rho = 0.0     Log Likelihood = -10459.226
rho = 0.1     Log Likelihood ~ -9457.8385
rho = 0.2     Log Likelihood ~ -8921.8396
rho = 0.3     Log Likelihood ~ -8584.5937
rho = 0.4     Log Likelihood ~ -8361.8598
rho = 0.5     Log Likelihood ~ -8225.3245
rho = 0.6     Log Likelihood ~ -8174.1754
rho = 0.7     Log Likelihood ~ -8243.8223
Iteration 0:  Log Likelihood = -8174.1754
Iteration 1:  Log Likelihood = -7999.4702
Iteration 2:  Log Likelihood = -7959.4034
Iteration 3:  Log Likelihood = -7956.9872
Iteration 4:  Log Likelihood =  -7956.984
Iteration 5:  Log Likelihood =  -7956.984
Full model
rho = 0.0     Log Likelihood = -10029.956
Iteration 0:  Log Likelihood = -8028.1384
Iteration 1:  Log Likelihood = -7797.9036
Iteration 2:  Log Likelihood = -7776.2779
Iteration 3:  Log Likelihood =  -7776.233
Iteration 4:  Log Likelihood =  -7776.233
Random-Effects Probit                              Number of obs    =   19213
                                                   Model chi2(7)    =  361.50
                                                   Prob > chi2      =  0.0000
Log Likelihood =  -7776.2329784                    Pseudo R2        =  0.0227

------------------------------------------------------------------------------
   union |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
union    |
     age |   .0099185   .0026351      3.764   0.000       .0047538    .0150833
   grade |   .0710084   .0112895      6.290   0.000       .0488814    .0931353
   black |    .679326     .06209     10.941   0.000       .5576319    .8010201
     msp |  -.0296221   .0383182     -0.773   0.439      -.1047244    .0454803
  c_city |   .1560987   .0483332      3.230   0.001       .0613674    .2508301
not_smsa |  -.0568532   .0588848     -0.965   0.334      -.1722654     .058559
   south |  -.7573341   .0522798    -14.486   0.000      -.8598006   -.6548676
   _cons |  -2.448033   .1758944    -13.918   0.000       -2.79278   -2.103286
---------+--------------------------------------------------------------------
rho      |
   _cons |   .6374987    .010156     62.770   0.000       .6175933    .6574042
------------------------------------------------------------------------------
LR test of rho = 0:   chi2(1)     = 4507.45
                      Prob > chi2 =  0.0000
```

We compare the results to a standard probit model:

```
. probit union age grade black msp c_city not_smsa south, nolog

Probit Estimates                                      Number of obs =  19213
                                                      chi2(7)       = 858.54
                                                      Prob > chi2   = 0.0000
Log Likelihood = -10029.956                           Pseudo R2     = 0.0410

------------------------------------------------------------------------------
   union |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   .0064769   .0016737      3.870   0.000       .0031965    .0097572
   grade |   .0315457   .0043774      7.207   0.000       .0229662    .0401251
   black |   .4617062   .0249721     18.489   0.000       .4127618    .5106506
     msp |  -.0104392   .0214401     -0.487   0.626      -.0524609    .0315826
  c_city |   .0774842   .0249138      3.110   0.002        .028654    .1263144
not_smsa |  -.0363566    .026501     -1.372   0.170      -.0882976    .0155845
   south |  -.4986367   .0229117    -21.763   0.000      -.5435429   -.4537305
   _cons |  -1.295481   .0766311    -16.905   0.000      -1.445675   -1.145286
------------------------------------------------------------------------------
```

The results for the two probit model are not qualitatively different, but, as we would expect, the z statistics for the coefficients are generally less extreme (i.e., less significant) for the random-effects model. We could also use Stata's `hprobit` to analyze these data:

```
. hprobit union age grade black msp c_city not_smsa south, group(idcode)

Probit Regression with Huber standard errors         Number of obs =   19213
Log Likelihood =-10029.956                           Pseudo R2     =  0.0410

Grouping variable: idcode
------------------------------------------------------------------------------
   union |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   .0064769   .0023834      2.718   0.007       .0018055    .0111482
   grade |   .0315457   .0086773      3.635   0.000       .0145384    .0485529
   black |   .4617062   .0462117      9.991   0.000        .371133    .5522794
     msp |  -.0104392   .0346756     -0.301   0.763       -.078402    .0575237
  c_city |   .0774842   .0433076      1.789   0.074      -.0073972    .1623656
not_smsa |  -.0363566   .0516286     -0.704   0.481      -.1375468    .0648336
   south |  -.4986367   .0432001    -11.542   0.000      -.5833073   -.4139661
   _cons |  -1.295481   .1385339     -9.351   0.000      -1.567002   -1.023959
------------------------------------------------------------------------------
```

`hprobit`'s z statistics are, for most of the coefficients, the least extreme of the three.

Computation time

`rfprobit` is slow. The preceding example with $N =$ 19,213 observations took about 8 hours to run on a bottom-of-the-line IBM RS-6000 (Pentiums are about 4 times faster). Remember, this is a big data set—almost 20,000 observations—and computation time is roughly proportional to N, so you can get results for small ($N <$ 1000) data sets in less time ($<$ 30 minutes).

But the ado-file program for `rfprobit` is slow. There are two reasons for this. The first is the need to loop over the M points for the Gaussian–Hermite quadrature. Because of this, computation time is roughly proportional to M. By default, M is set to a small value: $M =$ 6. M can be changed using the `quadrat(#)` option, but, in the simulations I have run, increasing M did not appreciably change the estimates for the coefficients or their standard errors. However, users may want to increase M when fitting their final model.

For specification searches, the options `nochisq` and `ltol(1e-4)` will reduce computation time by roughly 10–20 percent. Using `probit` or `hprobit` for initial specification searches is also a good idea.

Stata's ml commands

The second reason for `rfprobit`'s slowness is the use of the `deriv1` method of Stata's `ml` command. The `deriv0` and `deriv1` methods are intended as quick-and-dirty techniques (quick for the user to implement, that is). The `deriv1` method uses analytic first derivatives and numerical second derivatives, and because of this, its computation time is roughly proportional to the number of variables in the model. The `deriv0` method uses numerical first and second derivatives, and its computation time is roughly proportional to the square of the number of variables in the model. Actually, there is nothing "dirty" about `deriv0` and `deriv1`. With the updated versions of the `ml` commands distributed in STB-25, these methods give excellent accuracy.

The preferred methods to use with the `ml` commands are the `lf` (linear form) and `deriv2` (analytic first and second derivatives) methods. The `lf` method is easy to program, fast, and accurate, but requires that the log-likelihood be a simple sum over the observations (i.e., all observations independent) of a function of $\mathbf{x}_i\mathcal{B}$. Unfortunately, the log-likelihoods for random-effects models do not meet this requirement.

The `deriv2` method can be fast if the ado-file that computes the likelihood can be implemented without explicit looping over the variables in model. The usual trick for doing this is to use the `matrix accum` command, which computes terms of the form $\mathbf{X}'\mathbf{W}\mathbf{X}$, where $\mathbf{W}$ is diagonal. But, not all the terms in the likelihoods for random-effects models fit this form. The `matrix glsaccum` command is another tool one can possibly use.

The bottom line for random-effects probit is that the `deriv2` method is not feasible to implement—at least for this approach for computing the likelihood. A fast command for random-effects probit will only come when it is implemented as an internal command in Stata.

References

Borjas, G. J. and G. T. Sueyoshi. 1994. A two-stage estimator for probit models with structural group effects. *Journal of Econometrics* 64: 165–182.

Butler, J. S. and R. Moffitt. 1982. A computationally efficient quadrature procedure for the one-factor multinomial probit model. *Econometrica* 50: 761–764.

Center for Human Resource Research. 1989. *National Longitudinal Survey of Labor Market Experience, Young women 14–24 years of age in 1968.* Ohio State University.

Hamerle, A. and G. Ronning. 1995. Panel analysis for qualitative variables. In *Handbook of Statistical Modeling for the Social and Behavioral Sciences*, ed. G. Arminger, C. C. Clogg, and M. E. Sobel, 401–451. New York: Plenum Press.

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

sg42 Plotting predicted values from linear and logistic regression models

Joanne Garrett, University of North Carolina, FAX 919-966-2274

In linear and logistic regression models, it is often easier to interpret the plot of fitted values against a continuous predictor than to infer the same information from the table of regression coefficients. This insert presents two programs, `regpred` and `logpred`, designed to make the calculations and plots simple to do. `regpred` estimates a linear regression model (using `regress`) and `logpred` fits a logistic regression model (using `logistic`) to calculate and plot predicted values and 95 percent confidence intervals of the predictions for user-specified values of a continuous predictor variable. Although the model is fit using all the data, predictions are estimated and plotted only for the requested values of the predictor. Restricting the values against which predictions are plotted can produce a graph that is less "busy" than a graph that includes every possible value. All estimates are adjusted to the means of any covariates in the model. A linear specification of the predictor variable is assumed, but quadratic or cubic forms can be requested. By default, the model, the plot, and a list of the estimates are printed, but any of these can be suppressed.

Syntax and options

`regpred` and `logpred` employ the same syntax:

{ `regpred` | `logpred` } *yvar xvar* [`if` *exp*] `,` `from(`*#*`)` `to(`*#*`)` [`inc(`*#*`)` `adjust(`*covlist*`)` `poly(`*#*`)` `nomodel` `nolist` `noplot` *graph-options*]

In both commands, *xvar* is a continuous predictor variable. *yvar* is the outcome variable, which is continuous in `regpred` and binary (coded 0/1) in `logpred`.

`from(`*#*`)` specifies the minimum value of *xvar*, the predictor variable.

`to(`*#*`)` specifies the maximum value of *xvar*.

`inc(`*#*`)` specifies the amount by which *xvar* is incremented between `from()` and `to()`. The default increment is 1.

`adjust(`*covlist*`)` specifies the list of other covariates in the model, that is, the variables for which the estimates are adjusted.

`poly(`*#*`)` indicates the order of the polynomial used for *xvar* in the model. The default is '1', that is, a linear specification. Quadratic (`poly(2)`) and cubic (`poly(3)`) specifications also may be requested.

`nomodel` suppresses the output from the `regress` or `logistic` command.

`nolist` suppresses the list of predicted values and confidence intervals.

`noplot` suppresses the graph.

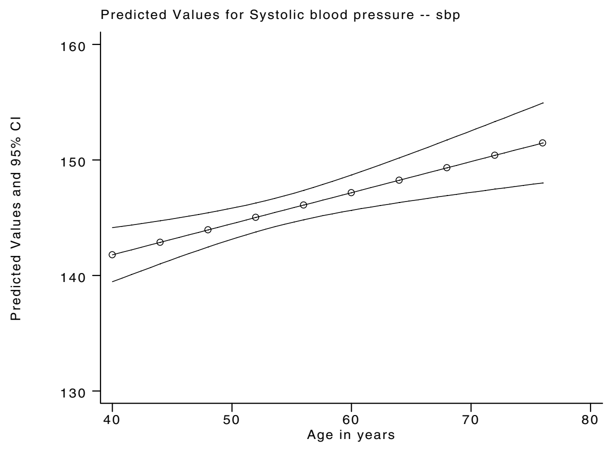

Figure 1: Systolic blood pressure as a function of age

Example 1

All the examples use data from a follow-up study of coronary heart disease among a cohort of 35 to 80-year-old men. In the first example, we use `regpred` to examine the effect of age on systolic blood pressure adjusted for a measure of socio-economic status (`ses`), cholesterol level (`chl`), a dummy variable for catecholamine level (`cat`: 1 = high, 0 = low), and a dummy variable for the results of an electrocardiogram (`ecg`: 1 = abnormal, 0 = normal). We request the predicted values of systolic blood pressure for ages 40 to 76 in increments of 4 years. In addition to the plot of predicted values, `regpred` displays the output from `regress` and a list of the ten requested predictions and their 95 percent confidence intervals.

```
. use evans2
(Example CHD Data)

. regpred sbp age, from(40) to(76) inc(4) adj(ses chl cat ecg) xlab ylab

  Source |       SS       df       MS                  Number of obs =    1218
---------+------------------------------               F(  5,  1212) =  142.50
   Model |  340431.471     5  68086.2942               Prob > F      =  0.0000
Residual |  579108.338  1212   477.81216               R-squared     =  0.3702
---------+------------------------------               Adj R-squared =  0.3676
   Total |   919539.81  1217  755.579137               Root MSE      =  21.859

------------------------------------------------------------------------------
     sbp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     age |   .2687184   .0740076      3.631   0.000       .1235211    .4139157
     ses |   .1799013   .0471049      3.819   0.000       .0874852    .2723175
     chl |   .0862696   .0162771      5.300   0.000       .0543352    .1182039
     cat |   33.78119   1.840977     18.350   0.000       30.16933    37.39304
     ecg |    8.60972   1.525701      5.643   0.000       5.616411    11.60303
   _cons |   93.25532   6.027453     15.472   0.000       81.42992    105.0807
------------------------------------------------------------------------------
```

(graph appears, see Figure 1)

```
Predicted Values and 95% Confidence Intervals

  Outcome Variable:     Systolic blood pressure -- sbp
  Independent Variable: Age in years -- age
  Covariates:           ses chl cat ecg
  Total Observations:   1218

          age     pred_y      lower      upper
  1.       40   141.7931   139.4565   144.1297
  2.       44   142.8680   141.0000   144.7359
  3.       48   143.9429   142.4623   145.4234
  4.        52   145.0177   143.7654   146.2700
  5.        56   146.0926   144.8207   147.3645
  6.        60   147.1675   145.6376   148.6974
  7.        64   148.2424   146.3093   150.1754
  8.        68   149.3172   146.9077   151.7268
  9.        72   150.3921   147.4683   153.3159
 10.        76   151.4670   148.0080   154.9260
```

Systolic blood pressure is positively and significantly associated with an increase in age ($\beta_{\rm age} = 0.2687$, $p = 0.000$). Predicted values and 95 percent confidence intervals are calculated and plotted for ages $40, 44, 48, \ldots, 76$. The graph displays a modest increase in systolic blood pressure from about 142 mm/Hg for a 40 year old to about 150 mm/Hg for someone who is 76. The list of the predicted values and confidence intervals also displays the outcome variable, the independent variable, any covariates, and the number of observations used in the regression.

Example 2

Using the same model, we request a graph showing the relationship between systolic blood pressure and cholesterol level (controlling for `age`, `ses`, `cat`, and `ecg`). Because we have already seen the regression table in the previous example, we use the `nomodel` option to suppress it now. Cholesterol values for this data set range from 90 to 350. We request the predictions of systolic blood pressure corresponding to a low cholesterol value of 100 to a high of 320, in increments of 20.

```
. regpred sbp chl, f(100) t(320) i(20) adj(age ses cat ecg) xlab ylab nomodel
```

(graph appears, see Figure 2)

```
 Predicted Values and 95% Confidence Intervals

  Outcome Variable:     Systolic blood pressure -- sbp
  Independent Variable: Serum cholesterol -- chl
  Covariates:           age ses cat ecg
  Total Observations:   1218

           chl     pred_y      lower      upper
  1.       100   135.8365   132.0663   139.6068
  2.       120   137.5619   134.3881   140.7357
  3.       140   139.2873   136.6902   141.8844
  4.       160   141.0127   138.9556   143.0698
  5.       180   142.7381   141.1468   144.3294
  6.       200   144.4635   143.1800   145.7469
  7.       220   146.1889   144.9333   147.4445
  8.       240   147.9143   146.3911   149.4374
  9.       260   149.6396   147.6705   151.6088
 10.       280   151.3650   148.8651   153.8649
 11.       300   153.0904   150.0187   156.1622
 12.       320   154.8158   151.1503   158.4814
```

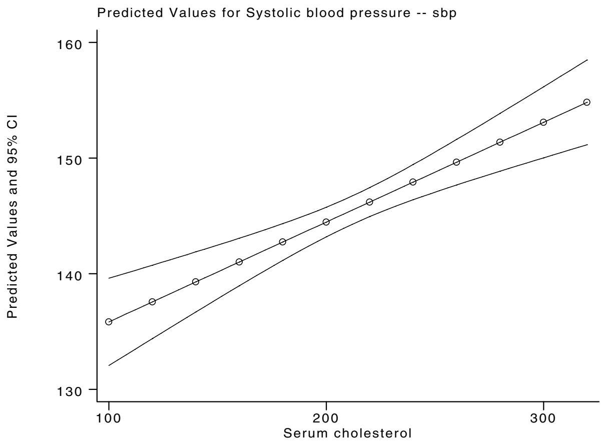

Figure 2: Systolic blood pressure as a function of cholesterol level

Systolic blood pressure is positively and significantly associated with an increase in cholesterol ($\beta_{\rm chl} = 0.0863$, $p = 0.000$). Predicted values and confidence intervals are calculated and plotted for cholesterol values of $100, 120, 140, \ldots, 320$. The graph shows an increase in systolic blood pressure from about 135 mm/Hg for an individual with a cholesterol reading of 100 to about 155 mm/Hg for someone with a cholesterol level over 300.

Example 3

To illustrate the use of `logpred`, we examine the effect of cholesterol level on the development of coronary heart disease, measured as a binary variable (`chd`: 1 = develops coronary heart disease, 0 = no coronary heart disease), controlling for catecholamine level (`cat`), electrocardiogram abnormality (`ecg`), age, and smoking status (`smk`: 1 = smoker, 0 = non-smoker). Once again, we request predictions corresponding to values of cholesterol from 100 to 320 in increments of 20. This time our results are the predicted probabilities, derived from the logistic regression model, of developing coronary heart disease.

```
. logpred chd chl, f(100) t(320) i(20) adj(cat ecg age smk) xlab
        ylab(0,.1,.2,.3,.4) b2(Serum Cholesterol -- Linear)
Logit Estimates                                       Number of obs =   1218
                                                      chi2(5)       =  71.77
                                                      Prob > chi2   = 0.0000
Log Likelihood = -402.67442                           Pseudo R2     = 0.0818

------------------------------------------------------------------------------
     chd | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     chl |   1.009411   .0023077      4.097   0.000       1.004898    1.013944
     cat |   2.172936   .5117931      3.295   0.001       1.369502    3.447712
     ecg |   1.516705    .313654      2.014   0.044       1.011283    2.274727
     age |   1.033073     .01107      3.036   0.002       1.011602    1.054999
     smk |   2.288827   .4923483      3.849   0.000       1.501452    3.489108
------------------------------------------------------------------------------
```

(graph appears, see Figure 3)

```
Predicted Values and 95% Confidence Intervals

  Outcome Variable:     Coronary heart disease -- chd
  Independent Variable: Serum cholesterol -- chl
  Covariates:           cat ecg age smk
  Total Observations:   1218

           chl       pred      lower      upper
  1.       100    .036077   .0203964   .0630371
  2.       120   .0431892   .0265678   .0694673
  3.       140   .0516284   .0344545   .0766827
  4.       160   .0616104   .0443882   .0849209
  5.       180   .0733731   .0565853   .0946419
  6.       200   .0871729   .0708539   .1068183
  7.       220   .1032788   .0861404   .1233667
  8.       240   .1219628   .1008455   .1467803
  9.       260   .1434861   .1144303   .1784332
 10.       280   .1680806   .1274682   .2183937
 11.       300   .1959264   .1405662   .2663331
 12.       320    .227126   .1540861   .3216245
```

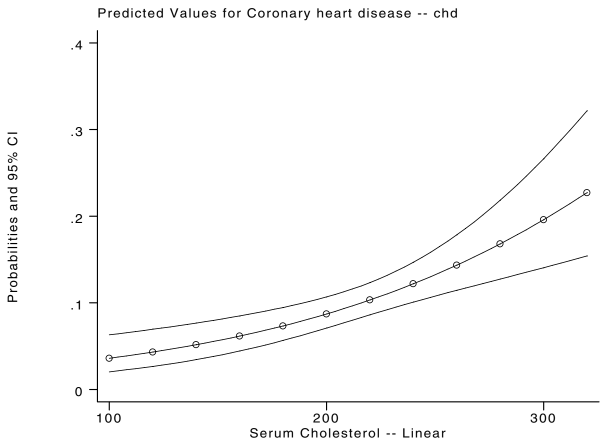

Figure 3: Probability of heart disease as a function of cholesterol level

Cholesterol is positively associated with the probability of developing coronary heart disease ($p < 0.001$). For example, people with a cholesterol level of 100 have a probability of about 0.03 of developing coronary heart disease, adjusted for the covariates in the model, while people with a cholesterol reading of 300 have a probability of 0.20 of developing coronary heart disease. The confidence interval around the probabilities starts to get rather wide as cholesterol increases, which suggests the data may be getting sparse at higher levels of cholesterol or the relationship may not be linear (in the log odds) in that range. The next example explores the latter possibility.

Example 4

We repeat Example 3, but we use the `poly` option to allow the probability of heart disease to be a cubic function of cholesterol level. The model estimates are displayed so we can examine the *p*-values for the two additional regressors. Note that the square and cube of cholesterol need not be generated in advance; they are created temporarily and then dropped. Since we are mainly interested in the plot, the final summary table is suppressed.

```
. logpred chd chl, f(100) t(320) i(10) adj(cat ecg age smk) xlab
         ylab(0,.1,.2,. 3,.4) b2(Serum Cholesterol -- Cubic) poly(3) nolist
Logit Estimates                                        Number of obs =   1218
                                                       chi2(7)       =  81.30
                                                       Prob > chi2   = 0.0000
Log Likelihood = -397.90735                            Pseudo R2     = 0.0927

------------------------------------------------------------------------------
     chd | Odds Ratio   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     chl |   1.450816   .1919012      2.813   0.005       1.119497    1.880191
    x_sq |   .9984854   .0005558     -2.723   0.006       .9973966    .9995753
  x_cube |   1.000002   7.64e-07      2.680   0.007       1.000001    1.000004
     cat |   2.151983   .5098775      3.235   0.001       1.352572    3.423869
     ecg |   1.551171   .3244517      2.099   0.036        1.02948    2.337232
     age |   1.032317   .0111402      2.947   0.003       1.010712    1.054384
     smk |   2.372324   .5158739      3.973   0.000       1.549087    3.633056
------------------------------------------------------------------------------
```

(graph appears, see Figure 4)

This graph suggests a somewhat different interpretation than does the graph from the linear model. In the cubic model, the probability of developing coronary heart disease increases as cholesterol increases to just above 200, levels off until the cholesterol values start approaching 300, and then starts increasing rather dramatically (although the confidence interval does get quite wide at the highest values).

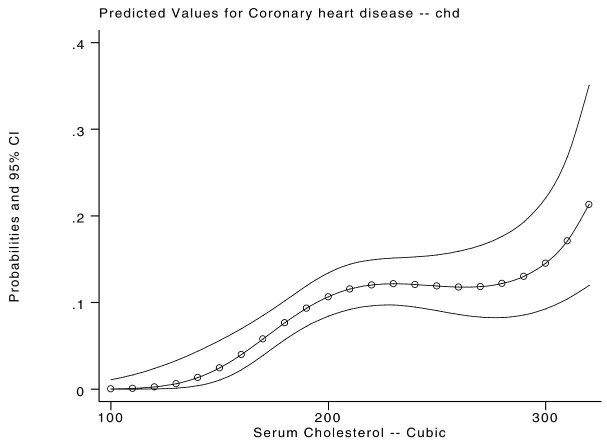

Figure 4: Probability of heart disease as a cubic function of cholesterol level

Note

`regpred` and `logpred` are similar to the `adjmean` and `adjprop` programs I presented in *sg33* (Garrett 1995). `adjmean` and `adjprop` calculate adjusted means and adjusted probabilities, respectively, for nominal, rather than continuous, predictor variables.

Reference

Garrett, J. 1995. sg33: Calculation of adjusted means and adjusted proportions. *Stata Technical Bulletin* 24: 22–25.

snp6.1 ASH, WARPing, and kernel density estimation for univariate data

Isaías Hazarmabeth Salgado-Ugarte, Makoto Shimizu, and Toru Taniuchi,
University of Tokyo, Faculty of Agriculture, Department of Fisheries, Japan
FAX (011)-81-3-3812-0529, EMAIL isalgado@tansei.cc.u-tokyo.ac.jp

Kernel density estimators are important tools for exploring and analyzing data distributions (see the references in Salgado-Ugarte et al. 1993). However, one drawback of these procedures is the large number of calculations required to compute them. As a consequence, it can be time consuming to compute kernel density estimators even for moderate sample sizes and when using fast processors. Scott (1985) suggested an alternative procedure to overcome this problem: the *A*veraged *S*hifted *H*istogram (ASH). Subsequently, Härdle and Scott (1988) developed a more general framework called WARPing (weighted averaging of rounded points).

This insert, based mainly on some chapters from the books by Härdle (1991) and Scott (1992), briefly introduces the ASH and WARPing procedures for density estimation and presents some ado-files and Turbo Pascal programs for their calculation.

Averaged shifted histograms and WARPing

As discussed in *snp6*, the histogram is defined by specifying two parameters: the origin, x_0, and the width, h, of the bins. Substantial evidence has accumulated (Silverman 1986; Fox 1990) that the choice of origin may have an important influence on the resulting histogram, despite some theoretical results indicating that this choice should have a negligible impact (Scott 1992). To demonstrate this phenomenon, we use well-known data on snowfall in Buffalo, New York (Parzen 1979). These data measure the annual snowfall in inches in each of the 63 winters from 1910/11 through 1972/73. The following Stata commands produce five different histograms for these data. Each histogram uses the same bin width ($h = 10$) but a different origin, and each histogram provides a valid density estimate (after rescaling).

```
. use bufsnow
. graph snow, bin(11) xscale(20,130)
. graph snow, bin(11) xscale(22,132)
. graph snow, bin(11) xscale(24,134)
. graph snow, bin(12) xscale(16,136)
. graph snow, bin(11) xscale(18,128)
```

The resulting graphs (after some editing with Stage) appear as Figure 1. (The final graph in Figure 1 will be explained below.) Some of these histograms are unimodal, others are bimodal and even trimodal. Scanning across the histograms, there appears to be a main mode around 80 inches. Some of the histograms indicate secondary modes around 50 or 100 inches. We can choose any of these histograms to represent the data distribution, but our choice would be arbitrary.

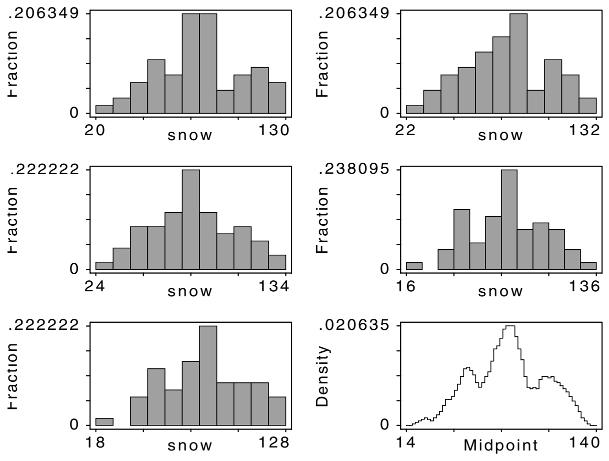

Figure 1. Five histograms with different origins and the corresponding ASH

To eliminate the influence of the choice of origin, Scott (1985) suggested an ingenious device: instead of choosing among several histograms, Scott proposed averaging several histograms with different origins to produce the average shifted histogram (ASH).

It is perhaps easiest to understand the ASH and WARP by tracing their development from the definition of a simple histogram. We can assume, without loss of generality, that all n observations of the variable of interest lie in the half-open interval $[0, Kh)$. (Clearly, we can translate the values of any n observations to lie within any specified interval.) We partition this interval into $K+1$ bins, each with width h. The kth bin, B_k, is defined as

$$B_k = [kh, (k+1)h), \quad k = 0, \dots, K$$

The histogram is defined as

$$\widehat{f}(x) = \frac{\nu_k}{nh} = \frac{1}{nh}\sum_i I_{[t_k, t_{k+1})}(x_i)$$

where ν_k is the number of observations in B_k, and I is the indicator function, equal to one when x_i lies in the specified interval and zero otherwise.

Now consider a collection of M histograms, $\widehat{f}_1$, $\widehat{f}_2$, $\dots \widehat{f}_M$, each with the bin width, h, and with the sequence of bin origins

$$t_0 = 0, \frac{h}{M}, \frac{2h}{M}, \dots, \frac{(M-1)h}{M}$$

We are now adding the inessential restriction

$$x_i \geq \frac{(M-1)h}{M}$$

The naive (unweighted) ASH is defined as

$$\widehat{f}(\bullet) = \widehat{f}_{ASH}(\bullet) = \frac{1}{M}\sum_{i=1}^{M}\widehat{f}_i(\bullet)$$

It is convenient in what follows to define the bins to be narrower, depending on the number of histograms to be averaged,

$$B_k = [k\delta, (k+1)\delta), \quad \delta = h/M$$

ν_k continues to denote the number of observations in B_k. The expression for the ASH can be generalized to

$$\begin{aligned} f(x;M) &= \frac{1}{M}\sum_{i=1-M}^{M-1}\frac{(M-|i|)\nu_{k+i}}{nh} \\ &= \frac{1}{n}\sum_{i=1-M}^{M-1}\left(1-\frac{|i|}{M}\right)\nu_{k+i} \quad \text{for } x \in B_k \end{aligned}$$

Since it is the average of a sequence of histograms, the ASH has the appearance of a histogram as well, although the ASH can be made continuous using linear interpolation schemes. The interpolation approach produces the frequency polygon of the averaged shifted histogram (FP-ASH).

The ASH method is a special case of the more general WARP or weighted average of rounded points, developed by Härdle and Scott (1988). The general expression for the WARP, and hence, by extension, the ASH, is

$$\widehat{f}(x;M) = \frac{1}{nh}\sum_{|i|<M}^{M-1} w_{M(i)}\nu_{k+1} \quad \text{for} x \in B_k$$

where the weights are defined by

$$w_M(i) = M \times \frac{K(i/M)}{\sum_{j=1-M}^{M-1} K(j/M)} \quad i = 1-M, \dots, M-1$$

As these formulae indicate, WARPing is based on the smaller bin mesh (B_k) defined by h, the bandwidth, and a new parameter, M, that indicates the number of shifted histograms to average. The rounded points are the bin counts in the B_k. The weighting operation is symbolized by $w_{M()}$, and the specification of this weighting function determines which kernel density estimator is used in computing the WARP.

The calculation of a WARP estimate takes three steps: (1) binning the data; (2) calculating the weights; and (3) weighting the bins. In the first step, a mesh of intervals is created and the number of observations in each interval is counted. The information about the data is reduced to a list of bin counts along with their midpoints. In the second step, a nonnegative, symmetric weight function is calculated. The weights are normalized to sum to M, the number of shifted histograms. For example, the weight function for the ASH is

$$w_{M(i)} = 1 - (|i|/M)$$

Finally, the density estimate in each bin is computed as the product of the bin count and the weight. This brief description covers only the highlights of the procedure. See Härdle (1991) or Scott (1992) for more details.

WARPing in Stata

Programs to calculate WARPs are available from a number of sources. Scott (1992) and Härdle (1991) present several algorithms for calculating WARPs and other density estimators. They also provide program listings in FORTRAN, C, and the S language. In addition, Brian Ripley maintains a full collection of C programs and S functions from Härdle's book. These programs are available from Statlib via EMAIL or FTP.

Based on these examples, we decided to write a Stata program to calculate WARPs. However, as we noted in the introduction, kernel density estimators, which are required to calculate the WARPs, are time-consuming to compute. To speed the process, we wrote a program in Turbo Pascal that performs the key calculations at high speed. This Pascal program is called from a Stata ado-file and, hence, is invisible to the Stata user. This approach allows users to retain the ease of use of Stata while gaining the speed advantage of the compiled Pascal program.

The Pascal program is supplied in two forms on the distribution diskette. `warpings.com` is a binary executable that can be used with the DOS and Windows versions of Stata. `warpings.pas` contains the source code of the program. This source can be adapted by the reader for other operating systems.

For those readers who will be adapting the source file, here is a brief overview of `warpings.pas`. The program is divided into a main routine and eight subroutines. `DataInput` requests the name of an ASCII file that contains the raw data. This file should consist of a single column of values and must not contain any missing values. This file can be created by Stata's `outfile` command. The program then prompts the user to specify the bandwidth, h, and the number of histograms to shift and average, M. Next, the subroutine `SelectKernel` prompts the user to select the type of weight function (kernel) to use. The choices are: uniform, triangular (ASH), Epanechnikov, quartic, triweight, and Gaussian. The `SortData` subroutine arranges the observations in ascending order. `InitialCalc` determines the origin of the mesh according to the values specified for h and M. `BinmeshCalc` counts the number of observations in each bin. `CreateWeight` calculates the weight function selected by the user. `WeightBins` forms the product of the bin counts and the weights. Finally, `ResulFile` writes an ASCII file, named `resfile`, that contains the density estimates and the corresponding bin midpoints.

We have integrated `warpings.com` into three Stata ado-files: `warpstep`, `warpoly`, and `warping`. `warpstep` and `warpoly` display graphs of WARP estimates. `warpstep` presents the estimates in histogram form, while `warpoly` linearly interpolates estimated points to display a frequency polygon. `warpings` generates new variables that contain the density estimates and bin midpoints.

Examples

We use the snowfall data introduced above to illustrate our programs. First, we use `warpstep` to display the ASH.

```
. warpstep

TYPE THE PATH, NAME AND EXTENSION OF TEXT DATA FILE
bufsnow.raw
THE NUMBER OF VALUES READ IS:         63
!!!!!WARNING!!!!!
IF THIS IS NOT CORRECT PLEASE INTERRUPT AND
USE STATA COMMAND outfile TO GENERATE AN ASCII
FILE WITH THE DESIRED DATA VECTOR
```

```
GIVE THE VALUE OF THE BANDWIDTH 'h'
10
GIVE THE NUMBER OF HISTOGRAMS TO SHIFT AND AVERAGE
5
SPECIFY THE WEIGHT FUNCTION:
1 = Uniform; 2 = Triangle (ASH); 3 = Epanechnikov
4 = Quartic; 5 = Triweight; 6 = Gaussian
2
(64 observations read)
```

The graph produced by **warpstep** is displayed as the last graph in Figure 1. This graph is the average of five shifted histograms of the snowfall data (as displayed in the first five graphs in Figure 1) where each histogram uses a bin width of 10 and the averaging uses the triangle weight function. The resulting ASH no longer depends in any important way on the origins of the original histograms. Notice that the ASH suggests that the unknown density may have three modes, a feature that is difficult to apprehend in the original histograms.

warping stores the calculations displayed by **warpstep** in two new variables. The syntax of **warping** is

warping *density midpoint*

where *denvar* and *midvar* are new variables that contain the density estimates and corresponding bin midpoints, respectively. **warping** prompts the user for the name of the ASCII file containing the data; the bandwidth; the number of histograms to calculate, shift, and average; and the weight function to use.

We can use **warping** to reproduce the ASH in Figure 1.

```
. warping density midpoint
[the sequence of prompts and responses is omitted]
. generate inter = midpoint[2] - midpoint[1]
. generate lowcut = midpoint - inter/2
. graph density lowcut, s(.) c(J) border
```

The resulting graph is essentially identical to the last graph that appears in Figure 1.

Figure 2 displays WARP estimates using $M \in \{1, 2, 4, 8, 16, 32\}$ and $h = 13.6$. The first of the six graphs is the ordinary histogram ($M = 1$). Two modes are apparent in this graph. Every ASH with $M > 2$ reveals the presence of a third mode to the left of the primary mode. The appearance of this additional mode is not an artifact of the WARPing procedure, but rather the result of a significantly improved signal-to-noise ratio obtained by averaging out the origin. The parameter for the origin, x_0, has been replaced by a different parameter: M, the number of shifted histograms. This replacement is justified by the improvement in the density estimate. Note, in particular, that the estimates are essentially the same for $M \geq 4$.

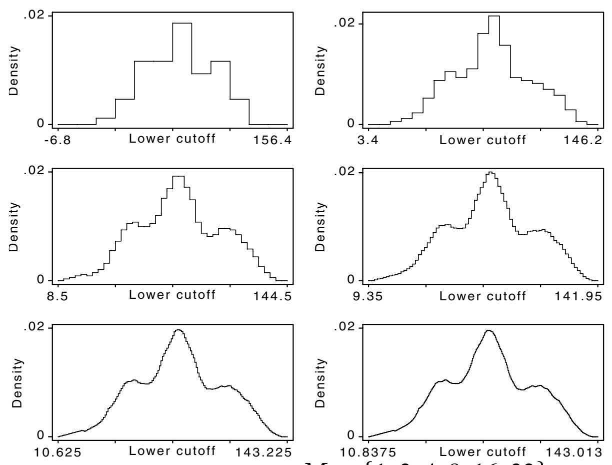

Figure 2. ASH with $M \in \{1, 2, 4, 8, 16, 32\}$

WARPs as approximate kernel density estimators

WARPing can be used to approximate a particular kernel density estimator by selecting the appropriate weight function. The WARP approaches the kernel function as the number of averaged histograms, M, increases (Härdle 1991). Figures 3 and 4 illustrate this fact. These figures display WARPs with $M \in \{1, 5, 15\}$ along with the analogous kernel density estimate.

Figure 3 displays the histogram-like, step WARP using the quartic weight function and, in the final graph, the quartic kernel density estimate for the Buffalo snowfall data. All estimates were calculated with $h = 10$. Figure 4 displays a similar sequence for the coral trout length data. Figure 4 shows the frequency polygon version of the WARP using the Gaussian weight function and, in the final graph, the Gaussian kernel density estimate. As these figures suggest, WARPs with $M \geq 5$ are nearly indistinguishable from the corresponding kernel estimates. Note, in Figure 4, that the WARP estimate with $M = 1$ fails to reveal the multimodality of the data. However, estimates with larger values of M clearly display the multiple modes.

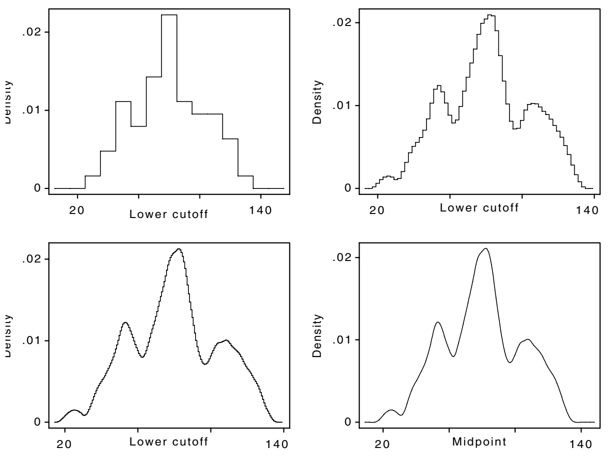

Figure 3. Step WARPs and the quartic kernel

In our previous insert (1993), we presented several programs to calculate kernel density estimators. It was necessary to revise these programs to support the calculation of WARPs. These updated programs are included on the distribution diskette. We also have added some new programs that augment the choice of kernel estimators. Our complement of ado-files now calculates estimates for the uniform, triangular (ASH), Epanechnikov, quartic (biweight), triweight, Gaussian, and cosinus kernels. (On-line help can be obtained by typing '`help kernel`'.)

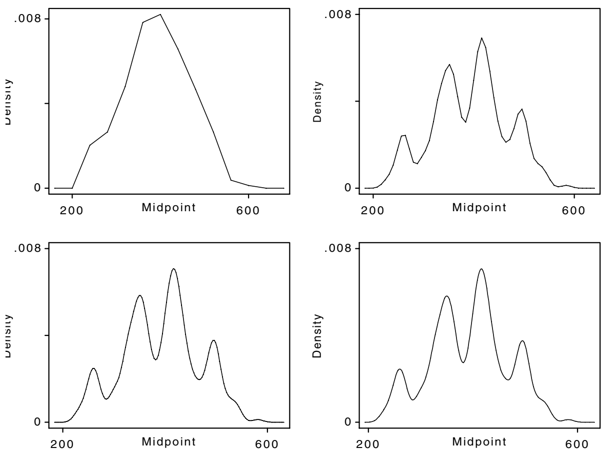

Figure 4. Polygon WARPs and the Gaussian kernel

All of these programs consider a mesh of 50 equally spaced points from $x_1 - h - (\text{range} * 0.1)$ to $x_n + h + (\text{range} * 0.1)$. To compare kernel estimates, the kernels must have the same support (Härdle 1991). Gasser et al. (1985) suggest comparing kernels over the common interval $[-1, 1]$. Applying that suggestion, our kernel programs are listed in the following table:

Table 1. Kernel programs provided

Program	Kernel	$K(z)$
`kernsim`	uniform	$\frac{1}{2}I(\lvert z\rvert \le 1)$
`kerntria`	triangle (ASH)	$(1-\lvert z\rvert)I(\lvert z\rvert \le 1)$
`kernepa`	Epanechnikov	$\frac{3}{4}(1-z^2)I(\lvert z\rvert \le 1)$
`kernquar`	quartic	$\frac{15}{16}(1-z^2)^2I(\lvert z\rvert \le 1)$
`kerntriw`	triweight	$\frac{35}{32}(1-z^2)^3I(\lvert z\rvert \le 1)$
`kerncos`	cosinus	$\frac{\pi}{4}\cos\frac{\pi}{2}zI(\lvert z\rvert \le 1)$
`kerngaus`	Gaussian	$\frac{1}{2\pi}e^{-z^2/2}$

Each of the kernel programs employs the same syntax:

program_name varname bandwidth density midpoint

where *varname* is the input variable, *bandwidth* is a scalar that specifies the half-width of each bin, and *density* and *midpoint* are new variables that will contain the density estimates and bin midpoints, respectively.

As an example, we can recreate the quartic kernel displayed as the last graph in Figure 3 by typing

```
. kernquar snow 10 den10 mid10
```
(output omitted)
```
. graph den10 mid10, xlab ylab c(s) s(.) border
```

(The graph shown in Figure 3 enjoyed some additional editing in Stage.)

Equivalent kernels

When we calculate two different kernels using the same window width, the results are not readily comparable. Consider, for example, Figure 5 which presents triweight and Gaussian kernels for the coral trout length data using $h_{\text{triweight}} = h_{\text{Gaussian}} = 15$. The triweight estimate is not as smooth as the Gaussian and suggests more modes.

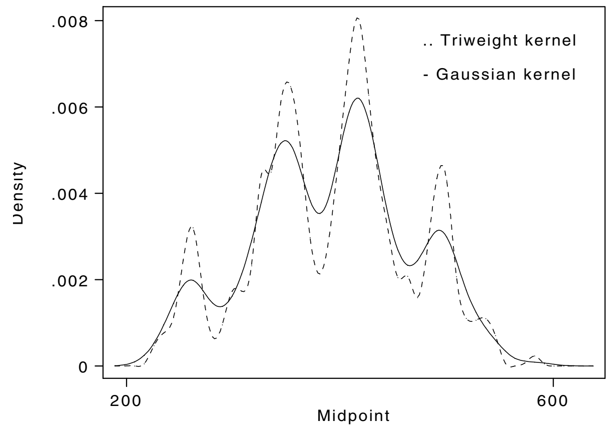

Figure 5. Gaussian and triweight kernels with $h_G = h_T = 15$

Different kernels have different variances, even when the kernels have identical support, and the difference in variances accounts for the qualitative differences in the density estimates. One way to correct for this difference is to adjust the window widths to produce equal variances. Scott (1976) calculates conversion factors that equalize the variances of different kernels. The following table summarizes conversion factors for some popular kernels.

Table 2. Inter-kernel conversion factors

To/From	Uniform	Triangle	Epanechnikov	Quartic	Triweight	Cosinus	Gaussian
Uniform		0.715	0.786	0.663	0.584	0.761	1.740
Triangle	1.398		1.099	0.927	0.817	1.063	2.432
Epanechnikov	1.272	0.910		0.844	0.743	0.968	2.214
Quartic	1.507	1.078	1.185		0.881	1.146	2.623
Triweight	1.711	1.225	1.345	1.136		1.302	2.978
Cosinus	1.315	0.941	1.033	0.872	0.768		2.288
Gaussian	0.575	0.411	0.452	0.381	0.336	0.437	

We can use these conversion factors to obtain approximately the same degree of smoothing with any pair of kernels. For example, Table 2 indicates we can produce a triweight kernel that is equivalent to a Gaussian kernel by using a bandwidth for the triweight kernel that is 2.978 times the bandwidth used for the Gaussian kernel. Figure 6 displays triweight and Gaussian kernels for the coral trout data where the bandwidth (h) of the Gaussian kernel is 15 and the bandwidth of the triweight kernel is $44.67 = 2.978 \times 15$. Now both density estimators show a similar degree of smoothness. A closely related but more general approach is the canonical bandwidth transformation proposed by Marron and Nolan (1988) in which the kernels functions are scaled to their canonical forms allowing the bandwidths to be equalized using the Gaussian kernel as the reference.

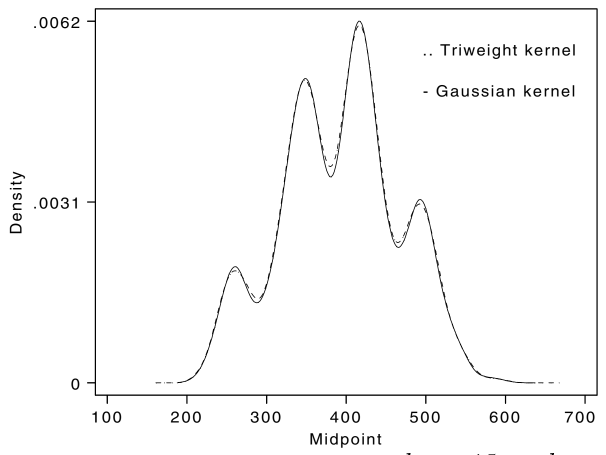

Figure 6. Gaussian and triweight kernels with $h_G = 15$ and $h_T = 44.67$

Some comparisons of WARP and kernel program performance

As documented in *snp6* (Salgado-Ugarte et al. 1993), calculating kernel density estimators is slow, even on a fast computer. The speed of the calculations depends crucially on n, the number of observations. In contrast, as we noted above, the WARPing technique uses a discretization of the data, therefore the calculations depend on n only at the binning stage. Since the results from any computerized procedure are discretized anyway when they are printed or displayed on screen, there is no loss of information (Härdle 1991).

Härdle and Scott (1988) compare the computational efficiency of WARPing and kernel density estimation in detail. As a guide to the relative performance of our implementations, we present some approximate timings in Table 3 below. The third column of the table displays a ‘W’ if the WARP program `warpstep` was used and a ‘K’ if one of the kernel density estimation programs was used. The value of h was set at 20 in each run of `warpstep` Times are measured, in seconds, from the last prompt of `warpings.com`. These timings were obtained using small Stata on a 25 MHz, 486SX processor with 6 MB of RAM and no math coprocessor. The data files listed are available on the distribution diskette and are described in *snp6*.

Table 3. Timing comparisons

Data file	n	W/K	M	Kernel	Bins	Time
`trocolen.raw`	316	W	15	Epanechnikov	303	5
”	”	W	15	Gaussian	303	4
”	”	W	50	Epanechnikov	1,004	13
`trocolen.dta`	”	K		uniform	50	150
”	”	K		Gaussian	50	170
`catfish.raw`	2,439	W	5	quartic	70	24
”	”	W	15	quartic	204	26
`catfish.dta`	”	K		quartic	50	1,200

The timings in Table 3 demonstrate the impressive time savings achieved by the WARP procedure, even when M and the number of bins become large. Because of this time savings, we now use `warpstep` and `warpoly` (with $M = 15$) to estimate kernels and to explore data distributions, and we use `warping` to produce numerical values for the estimates.

Some final notes

There is a limit, albeit a generous one, to the size of the problem that `warpings` can handle. The array definition in the Turbo Pascal program limits the number of sub-bins to 2000. For any particular series, the number of sub-bins is difficult to predict: it depends on the number of observations, on the intervals between the observations, and on the choice of M. As an example, though, with the Buffalo snowfall data, the maximum number of sub-bins is approached only when M exceeds 150. As the figures above have shown, setting $M \geq 5$ produces an adequate estimate of the kernel. Our convention is to set $M = 15$, which produces a smooth estimate without approaching the internal limits of `warpings`.

This suite of programs and ado-files have several other limitations. First, `warpstep` and `warpoly` use the `infile` command, thus, there can be no data in memory when the ado-files are called, or the programs will terminate with an error message. This feature may occasionally be a minor inconvenience, but it is safer than allowing `warpstep` and `warpoly` to silently delete the user's data set. Second, the input series must be stored as a single column of numbers in an ASCII data set. Third, none of the ado-files in the suite can handle `if` and `in` clauses. This design leaves the user with the responsibility for deleting unwanted cases prior to calling the WARP or kernel programs. Fourth, the results of `warping` must be saved and memory cleared before a new estimate can be calculated and compared. We ask users to let us know of any problems they encounter with these programs, and we also encourage them to send us any suggestions for improvements.

We have provided several example data sets on the distribution diskette, both as `.raw` and `.dta` files. We have included the Buffalo snow data (`bufsnow`, 63 observations), the coral trout length data (`trocolen`, 316 observations), and the catfish length data (`catfish`, 2,439 observations). The user can explore the distributions of these variables using different combinations of bandwidth, number of shifted histograms, and weight function. We also have included the source code of the Turbo Pascal program (`warpings.pas`) for users to examine and modify, if they wish.

Acknowledgments

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

References

Gasser, T., H. G. Müller, and V. Mammitzsch. 1985. Kernels for nonparametric curve estimation. *Journal of the Royal Statistical Society* Series B, 47: 238–252.

Härdle, W. 1991. *Smoothing Techniques with Implementations in S*. New York: Springer-Verlag.

Härdle, W. and D. W. Scott. 1988. Smoothing in low and high dimensions by weighted averaging using rounded points. Technical report 88–16, Rice University.

Marron, J. S. and D. Nolan. 1988. Canonical kernels for density estimation. *Statistics and Probability Letters* 7: 195–199.

Parzen, E. 1979. Nonparametrical statistical data modeling. *Journal of the American Statistical Association* 74: 105–131.

Salgado-Ugarte, I. H. 1985. Algunos aspectos biológicos del bagre *Arius melanopus* Gunther (Osteichthyes: Arridae) en el Sistema Lagunar de Tampamachoco. Ver. B. S. thesis, Carrera de Biología, E.N.E.P. Zaragoza, Universidad Nacional Autónoma de México.

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

Scott, D. W. 1976. Nonparametric probability density estimation by optimization theoretic techniques. Ph.D. thesis, Department of Mathematical Sciences, Rice University.

——. 1985. Averaged shifted histograms: effective nonparametric density estimators in several dimensions. *Annals of Statistics* 13: 1024–1040.

——. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: John Wiley & Sons.

snp8.1 Robust scatterplot smoothing: correction

Sean Becketti, Editor, Stata Technical Bulletin

In producing the distribution diskette for STB-25, I inadvertently omitted two ado-files from the *snp8* directory. This omission made it impossible to run the two-step lowess procedure. To correct this problem, I have reproduced the complete set of files for *snp8*—including the two omitted files—on the STB-26 distribution diskette.

I apologize to the authors and to the readers of the STB for this error, and I thank the readers who quickly brought this problem to my attention.

References

Salgado-Ugarte, I. S. and M. Shimizu. 1995. snp8: Robust scatterplot smoothing: enhancements to Stata's `ksm`. *Stata Technical Bulletin* 25: 23–29.

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	Company:	Oasis Systems BV
Address:	Prinzenstrasse 2 D-42697 Solingen Germany	Address:	Lekstraat 4 3433 ZB Nieuwegein The Netherlands
Phone:	+49 212-3390 99	Phone:	+31 3402 66336
Fax:	+49 212-3390 90	Fax:	+31 3402 65844
Countries served:	Austria, Germany	Countries served:	The Netherlands
Company:	Howching	Company:	Ritme Informatique
Address:	11th Fl. 356 Fu-Shin N. Road Taipei, Taiwan, R.O.C.	Address:	34 boulevard Haussmann 75009 Paris, France
Phone:	+886-2-505-0525	Phone:	+33 1 42 46 00 42
Fax:	+886-2-503-1680	Fax:	+33 1 42 46 00 33
Countries served:	Taiwan	Countries served:	Belgium, France, Luxembourg, Switzerland
Company:	Metrika Consulting	Company:	Timberlake Consultants
Address:	Ruddammsvagen 21 11421 Stockholm Sweden	Address:	47 Hartfield Crescent West Wickham Kent BR4 9DW, U.K
Phone:	+46-708-163128	Phone:	+44 181 462 0495
Fax:	+46-8-6122383	Fax:	+44 181 462 0493
Countries served:	Baltic States, Denmark, Finland, Iceland, Norway, Sweden	Countries served:	Eire, Portugal, U.K.