STATA TECHNICAL BULLETIN

September 1994

STB-21

A publication to promote communication among Stata users

Subscriptions are available from Stata Corporation, email stata@stata.com, telephone 979-696-4600 or 800-STATAPC, fax 979-696-4601. Current subscription prices are posted at www.stata.com/bookstore/stb.html.

Previous Issues are available individually from StataCorp. See www.stata.com/bookstore/stbj.html for details.

Submissions to the STB, including submissions to the supporting files (programs, datasets, and help files), are on a nonexclusive, free-use basis. In particular, the author grants to StataCorp the nonexclusive right to copyright and distribute the material in accordance with the Copyright Statement below. The author also grants to StataCorp the right to freely use the ideas, including communication of the ideas to other parties, even if the material is never published in the STB. Submissions should be addressed to the Editor. Submission guidelines can be obtained from either the editor or StataCorp.

The *Stata Technical Bulletin* (ISSN 1097-8879) is published six times per year by Stata Corporation. Stata is a registered trademark of Stata Corporation.

Contents of this issue

an1.1 STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:

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

Statistical Categories:

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

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

an47 New associate editors

Sean Becketti, Stata Technical Bulletin, FAX 914-533-2902

This issue marks the debut of a new editorial board for the *Stata Technical Bulletin*. Before introducing the new associate editors, though, I'd like to offer my thanks to the outgoing editorial board: J. Theodore Anagnoson, California State University, Los Angeles; Richard DeLeon, San Francisco State University; Paul Geiger, University of Southern California School of Medicine; Lawrence C. Hamilton, University of New Hampshire; and Stewart West, Baylor College of Medicine. Some of the editors are familiar to you through their many STB inserts. Others have contributed in less visible, but no less significant, ways to creating and sustaining the STB you are reading now. If you have found the STB useful, you owe these scholars a debt of gratitude. I hope and expect to continue receiving inserts and other support from the outgoing editors.

The success of the STB in promoting communication among Stata users and in making new features available quickly has far exceeded our original expectations. But even a venture as successful as the STB can benefit from new energy and new ideas. Thus I am very pleased to introduce the new editorial board of the STB. These individuals will build on and extend the pioneering efforts of our founding editorial board.

Francis X. Diebold is Associate Professor of Economics, University of Pennsylvania, and Faculty Research Fellow, National Bureau of Economic Research. Professor Diebold has published widely in econometrics, macroeconomics and finance, and he has served as a consultant to numerous organizations. Professor Diebold is on the editorial boards of *Econometrica*, *International Economic Review*, *Journal of Applied Econometrics*, *Journal of Business and Economic Statistics*, *Review of Economics and Statistics*, *Journal of Forecasting*, and *Journal of Empirical Finance*. His current research interests center on economic forecasting.

Joanne M. Garrett received a Master's degree (M.S.P.H.) in Biostatistics and a Ph.D. in Epidemiology at the School of Public Health at the University of North Carolina at Chapel Hill. From 1985 until 1990, she taught advanced quantitative methods courses in the Department of Epidemiology in the School of Public Health at UNC. In 1990, Dr. Garrett was appointed to the faculty of the Division of General Medicine and Clinical Epidemiology at UNC-CH, Adjunct Assistant Professor in the Department of Epidemiology in the School of Public Health, and a Core faculty member for the Robert Wood Johnson Clinical Scholars program. Her main activities have been teaching, consulting on study design and analysis issues, and working on several research projects. In 1992, Dr. Garrett became the Associate Director for Quantitative Methods at the Cecil G. Sheps Center for Health Services Research, where she was co-director of the NRSA fellowship program and a co-investigator on projects concerning ethical and cost issues for treatment of patients at the end of life, outcomes of care for acute low back pain, and improvement of childhood immunization rates. In addition, she has developed and presented an extensive set of short courses on logistic regression and other quantitative methods in epidemiology, and will be offering a series of these courses internationally for the International Clinical Epidemiology Network Program (INCLEN). This Fall, Dr. Garrett will begin a full-time faculty appointment in the Department of Medicine.

Marcello Pagano is professor of statistical computing at the Harvard School of Public Health. His focus is on research and teaching, and he has just written *Principles of Biostatistics* (Duxbury Press; available from Stata Corporation) which is based on the introductory course in biostatistics at HSPH. He has long had an interest in the use of the computer in statistics and believes that it is finally revolutionizing the practice of statistics.

James L. Powell is professor of economics at Princeton University and visiting professor of economics at the University of California at Berkeley. Professor Powell is on the editorial boards of the *International Economic Review* and the *Journal of Econometrics*. Professor Powell's research interests center on the application of semiparametrics to microeconometric problems, especially those that involve censoring. His research has explored new uses of bootstrapping and nonparametric techniques. His most recent work has considered problems of random censoring.

Patrick Royston, D.Sc., graduated as a mathematician but has worked as a medical statistician ever since. He currently heads the Medical Statistics Unit at Hammersmith Hospital in London. He has a longstanding interest in solving the applied statistical problems that continually arise in medical research, and especially in providing software which implements new or important statistical techniques. He has published widely in statistical and medical journals.

an48 Updated CPS labor extracts available

Daniel Feenberg, National Bureau of Economic Research, feenberg@nber.harvard.edu

An updated version of the Current Population Survey Outgoing Rotation Group Annual Merge Files is now available on CD-ROM from the National Bureau of Economic Research. For those who purchased the original extract (Feenberg 1992), this update covers 15 years, 1979 through 1993, two more years of data than the original extract. For those unfamiliar with these data, the annual files contain extracts from the monthly Current Population Surveys.

Interviews are included for all respondents 16 years of age or older who are in an outgoing CPS rotation group during the year. This selection produces about 300,000 observations per year. These data can be, and have been, used in research on wage determination, union wage effects, inter-industry wage differentials, wage inequality, and employment discrimination. There is no copyright restriction on use of the data. New, unified hardcopy documentation is included with each CD.

The CD-ROM in ISO 9660 format is available for $100 postpaid ($115 overseas). Check, VISA, or MasterCard are accepted—no purchase orders, please. Be sure to specify CPS Labor Extracts 1993 version.

Publications Department
National Bureau of Economic Research
1050 Massachusetts Avenue
Cambridge, MA 02138
617-868-3900 (voice)
617-868-2742 (fax)

Questions should be directed to me at feenberg@nber.harvard.edu.

Frequently asked questions

What is the CPS? This is the monthly household survey conducted by the Bureau of Labor Statistics to measure labor force participation and employment. 50,000 to 60,000 households per month are surveyed.

Is this a panel survey? No.

What are the Outgoing Rotation Groups? Every household that enters the CPS is interviewed for 4 months, then ignored for 8 months, then interviewed again for 4 more months. Weekly hours/earning questions are asked only of households in their fourth and eighth months in the survey. These are the only households included in the CD-ROM.

Is this the same as the March Annual Demographic Survey? No. The income and demographic questions asked in March are not available on the CD-ROM.

How are the weights defined? The weights for all the persons in the sample sum to the total population 16 years of age or older. Only one-fourth of the sample is in an outgoing rotation group, so twelve months of data sums to three times the total population.

Is software included on the CD-ROM? No.

Can I use a spreadsheet to analyze these data? No. The files are much too large to fit in any spreadsheet. You will need a statistical package or database language to use these data.

What is the format of the files? The data are stored in Stata's binary (`.dta`) format for ease of use. The Stata files are compact and portable across operating systems and machine types. A year's data can be read from the CD-ROM in only a few minutes. If you use another statistical package, the program Stat/Transfer (available from StataCorp) will convert Stata files into many other formats.

Variable list

Interview month
Month in sample
State
Central city status
MSA/PMSA FIPS code
PMSA ranking
CMSA/MSA ranking
MSA/CMSA size
CMSA code
Metropolitan status code
Central city code
Household ID
Sex
Veteran
Highest grade attended
Whether completed highest grade
What was doing most of last week
How many hours last week, all jobs
Usually works >=35 hours at this job
Why not at least 35 hours last week
Class of worker
Usual hours
Paid by the hour
Earnings per hour
Usual earnings per week
Union member
Covered by a union contract
Enrolled as a student full/part time
Relationship to reference person
Age
Marital status
Race
Major activity last week
How many hours last week?
Reason <=35 hours last week
Why absent from work last week?
3-digit industry code (1980)
3-digit occupation code (1980)
Class of worker one
Usual hours
Paid by the hour
Union member
Ethnicity
Labor force status recode MODE
Full-time or part-time status
Detailed industry code
Detailed occupation code
Earnings eligibility flag
Class of worker two
Earnings per hour
Earnings per week
Final weight
Earnings weight for all races
Usual hours (I25a) allocation flag
Paid by hour (I25b) allocation flag
Earnings/hr (I25c) allocation flag
Usual Earn/hr (I25d) allocation flag

Reference

Feenberg, D. 1992. an23: CPS labor extracts available. *Stata Technical Bulletin* 9: 2–3.

an49	Stata listserver available

David W. Wormuth, Harvard Medical School, dwormuth@dsg.harvard.edu

Stata users who have access to Internet email can join an email service dedicated to using and enhancing Stata. The STATALIST is a listserver that distributes mail to a list of subscribers. To use the list, mail your question or observation to statalist@dsg.harvard.edu. The listserver then forwards the mail to all members of the STATALIST. There is no cost for the listserver. The only source of mail is from Stata users. In order to receive the STATALIST mailings, you need to subscribe.

How to subscribe

Send a message to listproc@dsg.harvard.edu. The body of the message should say

```
subscribe statalist your-first-name your-last-name
```

For example, a user named Joe Smith would send the following in his message:

```
subscribe statalist Joe Smith
```

I moderate the list only to the extent of filtering misdirected messages (for subscription or unsubscription usually). Otherwise all messages are re-sent as soon as they are received. Any questions can be sent to me at dwormuth@dsg.harvard.edu.

[We welcome the establishment of a listserver for Stata users. We want to clarify, however, that this listserver is run by independent Stata users. It is neither sponsored nor controlled by Stata Corporation or the Stata Technical Bulletin—Ed.]

sbe11	Direct Standardization

Tim McGuire, Stata Corporation, FAX 409-696-4601
Joel A. Harrison, Houston, Texas

Syntax

`dstndiz` *casevar popvar stratavars* [`if` *exp*] [`in` *range*], `by`(*groupvars*) [`base`(*# | "string"*)
`using`(*filename*) `saving`(*filename*) `print` `format`("*%fmt*") `level`(*#*)]

generates a summary measure of occurrence which can be used to compare prevalence, incidence, or mortality rates between populations which may differ with respect to certain characteristics (for example, age, gender, race.) These underlying differences may affect the crude prevalence, mortality, or incidence rates.

Options

`by`(*groupvars*) is not optional; it specifies the variables identifying the study populations. If `base()` is also specified, there must be only one variable in the `by()` group. If you do not have a variable for this option, you can generate one using something like `gen newvar=1` and then using `newvar` as the argument to this option.

`using()` or `base()` may be used to specify the standard population. The options can be specified individually or not at all, but not together. `using`(*filename*) specifies the name of a file containing the standard population. The standard population must contain the *popvar* and the *stratavars*. If `using()` is not specified, the standard population distribution will be obtained from the data. `base`(*# | "string"*) specifies the value of *groupvar* which identifies the standard population. If neither `base()` nor `using()` are specified, the default is to use the entire data set to determine the standard population.

`saving`(*filename*) saves the standard population distribution computed in a Stata data set that can be used in further analyses.

`print` outputs a tabular summary of the standard population distribution before outputting the study populations specified in the `by()` option.

`format("%`*fmt*`")` specifies the format in which to display the final summary table. The default is `%10.0g`.

`level`(*#*) specifies the significance level for the confidence intervals of the coefficients. The default is the current value of Stata's `$S_level` macro (initially set to 95.)

Description

A frequently recurring problem in epidemiology and other fields is the comparison of rates for some characteristic across different populations. These populations often differ with respect to factors associated with the characteristic under study; thus, the direct comparison of overall rates may be quite misleading.

The *direct method* of adjusting for differences among populations involves computing the overall rates that would result, if, instead of having different distributions, all populations were to have the same standard distribution. The standardized rate is defined as a weighted average of the stratum-specific rates, with the weights taken from the standard distribution.

Direct standardization may be applied only when the specific rates for a given population are available.

Example 1

It will be easiest to understand these commands if we start with a simple example. Suppose we have data (Rothman 1986, 42) on mortality rates for Sweden and Panama for the year 1962.

```
. use mortality
(1962 Mortality, Sweden & Panama)

. describe

Contains data from mortality.dta
  Obs:     6 (max=  5117)                  1962 Mortality, Sweden & Panama
 Vars:     4 (max=    99)
Width:    15 (max=   200)
  1. nation       str6   %9s               Nation
  2. age_cat      byte   %8.0g    age_lbl  Age Category
  3. pop          float  %9.0g             Population in Age Category
  4. deaths       float  %9.0g             Deaths in Age Category
Sorted by:

. list
```

```
      nation   age_cat        pop     deaths
 1.   Sweden    0 - 29    3145000       3523
 2.   Sweden   30 - 59    3057000      10928
 3.   Sweden       60+    1294000      59104
 4.   Panama    0 - 29     741000       3904
 5.   Panama   30 - 59     275000       1421
 6.   Panama       60+      59000       2456
```

When the total number of cases in the population is divided by the population, we obtain the *crude rate*:

```
. collapse pop deaths, sum(pop deaths) by(nation)
. list
      nation         pop      deaths
 1.   Panama     1075000        7781
 2.   Sweden     7496000       73555
. gen crude = deaths/pop
. list
      nation         pop      deaths      crude
 1.   Panama     1075000        7781   .0072381
 2.   Sweden     7496000       73555   .0098126
```

If we examine the total number of deaths in the two nations, it is striking that the total crude mortality rate in Sweden is higher than that of Panama. From the original data set, we see one possible explanation: Swedes are older than Panamanians. This makes it difficult to directly compare the mortality rates.

Direct standardization gives us a means of removing the distortion caused by the differing age distributions. The adjusted rate is defined as the weighted sum of the crude rates, where the weights are given by the standard distribution. Suppose we wish to standardize these mortality rates to the following age distribution:

```
. use 1962
(Std. Pop. Distribution)
. list
     age_cat        pop
 1.   0 - 29        .35
 2.  30 - 59        .35
 3.      60+         .3
. sort age_cat
. save 1962, replace
```

If we multiply the above weights for the age strata by the crude rate for the corresponding age category, the sum gives us the standardized rate.

```
. use mortality, replace
(1962 Mortality, Sweden & Panama)
. gen crude=deaths/pop
. drop pop
. sort age_cat
. merge age_cat using 1962
. list
      nation   age_cat     deaths      crude        pop    _merge
 1.   Panama    0 - 29       3904   .0052686        .35         3
 2.   Sweden    0 - 29       3523   .0011202        .35         3
 3.   Panama   30 - 59       1421   .0051673        .35         3
 4.   Sweden   30 - 59      10928   .0035747        .35         3
 5.   Sweden       60+      59104   .0456754         .3         3
 6.   Panama       60+       2456   .0416271         .3         3
. gen product = crude*pop
. egen adj_rate = sum(product), by(nation)
. drop _merge
. list, noobs nodisplay
  nation   age_cat     deaths      crude        pop    product   adj_rate
  Panama   30 - 59       1421   .0051673        .35   .0018085   .0161407
  Panama       60+       2456   .0416271         .3   .0124881   .0161407
  Panama    0 - 29       3904   .0052686        .35    .001844   .0161407
  Sweden       60+      59104   .0456754         .3   .0137026   .0153459
  Sweden   30 - 59      10928   .0035747        .35   .0012512   .0153459
  Sweden    0 - 29       3523   .0011202        .35   .0003921   .0153459
```

A comparison of the standardized rates indicates that the Swedes have a slightly lower mortality rate.

To perform the above analysis with `dstndiz`:

```
. use mortality, replace
(1962 Mortality, Sweden & Panama)
. dstndiz deaths pop age_cat, by(nation) using(1962.dta)
------------------------------------------------------------
-> nation= Panama
                          -----Unadjusted-----  Std.
                                Pop.  Stratum  Pop.
  Stratum       Pop.      Cases  Dist. Rate[s] Dst[P]  s*P
------------------------------------------------------------
   0 - 29     741000       3904  0.689 0.0053  0.350 0.0018
  30 - 59     275000       1421  0.256 0.0052  0.350 0.0018
      60+      59000       2456  0.055 0.0416  0.300 0.0125
------------------------------------------------------------
Totals:      1075000       7781    Adjusted Cases:  17351.2
                                        Crude Rate:  0.00724
                                     Adjusted Rate:  0.01614
                     95% Conf. Interval: [0.01614  0.01614]
------------------------------------------------------------
-> nation= Sweden
                          -----Unadjusted-----  Std.
                                Pop.  Stratum  Pop.
  Stratum       Pop.      Cases  Dist. Rate[s] Dst[P]  s*P
------------------------------------------------------------
   0 - 29    3145000       3523  0.420 0.0011  0.350 0.0004
  30 - 59    3057000      10928  0.408 0.0036  0.350 0.0013
      60+    1294000      59104  0.173 0.0457  0.300 0.0137
------------------------------------------------------------
Totals:      7496000      73555    Adjusted Cases: 115032.5
                                        Crude Rate:  0.00981
                                     Adjusted Rate:  0.01535
                     95% Conf. Interval: [0.01535  0.01535]

Summary of Study Populations:
   nation          N       Crude    Adj_Rate     CI_Left    CI_Right
   Panama    1075000    0.007238    0.016141    0.016139    0.016143
   Sweden    7496000    0.009813    0.015346    0.015346    0.015346
```

The summary table above allows us to make a quick inspection of the results within the study populations, and the detail tables give the behavior among the strata within the study populations.

Example 2

For a larger example, consider the following small data set containing blood pressure information on individuals in four cities. It is desired to use the entire data set as a standardized population. In this case, each observation represents an implied population of one; thus, a population variable will be generated to fit the syntax of the command. The standardization will be performed by city with the data stratified by age group.

```
. use hbp
. describe
Contains data from hbp.dta
  Obs:  1130 (max=  5116)
 Vars:     7 (max=    99)
Width:    17 (max=   200)
  1. id           str10  %10s              Record identification number
  2. city         byte   %8.0g
  3. year         int    %8.0g
  4. sex          byte   %8.0g    sexfmt
  5. age_grp      byte   %8.0g    agefmt
  6. race         byte   %8.0g    racefmt
  7. hbp          byte   %8.0g    yn       high blood pressure
Sorted by:
. gen pop=1
. dstndiz hbp pop age_grp, by(city)
(5 observations deleted due to missing values)
------------------------------------------------------------
```

```
-> city= 1
                        -----Unadjusted-----  Std.
                               Pop.  Stratum  Pop.
  Stratum       Pop.     Cases  Dist. Rate[s] Dst[P]  s*P
----------------------------------------------------------
  15 - 19         91         3  0.257 0.0330  0.220 0.0072
  20 - 24        103         2  0.291 0.0194  0.362 0.0070
  25 - 29         95         4  0.268 0.0421  0.237 0.0100
  30 - 34         65         4  0.184 0.0615  0.181 0.0112
----------------------------------------------------------
Totals:          354        13    Adjusted Cases:     12.5
                                      Crude Rate:  0.03672
                                   Adjusted Rate:  0.03541
                      95% Conf. Interval: [0.03329  0.03754]
----------------------------------------------------------
                                          output omitted
----------------------------------------------------------
-> city= 5
                        -----Unadjusted-----  Std.
                               Pop.  Stratum  Pop.
  Stratum       Pop.     Cases  Dist. Rate[s] Dst[P]  s*P
----------------------------------------------------------
  15 - 19         65         0  0.301 0.0000  0.220 0.0000
  20 - 24         84         1  0.389 0.0119  0.362 0.0043
  25 - 29         30         2  0.139 0.0667  0.237 0.0158
  30 - 34         37         1  0.171 0.0270  0.181 0.0049
----------------------------------------------------------
Totals:          216         4    Adjusted Cases:      5.4
                                      Crude Rate:  0.01852
                                   Adjusted Rate:  0.02503
                      95% Conf. Interval: [0.02063  0.02943]

Summary of Study Populations:
   city                N      Crude     Adj_Rate       Confidence Interval
   ------------------------------------------------------------------------
   1                 354   0.036723     0.035415    [  0.033293     0.037537]
   2                 340   0.023529     0.023044    [  0.021270     0.024819]
   3                 215   0.139535     0.124565    [  0.118198     0.130933]
   5                 216   0.018519     0.025030    [  0.020633     0.029427]
```

Example 3

Consider the data set of Example 2, with the desired strata being determined by age, sex, and race. We consider each year within city as a study population. In addition, the standard population (created from the data set) is printed at the beginning of the listing, the significance level is changed to 90, and the format of the summary table is modified.

```
. dstndiz hbp pop age race sex if year==1990 | year==1992, by(city year)
> format("%8.3f") print level(90)
---------------Standard Population---------------
                    Stratum       Pop.     Dist.
-------------------------------------------------
  15 - 19    Black   Female         35     0.077
  15 - 19    Black     Male         44     0.097
  15 - 19 Hispanic   Female          5     0.011
  15 - 19 Hispanic     Male         10     0.022
  15 - 19    White   Female          7     0.015
  15 - 19    White     Male          5     0.011
  20 - 24    Black   Female         43     0.095
  20 - 24    Black     Male         67     0.147
  20 - 24 Hispanic   Female         14     0.031
  20 - 24 Hispanic     Male         13     0.029
  20 - 24    White   Female          4     0.009
  20 - 24    White     Male         21     0.046
  25 - 29    Black   Female         17     0.037
  25 - 29    Black     Male         44     0.097
  25 - 29 Hispanic   Female          7     0.015
  25 - 29 Hispanic     Male         13     0.029
  25 - 29    White   Female          9     0.020
  25 - 29    White     Male         16     0.035
  30 - 34    Black   Female         16     0.035
  30 - 34    Black     Male         32     0.070
  30 - 34 Hispanic   Female          2     0.004
  30 - 34 Hispanic     Male          3     0.007
```

```
  30 - 34    White   Female          5     0.011
  30 - 34    White     Male         23     0.051
-------------------------------------------------
Total:           455
(669 observations deleted)
(6 observations deleted due to missing values)
-------------------------------------------------------------------------------
-> city year= 1 1990
                                          -----Unadjusted-----  Std.
                                                Pop.  Stratum  Pop.
                     Stratum       Pop.     Cases  Dist. Rate[s] Dst[P]  s*P
-------------------------------------------------------------------------------
  15 - 19    Black   Female          6         2  0.128 0.3333  0.077 0.0256
  15 - 19    Black     Male          6         0  0.128 0.0000  0.097 0.0000
  15 - 19 Hispanic     Male          1         0  0.021 0.0000  0.022 0.0000
  20 - 24    Black   Female          3         0  0.064 0.0000  0.095 0.0000
  20 - 24    Black     Male         11         0  0.234 0.0000  0.147 0.0000
  25 - 29    Black   Female          4         0  0.085 0.0000  0.037 0.0000
  25 - 29    Black     Male          6         1  0.128 0.1667  0.097 0.0161
  25 - 29 Hispanic   Female          2         0  0.043 0.0000  0.015 0.0000
  25 - 29    White   Female          1         0  0.021 0.0000  0.020 0.0000
  30 - 34    Black   Female          1         0  0.021 0.0000  0.035 0.0000
  30 - 34    Black     Male          6         0  0.128 0.0000  0.070 0.0000
-------------------------------------------------------------------------------
Totals:                             47         3    Adjusted Cases:      2.0
                                                        Crude Rate:  0.06383
                                                     Adjusted Rate:  0.04176
                                        90% Conf. Interval: [0.02582  0.05769]
-------------------------------------------------------------------------------
                                    output omitted
-------------------------------------------------------------------------------
Summary of Study Populations:
    city      year          N     Crude  Adj_Rate   CI_Left  CI_Right
       1      1990         47     0.064     0.042     0.026     0.058
       1      1992         56     0.018     0.009     0.004     0.013
       2      1990         64     0.047     0.045     0.024     0.066
       2      1992         67     0.030     0.014     0.007     0.022
       3      1990         69     0.159     0.088     0.071     0.106
       3      1992         37     0.189     0.046     0.036     0.057
       5      1990         46     0.043     0.022     0.007     0.037
       5      1992         69     0.014     0.051     0.051     0.051
```

Methods and Formulas

Standardized rate (S_R) is defined by

$$S_R = \frac{\sum_i (w_i R_i)}{\sum_i w_i}$$

(Rothman 1986, 44) where R_i is the stratum-specific rate in stratum i, and w_i is the weight for stratum i derived from the standard.

If n_i is the population of stratum i, the standard error $\mathrm{se}(S_R)$ in stratified sampling for proportions (ignoring the finite population correction) is

$$\mathrm{se}(S_R) = \sqrt{\sum_i \frac{{w_i}^2 R_i(1-R_i)}{n_i}}$$

(Cochran 1977, 108) from which the confidence intervals are calculated.

References

Cochran, W. 1977. *Sampling Techniques.* 3d ed. New York: John Wiley & Sons.

Fisher, L. and G. Van Belle. 1993. *Biostatistics: A Methodology for the Health Sciences*. New York: John Wiley & Sons.

Fleiss, J. 1981. *Statistical Methods for Rates and Proportions.* 2d ed. New York: John Wiley & Sons.

Forthofer, R. and E. Lee. (in press) *Biostatistics: Why, When, and How.* New York: Academic Press.

Pagano, M. and K. Gauvreau. 1993. *Principles of Biostatistics.* Belmont: Duxbury Press.

Rothman, K. 1986. *Modern Epidemiology*. Boston: Little, Brown, and Company.

sed8	Finding significant gaps in univariate distributions

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

One of the first steps in exploratory data analysis is to examine the univariate distributions of the variables being studied. Unusual discontinuities, or gaps, in these univariate distributions may indicate the need for further analysis. A large gap in the distribution may be evidence of nonrandom sampling or of data transcription errors. Gaps near the middle of the distribution are most troubling because they are least likely under random sampling.

This insert implements an idea of Wainer and Schacht (1978) for measuring gaps in univariate distributions. Consider a variable x, sorted in ascending order. Define the weighted gaps, g, as

$$g_i = \sqrt{\frac{(i-1)(x_i - x_{i-1})}{N - i + 1}}$$

where N is the number of observations on x. An approximate z-score can be defined as

$$z_i = g_i / \mu^*$$

where μ^* is the mean of the middle 50 percent of the distribution. z-scores above 2.25 indicate observations that merit further investigation. This cutoff can be adjusted upward if multiple gaps are analyzed.

`wgap` implements this technique. The syntax is

`wgap` *varlist* [`if` *exp*] [`in` *range*] , `wgap(`*prefix*`)` `z(`*prefix*`)`

`wgap` calculates and stores either or both of g and z, depending on which of the options is specified. At least one of the options, `wgap()` or `z()`, must be specified. If there is more than one variable in the *varlist*, the prefixes specified in the options are suffixed with a number: `1` for the first variable in the *varlist*, `2` for the second variable, and so on.

This procedure is somewhat robust to heavy-tailed distributions, but it can be confused by very heavy-tails (uniform distributions, for instance) or by numerous ties. We do not want to claim too much for this procedure, though. As its creators note, "The scheme presented here for the determination of the significance of gaps in a univariate data string is of only modest importance and usefulness" (Wainer and Schacht, p. 210). Nonetheless, `wgap` can help in finding areas of the data that are surprisingly sparse. As a result, `wgap` can help identify questionable data elements and can signal when the the data is actually a mixture of two or more distributions. But `wgap` can only raise these questions. Other techniques are needed to resolve them definitely. `wgap` also can be helpful in later stages of data analysis in choosing grouping categories and in finding breaks and cut-offs.

`wgap` was checked against the example in Wainer and Schacht (1978); it gives approximately the same answer; this is the best that can be expected given the apparent typos in Wainer and Schacht's Table 6.

Example

We use Stata's automobile data to illustrate `wgap`:

```
. use auto
(1978 Automobile Data)
. generate lprice = log(price)
. wgap lprice, z(zlprice)
. count if zlprice>2.25 & zlprice!=.
   4
. list make price if zlprice>2.25 & zlprice!=.
                    make     price
  1.       Chev. Impala      5705
  5.            Olds 98      8814
 20.      Buick Electra      7827
 67.          Audi 5000      9690
```

`wgap` flags four observations that have prices that are surprisingly higher than the next cheaper model. `wgap` is applied to the log of the price, so price changes are approximately percentage differences. The figure below graphs the z-scores against the price (on a log scale), highlighting the four suspect observations. Note that the four observations are near the middle of the price distribution and that the distribution is somewhat sparse in the neighborhood of these observations.

Figure

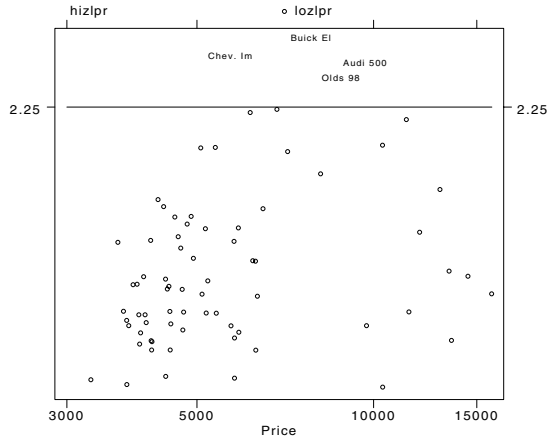

Figure 1: z-scores of log price

References

Wainer, H. and S. Schacht. 1978. Gapping. *Psychometrika* 43: 203–212.

sg26	Using fractional polynomials to model curved regression relationships

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011)-44-81-740-3119
Douglas G Altman, Imperial Cancer Research Fund, London, FAX (011)-44-71-269-3429

Fractional polynomial (FP) regression models (Royston and Altman, 1994) are intermediate between polynomial and nonlinear models. The aim in using FP functions in regression is to keep the advantages of conventional polynomials, while eliminating (most of) the disadvantages. Put briefly, FP functions are similar to conventional polynomials in that they include powers of X, but non-integer and negative powers are also allowed. FP models usually give a better fit than conventional polynomials of the same degree, and even than those of higher degree.

As the technique is new, we first describe it in some detail and give examples of its use. A description of the software is deferred to the section *Program fp*. Here we only outline the two possible syntaxes of `fp`, which are as follows:

`fp` *yvar* [*nvar*] *xvar* [`if` *exp*] [`in` *range*] [*weight*] [`,` *major_options minor_options*]

`fp` [`,` `summary` `comparison` `estimates`]

Background and motivation

Continuous variables are widely used in regression models. In most applications the variables are entered untransformed and modeled as having straight-line relationships with the outcome variable. An alternative strategy, common in epidemiology for example, is to group the continuous variables into c categories which are then modeled by fitting $c-1$ binary "dummy" variables. While these approaches are often effective, one frequently wishes to retain the full information in the measurements but not to assume a straight-line relationship. This is especially so in situations where one wishes to use a model to make predictions for future individuals, as is the case when constructing clinical reference intervals. Until recently the standard approach to modeling curved relationships was polynomial regression. After fitting a linear term in the variable X, we fit a quadratic curve, then cubic, and so on, by adding sequentially to the model terms in X^2, X^3, X^4, etc. until no worthwhile (or significant) improvement in fit is achieved.

Polynomial models are both parametric and global, in the sense that all the data are used to derive all the fitted values. Recent developments have introduced various local nonparametric smoothing techniques that allow curved relationships to be modeled. These enable the relationship of Y with X to be visualized and may suggest an appropriate functional form. Local regression models include regression splines, smoothing splines, and kernel methods. The Stata program `ksm` can fit some models of this type, including the well-known *lowess*. Generalized additive models (Hastie and Tibshirani 1990) offer even greater scope, but are not yet implemented in Stata.

The main advantages of local regression models are their flexibility and their ability to reveal the "true" curve shape. They have some disadvantages too: they do not provide a simple expression for the model (so that individual predictions are not obtained simply), the curves are not necessarily smooth, and statistical inference (significance tests for inclusion of model terms, etc.) is not properly worked out.

Disadvantages of conventional polynomials

Despite its ubiquity, polynomial regression has long been recognized to have some serious weaknesses, notably lack of flexibility (in low order models such as quadratics), a propensity to produce artifacts (waviness and "end-effects", making them useless for extrapolation) in higher order fitted curves, and an inability to model relations which have asymptotes.

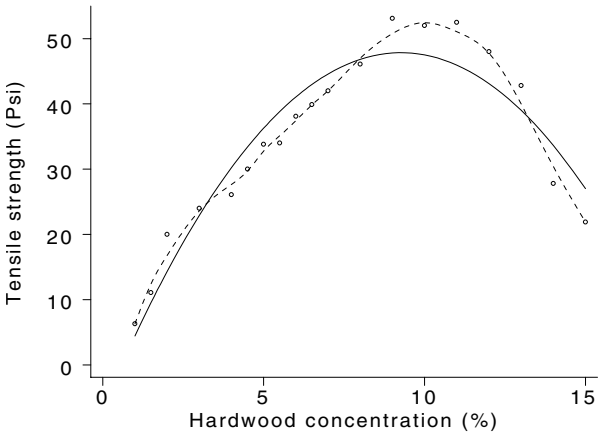

Figure 1: Hardwood concentration in pulp and tensile strength of paper (Psi) (Montgomery and Peck, 1992)

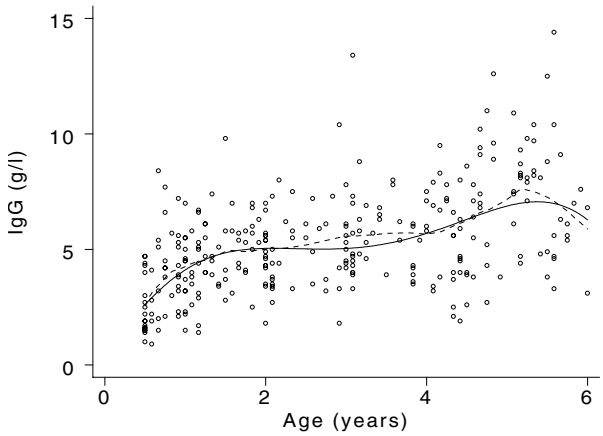

Figure 2: IgG and age for 298 children

Figure 1 shows some data relating the tensile strength of paper to the proportion of hardwood in the pulp. These data were used by Montgomery and Peck (1992) to illustrate the technique of quadratic regression. The data show clear asymmetry, however, and the quadratic curve (solid line) is not a good fit. For comparison, also shown is a smooth curve (dashed line) obtained from `ksm`, using `bwidth(0.3)`. The quadratic model does not reflect the asymmetric curvature. Among conventional polynomials, a quintic curve is needed to get a satisfactory fit to the data.

Figure 2 shows data relating serum immunoglobulin IgG level in children to their age (Isaacs et al. 1983). Also shown are the quartic curve (solid line) and a curve from `ksm` (dashed line), again using `bwidth(0.3)`. The quartic curve shows artifactual waves, and the apparent downturn at the upper ages (which appears in both fits) is clinically implausible.

For each of these two data sets, the fit of the nonparametric curves derived from `ksm` is similar to that of conventional high-order polynomials. We show how to find simple parametric models that also fit the data sets well.

Fractional polynomials

FP functions can be used with any generalized linear model and with Cox proportional hazards regression models for survival data. Examples are normal errors regression (multiple linear regression), (multiple) logistic regression and log-linear modeling of contingency table data which have ordered categories. In all of these a response variable Y is regressed on a single covariate X, or on several covariates $X_1, \ldots, X_k$. Note that, for reasons explained below, we require $X > 0$. We initially concentrate on the single covariate case, and mention multiple covariates later. Fuller details and further examples are given in Royston and Altman (1994).

We define the degree of an FP model as the number of terms in powers of X in the model and denote it m. Thus, for example, $Y = b_0 + b_1 X^{-1}$ has degree $m = 1$ and $Y = b_0 + b_1 X^{-1} + b_2 X^2$ has $m = 2$. We call the powers in the FP model p_1, p_2, etc., and denote the vector of powers as $\mathbf{p}$. In the two examples just given we have $p = -1$ and $\mathbf{p} = (-1, 2)$ respectively. Models with $m = 1$ or $m = 2$ provide a wide range of curve shapes—some examples are given in Figure 3. It is uncommon to need models with $m > 2$, and so `fp` concentrates on models with $m = 1$ or $m = 2$. However, `fp` can also be used to fit models with $m > 2$; we show these later.

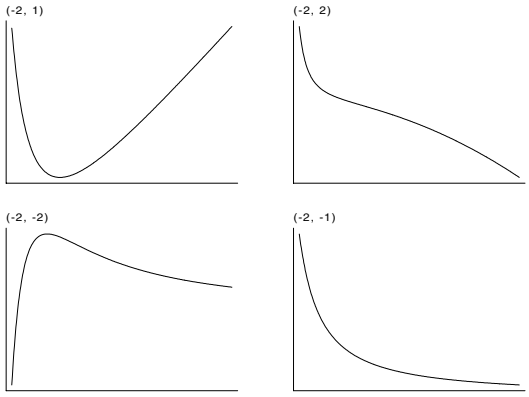

Figure 3: Examples of fractional polynomials with p=$(-2,1)$, $(-2,2)$, $(-2,-2)$ and $(-2,-1)$

Models are chosen by including m powers from a predefined set $\mathcal{P}$. We suggest using

$$\mathcal{P} = \{-2, -1, -\tfrac{1}{2}, 0, \tfrac{1}{2}, 1, 2, 3\},$$

which is rich enough to cover many practical cases adequately. When $m > 3$ we add to $\mathcal{P}$ integer powers up to m. Note that the power 0 corresponds to $\ln X$. We use round bracket notation for powers of X, such as $X^{(p)}$, to signify the Box–Tidwell transformation

$$X^{(p)} = \begin{cases} X^p, & \text{if } p \neq 0; \\ \ln(X), & \text{if } p = 0, \end{cases}$$

not the more familiar Box–Cox transformation, $(X^p - 1)/p$.

Certain combinations of powers from the specified $\mathcal{P}$ correspond to quadratics (that is, functions of the form $b_0+b_1X^p+b_2X^{2p}$) in X^{-1}, $X^{-\frac{1}{2}}$, $X^{\frac{1}{2}}$ and X, as well as a cubic in X and all degree-m conventional polynomials.

An interesting feature of FP models with $m > 1$ is that duplication of powers, such as $\mathbf{p} = (1, 1)$, does not result in models with reduced degree. Rather, by finding the mathematical limit of the expression

$$b_0 + b_1X^{(p_1)} + b_2X^{(p_2)}$$

as p_2 tends to p_1, the limiting FP model turns out to be

$$b_0 + b_1^*X^{(p_1)} + b_2^*X^{(p_1)} \ln X,$$

a three-parameter family of curves (i.e. retaining degree $m = 2$, rather than $m = 1$). For example, $\mathbf{p} = (1, 1)$ defines the function $b_0 + b_1X + b_2X \ln X$, and $\mathbf{p} = (0, 0)$ leads to a quadratic in $\ln X$. A similar result applies for more than two repeated powers (see Formulae at end).

Model choice (with fixed m)

To fit a model with $m = 1$ we simply take each power from $\mathcal{P}$ in turn. For $m = 2$ we fit models corresponding to each possible pair of powers from $\mathcal{P}$. The deviance (-2 times log-likelihood) for each model is noted (further information about deviances is given in the Technical Note below). The $\mathbf{p}$ associated with the model with the lowest deviance we denote $\tilde{\mathbf{p}}$.

Often, for a given m, several values of $\mathbf{p}$ correspond to models with similar deviances. Models whose deviance is "much higher" than that of the $\tilde{\mathbf{p}}$ model are discarded. In general, we suggest deviance thresholds based on the 90th centile of χ^2 with m degrees of freedom (DF); see Royston and Altman (1994) for explanation. The thresholds are thus 2.7 (for $m = 1$) and 4.6 (for $m = 2$). For normal-errors models, we use appropriate critical points of the F distribution instead of those of χ^2.

There may be several candidate models which are indistinguishable according to these criteria. We do not recommend automatic choice of the $\tilde{\mathbf{p}}$ model. Rather, the "best" model is chosen from the candidates by criteria which include the visual appearance of the fitted curve, the science of the problem and the plausibility of the curve when extrapolated.

As with conventional univariate polynomial regression, the number of terms in the model is determined by increasing m until no worthwhile improvement in fit occurs. We would not usually increase m unless the deviance was reduced by more than 4.6, this value being the 90th centile of χ^2 with 2 DF.

As we are interested in modeling curvature it is convenient to use the deviance for the straight line model (i.e. with $m = 1,\ p = 1$) as a baseline for calculating the deviances of other models. We can convert deviance to *gain*, G, defined as the deviance for the straight-line model minus the deviance for the FP model. A bigger gain means a better fit. `fp` gives deviances and gains for various models, as illustrated in the following examples. The P-values for differences between models reported by `fp` are obtained from differences in deviances using the χ^2 or F distributions; generally they are conservative (slightly too large compared with the true Type I error probability).

Degrees of freedom of an FP model

Since each estimated power, p, must belong to the set $\mathcal{P}$, it does not use up a full DF, as it would do if p was allowed to be any real number. However, for practical purposes we assume (as a worst case) that it does consume 1 DF. So, apart from the constant term, β_0, an $m = 1$ model uses 2 DF and an $m = 2$ model uses 4 DF; in general, an FP of degree m uses $2m$ DF. One might ask whether there are models with odd DF (3, 5, etc.). For each m, there is in fact a submodel of the full degree-m model which has $2m - 1$ DF. For $m = 1$ it is the straight line, $\beta_0 + \beta_1 X$. For $m > 1$, it comprises a linear term, $\beta_1 X$, and a full FP component of degree $m - 1$; for $m = 2$ the 3 DF family is of the form $\beta_0 + \beta_1 X + \beta_2 X^{(p)}$. The `fp` program allows you to fit these odd-DF models (see *Program fp* below).

Example 1: Fetal mandible length

Figure 4 shows 158 observations of fetal mandible length in relation to gestational age. Nine further measurements with $X > 28$ were excluded from the analysis as the clinician considered that they were unreliable. It is clear that both the mean and SD of mandible length increase as age increases, a typical pattern with measurements of fetal size. Log transformation of mandible length makes the SD approximately constant (the `boxcox` command will sometimes find a successful variance-stabilizing transformation of this kind). Thus in the analysis Y is log mandible length and X is gestational age.

```
. use mandible
. generate lmand = log(mand)
. fp lmand gest, comparison estimates
MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) for Y = lmand, X = gest.
(*) Base model  Linear   Quadratic     Cubic     BoxTid      df(2)      df(4)
------------------------------------------------------------------------------
p       --          1        1, 2     1, 2, 3      1, 1        -1      -2,   1
D   113.358   -215.671   -285.916   -294.477   -289.352   -293.178   -293.736
G                0.000     70.245     78.806     73.681     77.506     78.065
MODEL COMPARISONS  (Note: P-values for df(2) and df(4) are conservative.)
Model comparison  DF  Dev. diff.  P    | Model comparison  DF  Dev. diff.  P
-----------------------------------------+-----------------------------------
Linear vs base     1   329.030  0.000 | Quad vs linear     1    70.245  0.000
Cubic vs quad      1     8.561  0.004 | BoxTid vs linear   1    73.681  0.000
                                       |
df(2)  vs base     2   406.536  0.000 | df(2)  vs linear   1    77.506  0.000
                                       |
df(4)  vs base     4   407.095  0.000 | df(4)  vs linear   3    78.065  0.000
df(4)  vs quad     2     7.820  0.023 | df(4)  vs df(2)    2     0.558  0.763
(*) Base model = [none] (158 obs.)
```

```
  Source |       SS       df       MS                  Number of obs =     158
---------+------------------------------                F(  2,   155) =  941.75
   Model |  17.5156355     2  8.75781773                Prob > F      =  0.0000
Residual |  1.44141936   155   .00929948                R-square      =  0.9240
---------+------------------------------                Adj R-square  =  0.9230
   Total |  18.9570548   157  .120745572                Root MSE      =  .09643
------------------------------------------------------------------------------
   lmand |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     X_1 |  -191.3771   19.22988     -9.952   0.000      -229.3636   -153.3907
     X_2 |   .0257281   .0059736      4.307   0.000        .013928    .0375283
   _cons |   3.129596   .1721908     18.175   0.000       2.789453     3.46974
------------------------------------------------------------------------------
Fractional power(s) of X used: -2 1
```

`fp` produces three types of output, two of which arise from using options `comparison` and `estimates`. The default output is titled MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) and summarizes the fractional powers, deviances and gains for the seven different models that `fp` fits to the data. The `Base model` comprises the variables specified in the `base()` option (see the description of *Program fp* below). If no variables are specified, the base model is just the constant. All the other models include the base variables. `Linear`, `Quadratic` and `Cubic` are conventional polynomials in X (here `gest`). `BoxTid` is the Box–Tidwell function (itself a fractional polynomial with $\mathbf{p} = (1, 1)$), namely

$$E(Y) = \beta_0 + \beta_1 X + \beta_2 X \ln X$$

`df(2)` and `df(4)` refer to fractional polynomial models with 2 DF ($m = 1$) and with 4 DF ($m = 2$) respectively.

The second type of output (`comparison`), titled MODEL COMPARISONS, performs significance tests of the fit of various pairs of models. From the point of view of FP modeling, the most important comparisons are `df(2) vs linear`, which tells us whether an $m = 1$ model fits significantly better than a straight line, and `df(4) vs df(2)`, which tests whether $m = 2$ is significantly better than $m = 1$. It may be of interest to compare the $m = 2$ model with a quadratic; this test is also given. The comparison `BoxTid vs linear` gives a test of nonlinearity that is sensitive to non-monotonic relationships between Y and X, that is, whether Y has a maximum or minimum within the range of the data. `df(2) vs linear`, also a test of nonlinearity, is likely to be insensitive to non-monotonicity since all $m = 1$ models are monotonic.

The third type of output (`estimates`) comprises the regression results for the best-fitting FP model.

The interpretation of the output for the mandible example is as follows. There is minimal reduction in deviance (0.558, see entry `df(4) vs df(2)`) for $m = 2$ (`df(4)`) compared with $m = 1$ (`df(2)`), so we prefer the simpler model ($p = -1$), with Y linear in $1/X$. Figure 5 shows that this model is an excellent fit to the data. Note that we need a cubic curve to get as good a fit with a conventional polynomial. Figure 5 shows, however, that while the two fits are indistinguishable for the 158 points analyzed, the FP model behaves sensibly beyond the range of X analyzed, passing through the cloud of extra points that had been omitted, whereas the cubic shoots upwards rapidly and implausibly.

The state of the data revealed by `describe` after using the `fp` command is shown below.

```
Contains data from mandible.dta
  Obs:   158 (max= 32766)
 Vars:     6 (max=    99)
Width:    24 (max=   200)
  1. mandible     float  %9.0g            Mandible (mm)
  2. gest         float  %9.0g            Length of gestation (weeks)
  3. lmand        float  %9.0g
  4. X            float  %9.0g            gest
  5. X_1          float  %9.0g            X^-2
  6. X_2          float  %9.0g            X
```

Note also that `fp` has quietly added three new variables to the data: `X`, `X_1` and `X_2`. `X` is a copy of the X variable, here `gest`. `X_1` and `X_2` are `X` raised to the best $m = 2$ fractional powers. Stata needs these variables in order to "replay" the best FP model and to produce graphs of the fit (see the descriptions of programs `fp` and `fpgraph` below). If variables `X`, `X_1`, `X_2`, etc. exist before `fp` is used, they are overwritten without warning. `X` raised to the best $m = 1$ fractional power is only saved if the models are restricted to $m = 1$ by using the `df(2)` option.

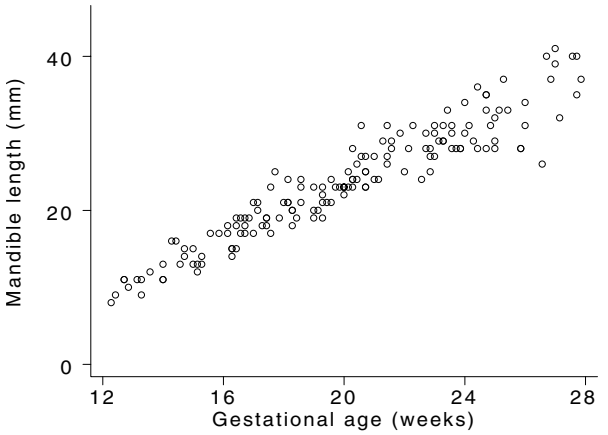

Figure 4: Fetal mandible length and gestational age (n=158) (Chitty et al. 1993)

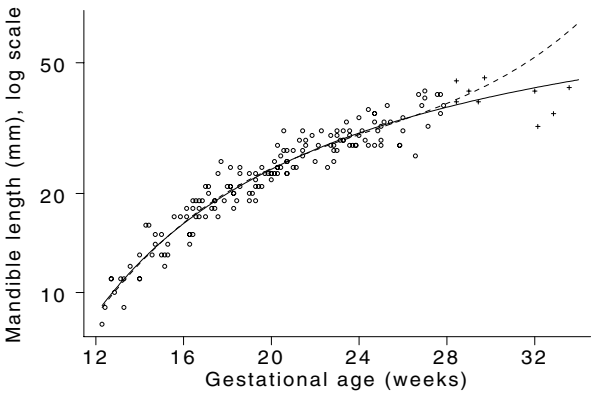

Figure 5: Mandible length data, showing the cubic spline (dashed line) and the chosen FP model (solid line). Extra 9 points excluded from analysis are shown as +.

Example 2: Serum IgG in 298 children

The IgG data were shown in Figure 2. Although the variance does not change markedly with age, taking the square root of IgG improves the normality of the data and stabilizes the variance. Thus Y is $\sqrt{\text{IgG}}$ and X is age.

```
. generate rigg = sqrt(igg)
. fp rigg age
MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) for Y = rigg, X = age.
(*) Base model  Linear   Quadratic     Cubic      BoxTid       df(2)       df(4)
-------------------------------------------------------------------------------
p       --         1        1, 2     1, 2, 3      1, 1           0      -2,   2
D   427.539   337.561     333.884     327.687   331.294     327.436     319.448
G               0.000       3.677       9.874     6.267      10.125      18.113
```

The best FP model for $m = 1$ is $p = 0$ giving a gain of 10.13, but the best FP model with $m = 2$, $\tilde{\mathbf{p}} = (-2, 2)$, has a gain of 18.11 and so is a significantly better fit.

As the output shows, the ordinary cubic is no better than the best FP model with $m = 1$. Among conventional polynomials we need a quartic to get a fit whose gain (19.87) is similar to that of the best $m = 2$ model. However, the quartic is not acceptable because of "end-effects" (see Figure 2). The FP model is clearly preferable; it is shown in Figure 6.

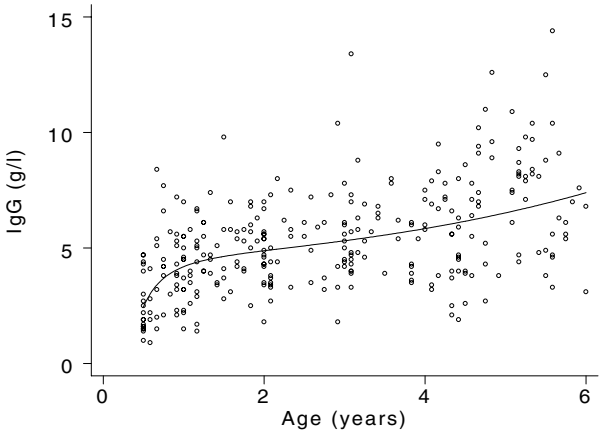

Figure 6: FP model with $\tilde{p}$=(−2,2) fitted to IgG data

Example 3: Automobile data

Using Stata's `auto.dta` dataset, suppose we want to model a car's economy (`mpg`) in terms of its engine size (`displ`). The scatterplot in Figure 7 indicates a curved relationship. We try fitting FP models:

```
. fp mpg displ
MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) for Y = mpg, X = displ.
(*) Base model  Linear   Quadratic     Cubic     BoxTid      df(2)      df(4)
-------------------------------------------------------------------------------
p       --         1        1, 2     1, 2, 3      1, 1         -2      -2,   3
D   468.789   417.801     409.292     399.932     406.687    400.592    397.971
G               0.000       8.510      17.869      11.114     17.209     19.830
```

The results show that the best single power (i.e. the best model with $m = 1$) for `displ` is –2. The deviance is 400.59. The best $m = 2$ model has powers (–2, 3) and a deviance of 397.97, which is only 2.62 lower and not statistically significantly better. (Note that a cubic polynomial, with a deviance of 399.93, is required to give a fit as good as that of the best $m = 1$ model.) Figure 8 shows the fitted model.

Preliminary stepwise regression analysis indicated that `foreign` and `weight` are also significant predictors of `mpg`, so we include them in the model as base variables:

```
. fp mpg displ, base(foreign weight)
MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) for Y = mpg, X = displ.
(*) Base model  Linear   Quadratic     Cubic     BoxTid      df(2)      df(4)
-------------------------------------------------------------------------------
p       --         1        1, 2     1, 2, 3      1, 1         -2      -2,  -1
D   388.366   388.327     381.517     377.588     379.854    376.525    374.622
G               0.000       6.810      10.739       8.474     11.803     13.705
```

The same $m = 1$ power (–2) is obtained. The deviance is now 376.53, a large reduction due to including `foreign` and `weight`. The previous deviance was 400.59, so the reduction is 24.06 on 2 DF and is significant at the 0.001 level using a χ^2 distribution with 2 DF (which gives an approximate test). The $m = 2$ (or `df(4)`) model is virtually no better than the $m = 1$ model. The deviance of the $m = 1$ model is lower than that of a cubic polynomial (377.59), which has two more terms. The $m = 1$ model has an asymptote and is therefore likely to extrapolate better to very large engine sizes than a cubic polynomial, which does not have an asymptote. However, we can't be too confident about how well it will extrapolate to small engine sizes, where the FP function is climbing steeply; for example, the estimated consumption for X = 60 cu in (about 980 cc) is 46 mpg, which may be somewhat too high.

To find *simultaneously* the best FP functions for `mpg` for several covariates, such as `displ`, `weight`, `gratio`, etc., is beyond the scope of `fp`. However, program `swfp`, to be released later, will have the necessary capability.

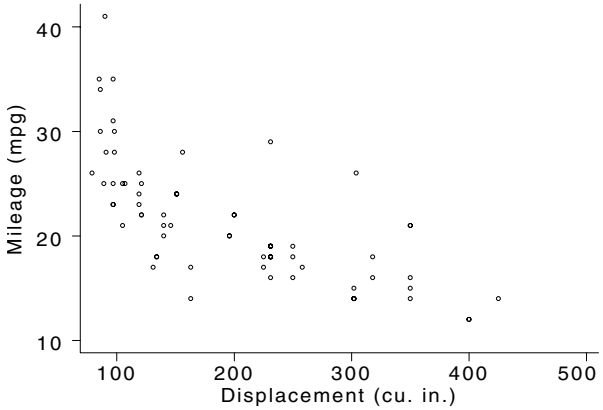

Figure 7: Relation between economy and engine size

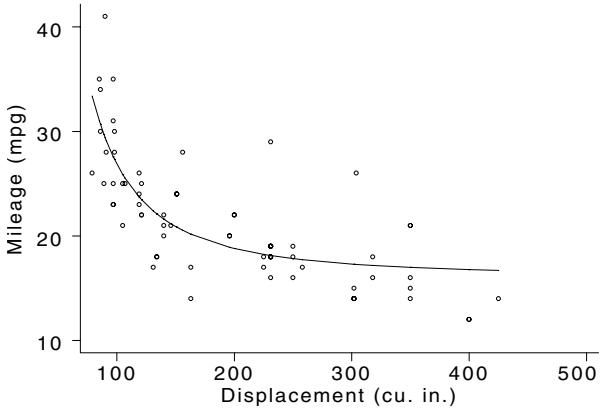

Figure 8: Data of Figure 7 with fitted FP model with $p=-2$.

Example 4: logistic regression—in vitro fertilization

The outcome variable (Y) is whether or not a woman undergoing in vitro fertilization becomes pregnant (coded 0 = no, 1 = yes) in relation to plasma oestradiol (E2) level (X) (Afnan et al. 1993). We cannot assess the shape of the relation by simply plotting the raw data, but we can use a *lowess* smooth of Y, with `bwidth(0.8)`, as in Figure 9.

We use `fp` with logistic regression to model the relationship, using the `comparison` and `estimates` options to get the complete output:

```
. fp outcome e2, cmd(logit) comparison estimates
MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) for Y = outcome, X = e2.
(*) Base model  Linear   Quadratic     Cubic     BoxTid      df(2)      df(4)
-------------------------------------------------------------------------------
p       --          1        1, 2     1, 2, 3      1, 1        -2      -1,  -1
D   364.270    364.268     359.979     357.378     359.097    362.016    357.148
G                0.000       4.289       6.890       5.171      2.252      7.119
MODEL COMPARISONS  (Note: P-values for df(2) and df(4) are conservative.)
Model comparison  DF  Dev. diff.  P    | Model comparison  DF  Dev. diff.  P
-----------------------------------------+-----------------------------------
Linear vs base     1     0.002  0.962 | Quad vs linear     1     4.289  0.038
Cubic vs quad      1     2.601  0.107 | BoxTid vs linear   1     5.171  0.023
                                       |
df(2)  vs base     2     2.255  0.324 | df(2)  vs linear   1     2.252  0.133
                                       |
df(4)  vs base     4     7.122  0.130 | df(4)  vs linear   3     7.119  0.068
df(4)  vs quad     2     2.831  0.243 | df(4)  vs df(2)    2     4.867  0.088
(*) Base model = [none] (268 obs.)
Logit Estimates                                       Number of obs =    268
                                                      chi2(2)       =   7.12
                                                      Prob > chi2   = 0.0284
Log Likelihood = -178.57425                           Pseudo R2     = 0.0196
------------------------------------------------------------------------------
 outcome |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     X_1 |  -205075.2   87935.59     -2.332   0.020      -377425.8   -32724.64
     X_2 |   26550.07   11479.38      2.313   0.021       4050.903    49049.24
   _cons |  -4.385136   1.925117     -2.278   0.023      -8.158296   -.6119765
------------------------------------------------------------------------------
Fractional power(s) of X used: -1 -1
```

The best model with $m = 1$, with $\tilde{p} = -2$, gives a non-significant gain of $G = 2.25$. However, the best model with $m = 2$, with $\tilde{\mathbf{p}} = (-1, -1)$, gives $G = 7.12$. Thus we choose the latter FP model, corresponding to $b_0 + b_1 X^{-1} + b_2 X^{-1} \ln X$, where $b_0 = -4.385$, $b_1 = -205075$, $b_2 = 26550$ (see above).

Figure 10 shows that this model is an excellent fit to the data as judged by its similarity to the curve from `ksm`. As noted above, we would not automatically take the model corresponding to $\tilde{\mathbf{p}}$. In this case $\mathbf{p} = (0, 0)$ is almost as good a fit, with a gain of $G = 6.60$. (This function is a quadratic in $\ln X$.) However, $\mathbf{p} = (1, 2)$, a conventional quadratic in X, although giving $G = 4.29$, is a poor fit, as Figure 10 shows.

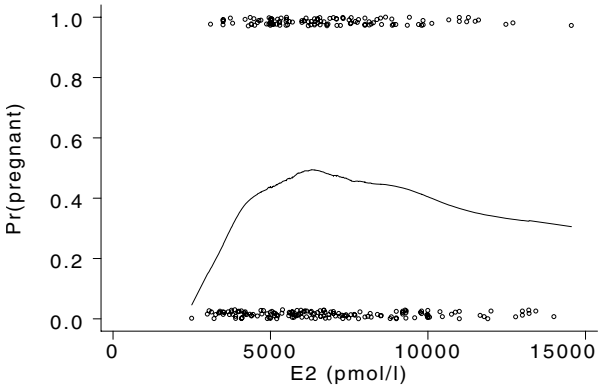

Figure 9: Relation between pregnancy and plasma E2 showing *lowess* smooth

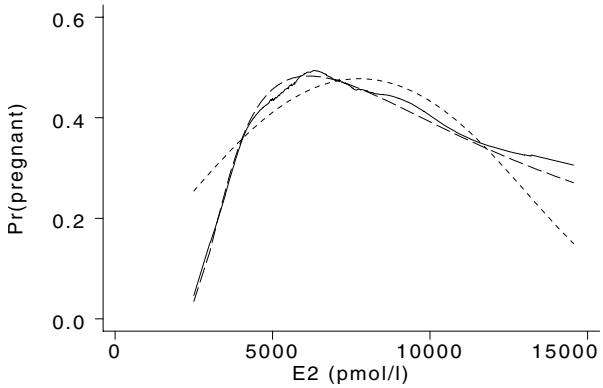

Figure 10: Relation between pregnancy and plasma E2 with *lowess* smooth (solid line), conventional quadratic (short dashes), and best FP model with m=2 (long dashes)

Program fp

The syntax for fp is

fp *yvar* [*nvar*] *xvar* [if *exp*] [in *range*] [*weight*] [, *major_options minor_options*]
fp [, summary comparison estimates]

The *major_options* (most used options), in alphabetic order, are

base(*basevars*) cmd(*regression_cmd*) comparison estimates df(#)
fixpowers(*fixlist*) log powers(*powlist*|none) *regression_cmd_options*

The *minor_options* (less used options), in alphabetic order, are

addpowers(*addlist*) continuous(*contvars*) devthr(#) expx(#|sd) fast gplot(#)
name(*newxvar*) origin(#) norepeat saving(*graph_file* [, replace]) zero

The first form of the fp command performs the analysis. The second form is used to (re)display results already obtained by using the first form.

fp fits fractional polynomial (FP) models in *xvar* to *yvar*. A wide variety of model types is supported, including normal-errors regression (the default), logistic, Cox, Poisson, and so on (see the cmd() option for further details).

nvar is required if "blocked" logit (blogit) or probit models are to be fitted; then *yvar* is the number of successes out of *nvar* trials.

The degrees (m) of FP which fp fits are 1 ($m = 1$) or 2 ($m = 2$); to fit models with $m > 2$, use either the fixpowers() option (see below) or the fpx command (see help fpx). The fractional powers of *xvar* are supplied by the program (but may be altered by using the powers() and addpowers() options). "Base model" variables are defined as variables which are always included in the model. By default, the base model consists only of the regression constant; other variables may be added by using the base() option. The model actually fitted, therefore, consists of the base variables and the terms representing FPs in *xvar*. The latter variables are calculated automatically.

fp also fits linear, quadratic and cubic polynomials, and the Box–Tidwell (BoxTid) model $\beta_0 + \beta_1 X + \beta_2 X \ln X$. If BoxTid fits significantly better than a straight line, there is evidence of curvature in the relation between *yvar* and *xvar*.

fp reports the deviances for all models it fits (see the Technical Note below for definitions and formulae).

If the comparison option is used, fp compares the fits of various pairs of models using P-values, which are approximate (typically conservative) when FP models are compared with other models. The most important comparisons are between $m = 1$ and linear, and between $m = 2$ and $m = 1$. If the deviance of a cubic polynomial (for which $m = 3$) is much lower than that of the best $m = 2$ model, you will need an $m > 2$ model in order to fit the data adequately.

fp also calculates the regression analysis of the best-fitting model. The output may be (re)displayed by repeating *regression_cmd*, or by using the estimates option, which gives slightly more information. You may also use predict, test, etc. after fp; the results will depend on the *regression_cmd* you used.

Some options

We comment here on some of the major options; full details of all options can be found in the help file fp.hlp.

base() defines the variables in the base model. The default is none (depending on context, the base model is usually just the constant term _cons).

cmd() determines the type of regression to be used. The default, equivalent to specifying cmd(regress), is ordinary regression. Other possibilities that are known to work include anova, logit, probit, blogit, bprobit, cox, mlogit, poisson, qreg, and glmr. In addition, fp is written in such a way that you can supply any regression command to cmd(). Provided that the command in question stores the log likelihood for the fitted model in _result(2), fp will work correctly with it.

df(#) specifies that the degrees of freedom of the highest-degree FP model to be fitted are #, and that comparison is to be made between models with df = # and those with one fewer df. For example, df(3) compares $m = 2$ models in which one power is always 1 with all $m = 1$ models. df(1) is equivalent to specifying powers(1). If df() is not specified (the default), the best $m = 2$ model is found and its fit is compared with that of the best $m = 1$ model.

`fixpowers(`*fixlist*`)` includes fractional power(s) of *xvar* corresponding to *fixlist* in every FP model fitted. This increases the degree of the FPs by the number of values in *fixlist*. For example, with `fixpowers(0,1)`, all FP models will include terms in ln(*xvar*) and *xvar*; the FP degree will then be 3 for $m = 1$ models and 4 for $m = 2$ models. In the reported results, the presence of `fixpowers()` in the models is indicated by a plus (+) symbol.

`log` displays deviance differences from the base model and (for normal errors regression) residual standard deviations for each FP model fitted. This option is valuable for seeing whether there are other models that fit almost as well as the best $m = 1$ and $m = 2$ models. See also the `gplot()` option, which plots the deviances for all the models against the first power.

`powers()` is the set of fractional polynomial powers to be used. The default set is *powlist* = {–2, –1, –0.5, 0, 0.5, 1, 2, 3} (0 giving log). Note that specifying `powers(none)` in conjunction with `fixpowers()` (see above) prevents `fp` from searching for the combination of powers with the lowest deviance, and fits only the model defined by *fixlist*. This important combination of options allows you to inspect particular models; you can plot the fit by using the `fpgraph` command (see below).

Choice of origin for X

As we have already noted, to fit an FP model we need all the values of *xvar* to be positive. In many cases *xvar* will contain zeroes, usually as a result of rounding or of *xvar* being a discrete measurement (such as the number of cigarettes smoked per day or the number of children in a family), or even negative values, such as a difference between two quantities. A simple strategy, which usually works well, is to subtract the minimum of *xvar* from *xvar* and then to add the rounding or counting interval before applying FP analysis. For example, if the lowest value of *xvar* is zero and *xvar* is measured to the nearest 0.1 units, we add 0.1 to all the *xvar* values (not just the zero values!) first. `fp` does this automatically: it determines the rounding interval as the minimum positive distance between successive ordered values of *xvar* and adds this quantity minus the minimum of *xvar* (often 0, but it could be negative) to *xvar* before fitting FP models. The number actually added is stored in `$S_23`.

Sometimes it may be necessary to use a more sophisticated approach. `origin(#)` transforms *xvar* so that its maximum is 1 and its minimum is #, where $0 < \# < 1$. Specifically, we replace *xvar* by $(xvar - \zeta)/(xmax - \zeta)$, where *xmin* and *xmax* are the minimum and maximum respectively of *xvar* and where $\zeta = (xmin - \# \times xmax)/(1 - \#)$. This transformation is useful if *xvar* contains negative values, or if its range is too narrow for effective FP modeling. # must be between 0 and 1 exclusive. This option relates to work still in progress. Experience so far suggests that if $\# > 0.5$ there is usually little advantage in using FP models over conventional polynomial models. We can also use the transformation for strictly positive Xs when we cannot get a satisfactory fit otherwise.

We illustrate the use of this option by considering further the automobile data from Example 3. The scaled origin of `displ` (its minimum divided by its maximum) is 0.186. We generally find that a value between about 0.05 and 0.2, typically 0.1, gives good results in FP modeling. To check the effect on the fit of the $m = 1$ model, we reset the scaled origin to 0.1 using the `origin(0.1)` option, and refit the $m = 1$ model (making use of the `df(2)` option). We then display the corresponding value of the origin (ζ), in the same units as `displ`, that `fp` has calculated and used when transforming `displ`:

```
. fp mpg displ, base(foreign weight) df(2) origin(0.1)
MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) for Y = mpg, X = displ.
(*) Base model  Linear   Quadratic     Cubic     BoxTid      df(2)
-------------------------------------------------------------------
p       --         1        1, 2     1, 2, 3      1, 1        -2
D   388.366    388.327    381.517    377.588    379.278    375.803
G                0.000      6.810     10.739      9.049     12.524
. display $S_23
40.555556
```

The deviance has changed from 376.53 to 375.80, a reduction of 0.73, indicating a slightly (but not statistically significantly) better fit. The value of ζ (which was stored in macro `$S_23`) is 40.55, so `fp` has transformed `displ` to $(xvar - 40.55)/(xmax - 40.55)$ before fitting. (See the help file for a full list of values saved in the `$S_#` macros.)

An alternate way of treating zeroes and negative values of X is provided by the `zero` option, which converts such values to zero in the regression analysis. This option is described and explained in the help file `fp.hlp`

Models with m>2

In principle, we could fit models with $m > 2$ by including one or more powers of X specified using `fixpowers()`. To investigate all models with $m = 3$, for example, we could include each single power in our set $\mathcal{P}$ (from -2 to 3) in turn in `fixpowers()`, and then run `fp`. The best models with "$m = 2$" would then be compared to identify the best $m = 3$ models. In practice, it is more convenient to use the alternative program `fpx`, which is designed to fit all models with degree $\leq m$ for any m (see brief description below). A word of warning: because `fpx` carries out a large number of fits when $m > 2$, it can burn

a lot of CPU time. Restricting the set of powers using the `powers()` option will reduce the time, usually at the cost of only a minor loss of flexibility. For example, trying `powers(-2,-1,0,1,2,3)` or even `powers(-1,0,1,2)` is worthwhile.

We illustrate the fitting of a model with $m = 3$ on the hardwood data already shown in Figure 1. For these data the best model with $m = 3$ is $\tilde{\mathbf{p}} = (-2, 2, 3)$. The gain for this model is 67.86, compared with 55.68 for the best model with $m = 2$, for which $\tilde{\mathbf{p}} = (2, 2)$. The reduction in deviance is 12.18, which is much larger than the critical value of $\chi^2_{2;0.90} = 4.61$. In fact four other models have similar gains: (–1,3,3), ($-\frac{1}{2}$,3,3), (0,3,3), and ($\frac{1}{2}$,3,3). The powers in these five models are very similar, and the curves they represent are similar too. Figure 11 shows the $m = 3$ model (solid line) with the greatest gain. The previous *lowess* curve (dashed line) is also shown.

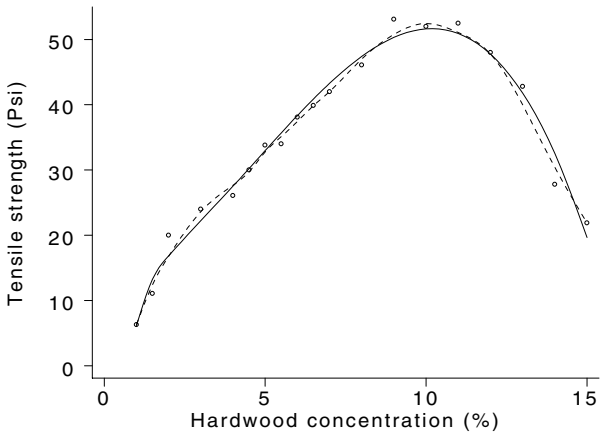

Figure 11: Data from Figure 1 showing *lowess* curve and best FP model with m=3

The best $m = 4$ model has $\tilde{\mathbf{p}} = (\frac{1}{2}, 1, 1, 2)$, and a deviance that is 3.10 lower than the best $m = 3$ model, which is not a statistically significant improvement in fit.

Multiple covariates

Data analysts do not often seem to consider the possibility of curved relationships within multiple regression models, though in reality curvature is probably common. Elsewhere we have suggested an iterative stepwise algorithm to find best-fit FP models when there are several continuous covariates, and we have presented an example using Cox regression analysis (Royston and Altman, 1994). This algorithm is not built into `fp`, but is under development as a separate program (see the entry for `swfp` below).

Related programs: fpx, fpgraph, fpgen and swfp

`fpx` fits FP models of any degree. The syntax has many similarities to that of `fp`; see the help file `fpx.hlp` for further details. `fpx` extends `fp` in two ways:

1. With `fpx`, you can fit an FP model with an arbitrarily large number of degrees of freedom (`df()`), whereas `fp` is limited to `df(4)`. This permits a much wider family of models to be explored.
2. `fpx` outputs the value of the Akaike Information Criterion (AIC) for each model it fits. Some statisticians recommend selecting as "best" the model which has the minimum AIC. The AIC is defined as -2 times the log likelihood plus 2 times the number of parameters estimated; thus, overfitting tends to be penalized. Each additional degree (m) of FP model fitted increases the DF by 2 and therefore adds 4 to the AIC.

The main omissions from `fpx` concern model comparisons: `fpx` does not separately fit and report on quadratic, cubic or Box–Tidwell models, nor does it calculate P-values for comparing selected models.

`fpgraph` (see `help fpgraph`) is available after using `fp` or `fpx` to plot the best-fitting model. The syntax is

`fpgraph` [`,` `dresid(`*dresvar*`)` `eta(`*etavar*`)` `nopts` `nograph` *graph_options*]

`fpgraph` plots the data and fit from the most recently fitted FP model. More precisely, it produces a component-plus-deviance-residual plot. For constant-weight, normal-error models with a single covariate, this amounts to a plot of the observations with the fitted line inscribed. For other normal-error models, (weighted) residuals are calculated in the usual way.

For models with base covariates (specified in `fp` by `base()`), the line is the partial linear predictor, obtained from the FP part of the model only. In order to display sensible values, an adjustment is made so that the mean is the same as the mean of the fitted values from the model which includes all the covariates.

For models with non-normal errors (e.g. logistic and Poisson regression), ordinary residuals have unsatisfactory properties and deviance residuals are calculated instead. These are added to the (partial) linear predictor to give the component-plus-residual values, which are plotted as small circles.

Limitations: `fpgraph` is work in progress and does not handle all the possible models that `fp` itself can. In particular, `mlogit` produces a prediction equation for each level of the outcome variable and is not supported by `fpgraph`. The `offset()` and `exposure()` offset variables, which are options of the `poisson` command, are not currently catered for. For Cox models, only the fitted FP function is plotted.

The options are described in the help file (`fpgraph.hlp`).

`fpgen` generates arbitrary fractional powers of a variable using the formula given under *Formulae*. This can be useful if you want to fit specific FP models directly, without using `fp`. The syntax is

`fpgen` *xvar* [`if` *exp*] [`in` *range*] `,` `powers(`*powlist*`)` [`name(`*pname*`)` `index(#)` `expx(#|sd)` `origin(#)` `replace`]

`fpgen` creates powers of *xvar* according to the values in *powlist*. The new variables are labelled according to their associated powers.

Positive or negative integer powers of *xvar* are defined in the usual way. A power of zero is interpreted as log, so $xvar^0 = \ln(xvar)$. If *xvar* contains any non-positive values, it is transformed as described in the section *Choice of origin for X*.

The options are described in the help file (`fpgen.hlp`).

`swfp` implements a stepwise regression algorithm suggested by Royston and Altman (1994). The program will be released in a future issue of the STB.

Final comments

There is no particular reason other than convention why regression models should include only positive integer powers of covariates. The advantages of the FP approach are that it is a simple extension of existing methods, it is parametric, so we can easily derive predicted values (and standard errors), and we can use standard regression software.

There is no right answer when we fit regression models. What we consider to be the "best" model will depend upon many factors, including the purpose to which the model will be put. The need for a good fit to the data will vary according to circumstances. We do not suggest that the FP approach should replace all others. In particular, nonparametric local smoothing methods have considerable importance and are of much wider applicability. The best approach will vary for different sets of data.

As the examples show, FP models often give a simple model that fits the data well. In fact, the FP approach usually leads to a better fit than conventional polynomials, and usually with fewer terms in model. At least in terms of deviance, you cannot do worse using FPs as standard polynomials are included in the family of models examined.

The IgG, hardwood, mandible and IVF data sets are supplied on the STB-21 diskette in files `igg.dta`, `hardwood.dta`, `mandible.dta` and `ivf.dta` respectively.

Technical note: Deviances

For normal-errors models, suppose we have n observations $Y_1, \ldots, Y_n$, each Y_i being normally distributed with mean μ_i and variance σ^2/w_i. The w_i are weights, assumed to have been normalized in Stata fashion so that $\sum w_i = n$, and σ^2 is known as the scale parameter. Twice the negative log-likelihood may be derived as

$$-2\ln L = n\left[\ln(2\pi\sigma^2) - \bar{w}\right] + (1/\sigma^2)\sum w_i(Y_i - \mu_i)^2,$$

where $\bar{w}$ is the mean of the log of the weights. (If all the weights are equal, $\bar{w}$ is 0.) After obtaining maximum likelihood estimates (MLE) of all the parameters, the expression becomes

$$-2 \ln L = n \left[1 + \ln(2\pi \mathrm{RSS}/n) - \bar{w}\right],$$

the definition of deviance used by `fp` for normal-errors models. Here RSS$= \sum w_i (Y_i - \mu_i)^2$ is the (weighted) residual sum of squares and RSS/n is the MLE of σ^2. The definition differs from that used by McCullagh and Nelder (1989) and by well-known packages such as Glim, Genstat, etc., in that (a) they assume the scale parameter to be known (fixed) in the deviance calculation, (b) they subtract $-2 \ln L$ for the saturated model which has n parameters, one for every observation, and (c) they multiply the resulting expression by the scale parameter, which disappears from the final expression for the deviance. This "unscaled" deviance is simply the RSS as defined above.

For all models (notably logistic and Poisson regression) where the scale parameter is 1, `fp` uses the McCullagh/Nelder definition of deviance. For Cox regression, it uses minus twice the maximized partial log likelihood. For other models where the scale parameter is estimated from the data, it uses the scaled deviance.

The advantage of `fp`'s use of a scaled deviance as a measure of fit is that it makes it easy to interpret changes in deviance, irrespective of the error structure of the observations: asymptotically (for very large n) the difference in scaled deviance between two nested models, where the population values of extra parameters in the "larger" model are zero, has a χ^2 distribution.

Formulae

The full definition of a fractional polynomial model is

$$\phi_m(X; \boldsymbol{\beta}; \mathbf{p}) = \beta_0 H_0(X) + \beta_1 H_1(X) + \cdots + \beta_m H_m(X),$$

where $H_0(X) = 1$, $p_0 = 0$ and for $j = 1, \ldots, m$,

$$H_j(X) = \begin{cases} X^{(p_j)}, & \text{if } p_j \neq p_{j-1}; \\ H_{j-1}(X) \ln X, & \text{if } p_j = p_{j-1}. \end{cases}$$

Round bracket notation signifies the Box–Tidwell transformation, as described above.

Example: $m = 4$, $\mathbf{p} = (-1, 2, 2, 2)$:

$j =$	1	2	3	4
$p_j =$	-1	2	2	2
$H_j =$	X^{-1}	X^2	$X^2 \ln X$	$X^2 (\ln X)^2$

References

Afnan M., M. Mastrominas, E. S. Dimitry, S. Davis, R. M. L. Winston. 1993. The relationship between endometrial thickness and implantation rate following in-vitro fertilization and embryo transfer. unpublished.

Box, G. E. P. and P. W. Tidwell. 1962. Transformation of the independent variables. *Technometrics* 4: 531–550.

Chitty, L. S., S. Campbell and D. G. Altman. 1993. Measurement of the fetal mandible—feasibility and construction of a centile chart. *Prenatal Diagnosis* 13: 749–756.

Isaacs D., D. G. Altman, C. E. Tidmarsh, H. B. Valman and A. D. B. Webster. 1983. Serum immunoglobulin concentrations in preschool children measured by laser nephelometry: reference ranges for IgG, IgA, IgM. *Journal of Clinical Pathology* 36: 1193–1196.

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

McCullagh, P. and J. A. Nelder. 1989. *Generalized Linear Models.* 2d ed. London: Chapman and Hall, 34.

Montgomery, D. C. and E. A. Peck. 1992. *Introduction to Linear Regression Analysis.* 2d ed. New York: John Wiley & Sons, 202–210.

Royston, P. and D. G. Altman. 1994. Regression using fractional polynomials of continuous covariates: parsimonious parametric modeling. *Applied Statistics* 43: 429–467.

ssi6.2	Faster and easier bootstrap estimation

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

The syntax of the `bstrap` and `bs` commands is

```
bstrap progname [, args(...) cluster(varnames) dots reps(#) size(#) ]
bs     "cmd" [exp|macro] exp [, dots eci leave level(#) reps(#) ]
```

`bstrap` and `bs` provide bootstrap resampling for standard errors, confidence intervals, and other measures of statistical accuracy (Efron 1979, Efron and Stein 1981, Efron 1982, Efron and Tibshirani 1986; also see Mooney and Duval 1993 and Stine 1990). `bs` is easy to use and provides an estimate of the standard error and confidence interval for a single statistic. `bstrap` is more difficult and is intended for complex bootstrapping problems.

Beginning with `bstrap`, `bstrap` runs the user-defined program *progname* `reps()` times on bootstrap samples of size `size()`. `bstrap` calls *progname* in two ways. At the outset, `bstrap` issues "*progname* `?`" and expects *progname* to set the global macro `S_1` to contain a list of variable names under which results are to be stored. Thereafter, `bstrap` issues straight "*progname*" calls, having first set memory to contain a bootstrap sample, and expects *progname* to perform the statistical calculation and store the results using `post`. Details of `post` can be found in Gould (1994a), but enough information is provided below to use `post` successfully. `bstrap` is a faster and more convenient variation on `boot`; see [5s] boot. For those wishing to implement their own special-purpose bootstrapping routines, `bstrap` performs its resampling using `bootsamp`—also see [5s] boot—and collects results using `post`.

`bs` provides an even faster and more convenient way to achieve bootstrap standard errors on single statistics. *cmd* may be any Stata command, program, or ado-file—including user-written programs or ado-files—that calculates and saves a statistic of interest. Although `bs` is limited to evaluating single statistics, it is not necessary that *cmd* calculate only one statistic. `bs` allows any statistic calculated by *cmd* to be selected as the single one of interest. For instance, `summarize` calculates the mean, standard deviation, various percentiles, skewness, and kurtosis. `summarize` would be a good candidate for use with `bs`.

Options

`args(...)`, allowed only with `bstrap`, specifies any arguments to be passed to *progname* on invocation. The query call is then of the form "*progname* `?` ..." and subsequent calls of the form "*progname* ...".

`cluster(`*varnames*`)`, allowed only with `bstrap`, specifies the variable(s) identifying resampling clusters. The default is to treat each observation as representing its own cluster. The sample drawn during a replication is actually a bootstrap sample of clusters.

`dots` requests that a dot be placed on the screen at the beginning of each replication, thus providing entertainment if a large number of `reps()` are requested.

`eci`, allowed only with `bs`, requests that, in addition to normal-distribution based confidence intervals, an empirically based confidence interval be presented. `eci` is the default for `reps()` ≥ 200; for fewer replications, `noeci` is the default. Empirical confidence intervals, calculated by the percentile method, are the percentiles corresponding to $\alpha/2$ and $1-\alpha/2$ from the bootstrap distribution of statistics. Adequately estimating the tails requires more replications than adequately estimating the standard error; see `reps()` below. While you can specify `eci` to force presentation of the empirical confidence interval, you should not.

`leave`, allowed only with `bs`, specifies that a data set of the bootstrapped statistic be left behind in place of the data currently in memory. The default is to leave the original data undisturbed and to discard the bootstrapped statistics once the summary table has been reported. If `leave` is specified, the data left behind contain a single variable named `result` with `reps()` observations. (`bstrap` always leaves behind the data of bootstrapped statistics. The names of the variables are specified in *progname*.)

`level(#)`, allowed only with `bs`, specifies the significance level in percent for the confidence interval.

`reps(#)` specifies the number of bootstrap replications to be performed; `reps(50)` is the default. Since bootstrap estimates are a result of randomly resampling the data, performing two bootstraps in a row on the same problem will not result in the same answer. The difference between two runs, however, goes to 0 as the number of replications goes to infinity. How many replications are enough? The conventional wisdom, summarized for instance in Mooney and Duval (1993, 11), is that for estimates of the standard error, 50–200 replications is generally adequate; for estimates of empirical confidence intervals, you should use at least 1,000 replications.

Alternatively, you may wish to follow more the more liberal, specific, and questionable guidelines developed here. How many is enough should be determined by your standards which, in turn, should be determined by the accuracy requirements of the problem at hand. The formulas

$$.67 \times 100 \times \frac{1}{\sqrt{2n}} \approx \frac{47.4}{\sqrt{n}}$$

and

$$1.96 \times 100 \times \frac{1}{\sqrt{2n}} \approx \frac{138.6}{\sqrt{n}}$$

provide a crude measure of the maximum percentage variation in the estimated standard error that will be observed 50 and 95 percent of the time in sequential runs of `bs` or `bstrap`, where n is the number of replications. (In deriving these formulas, the statistic is assumed to be normally distributed and the bootstrap samples are assumed to be drawn from the underlying population rather than the observed sample. Relaxing the second assumption, I believe, would actually reduce the variation and, if so, these formulas are too pessimistic.)

To use the formulas, imagine you used `bstrap` or `bs` with 50 replications and obtained an estimated standard error of 2. If you ran it again, 50 percent of the time the new standard error would lie within $47.4/\sqrt{50} \approx 6.7$ percent of the original estimate and 95 percent of the time, within $138.6/\sqrt{50} \approx 19.6$ percent. Thus, 50 percent of the time, the new estimate will be between 1.9 and 2.1 and 95 percent of the time between 1.6 and 2.4.

If your interest is in the empirically determined 95 percent confidence interval, you should increase the number of replications by 72 percent over the level you find acceptable based on the standard error but you should never use less than 50 replications and probably not less than 200. (This recommendation is based on the same two assumptions—normally distributed statistic and resampling from the population rather than the sample. In that case, the standard error of the 2.5th percentile is approximately 1.93 times the standard error of the standard deviation. Variation still falls by a multiplicative factor of $1/\sqrt{n}$, so increasing n by a factor of $1/\sqrt{1.93} \approx .72$ offsets the increased variation. The second part of the recommendation deals with ensuring that the 2.5th percentile point is in the interior of the sampled and resampled distributions.)

The table below provides evaluations of the percentage variation formulas at popular levels for `rep()`:

replications	50% variation	95% variation
20	10.6%	31.0%
50	6.7	19.6
100	4.7	13.6
200	3.4	9.8
500	2.1	6.2
1000	1.5	4.4
10000	.5	1.4
100000	.1	.4

`size(#)`, allowed only with `bstrap`, specifies the size of the samples to be drawn. The default is `_N`, meaning to draw samples of the same size as the data. If `cluster()` is specified, the default `_N` means to draw samples containing the same number of clusters as the data. Unless all the clusters contain the same number of observations, resulting sample sizes will differ from replication to replication. If `size(#)` is specified, # must be less than or equal to the number of clusters or, if not clustered, the number of observations.

Remarks

With few assumptions, bootstrapping provides a way of estimating standard errors and other measures of statistical accuracy. It provides a way of obtaining such measures when no formula is otherwise available, when available formulas make assumptions that are not tenable, or when one merely wants to verify that the assumptions typically made do not affect results in this case. Mechanically, the procedure is this: One has a data set containing N observations and an estimator which, when applied to the data, produces certain statistics. One draws, with replacement, N observations from the N observation data set. In this random drawing, some of the original observations will appear once, some more than once, and some not at all. Using that data set, one applies the estimator and estimates the statistics. One then does it again, drawing a new random sample and re-estimating, and again, and keeps track of the estimated statistics at each step of the way (called a replication).

Thus, one builds a data set of estimated statistics. From this data, one can calculate, say, the standard deviation using the standard formula ($\sqrt{\sum(z_i - \bar{z})^2/(k-1)}$, where k is the number of replications). That number is your estimate of the standard error of the statistic. Note that, while the average value of the observed statistic $\bar{z}$ is used in the calculation of the standard deviation, it is not used as the estimated value of the statistic itself. The point estimate is obtained in the normal way using the original N observation. (Many researchers new to bootstrapping think that this average is somehow a better estimate of the parameter than the statistic obtained in normal ways. That is not true. What is true is that if the statistic is biased in some way, $\bar{z}$ exaggerates the bias. Denoting z as the statistic calculated over the entire sample, the amount of the bias can be estimated as $\bar{z} - z$ and, in fact, an unbiased statistic would be $z - (\bar{z} - z) = 2z - \bar{z}$ [Efron 1982, 33]. This adjustment, however, should only be applied if there are strong theoretical reasons to believe the statistic is biased.)

The logic behind the bootstrap is this: All measures of accuracy come from a statistic's sampling distribution. The sampling distribution tells you, when the statistic is estimated on a sample of size N from some population, the relative frequencies of the values of the statistic. The sampling distribution, in turn, is determined by the distribution of the population and the formula used to estimate the statistic.

In some cases, the sampling distribution can be derived analytically. For instance, if the underlying population is distributed normally and one calculates means, the sampling distribution for the mean is distributed t with $N - 1$ degrees of freedom. In other cases, deriving the sampling distribution is too hard and we than say it is unknown, as it is in the case of means calculated from non-normal populations. Sometimes, as it in the case of means, it is not too difficult to derive the sampling distribution as $N \to \infty$. The distribution of means converges to a normal. We will then use that "asymptotic" result to calculate some measure of statistical accuracy on a finite sample of size N even though we know it is incorrect.

As a mechanical matter, if we knew the population distribution, we could obtain the sampling distribution by simulation: we would draw samples of size N, calculate the statistic, and make a tally. Bootstrapping does precisely this, using the observed distribution of the finite sample in place of the true population distribution. Thus, the bootstrap procedure hinges on the assumption that the observed distribution is a good estimate of the underlying distribution. In return, the bootstrap produces not only an estimate of the standard error, but any measure of accuracy desired because it produces an estimate of the sampling distribution.

The accuracy with which the sampling distribution is estimated, given the assumed population distribution, is a function of the number of replications. A crudely estimated sampling distribution is quite adequate if one is only going to extract, say, the standard deviation. A better estimate is needed if one is going to extract a 95 percent confidence interval and, if one is going to extract many features simultaneously about the distribution, a quite good estimate is needed. It is generally believed that replications on the order of a 1,000 produce quite good estimates and that, for measurements of standard errors, many fewer replications are needed; see `reps()` under *Options* above.

Let us begin with the easier-to-use, and more convenient, `bs` command.

Example 1

We wish to obtain a bootstrap standard error and confidence interval for the median of miles per gallon (`mpg`) in the auto data set that is supplied with Stata. `summarize` with the `detail` option calculates, among other things, medians and, according to *Saved Results* in [5s] summarize, `summarize` stores the median in `_result(10)`. Thus:

```
. bs "summarize mpg, detail" _result(10)
                           Bootstrap
    Reps      Pt. Est.    Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
      50            20    .8668498        18.30101    21.69899  (normal based)
(average)       19.94
```

The point estimate of the median (which `bs` obtained by performing `summarize mpg, detail` on the entire sample) is 20. `bs` performed 50 replications to obtain a standard error. The standard deviation of the statistic across the resamples is .8668 and this is an estimate of the standard error. The 95 percent confidence interval was obtained as $20 \pm (1.96)(.8668)$. In addition, and merely for your reassurance, the average value of the statistic across the 20 resamples was 19.94.

Example 2

Problem: Obtain a bootstrap standard error for the coefficient on `weight` in a regression of `mpg` on `weight` and `displ` in the auto data. Use 100 replications.

Solution: `regress` estimates regression models. `_b[weight]` is the coefficient on `weight` after a regression:

```
. bs "regress mpg weight displ" _b[weight], reps(100)
                         Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
     100    -.0065671    .0009789       -.0084857   -.0046486  (normal based)
(average)   -.0067433
```

Example 3

Problem: Obtain a bootstrap standard error for the standard error of the mean of `mpg` in the auto data.

Solution 1: The standard error of the mean is defined as $\sqrt{s^2/n}$ where s^2 is the estimated variance of the sample and n is the number of observations. `summarize` saves s^2 in `_result(4)` and n in `_result(1)`.

```
. bs "sum mpg" sqrt(_result(4)/_result(1))
                         Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
      50     .6725511    .0785286        .5186379    .8264643  (normal based)
(average)    .6607228
```

Solution 2: `ci` calculates the standard error of the mean. The standard error is saved in the global macro `S_4`.

```
. bs "ci mpg" macro S_4
                         Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
      50     .6725511    .0610494        .5528964    .7922058  (normal based)
(average)     .667007
```

Note the use of the word `macro` in the `bs` command. Without it, `bs` would assume that the expression following the command is an ordinary expression. When the expression is a macro, you type the word `macro` followed by the macro name, omitting the dollar sign.

More interesting is why we obtained an estimated standard error of .0785 in the first case and .0610 in the second. The difference has nothing to do with having used `summarize` in one and `ci` in another. Bootstrapping involves randomly resampling the data and two runs will not produce exactly the same answer. However,

- Two runs will produce exactly the same answer if you first set the random-number seed; see [5d] generate.
- Two runs will produce more nearly identical answers as the number of replications increases even if you do not set the random-number seed; see `reps()` under *Options* above.

Example 3.1

Continuing with the previous example, let us calculate both a more accurate estimate of the standard error of the standard error of the mean—we will use 1,000 replications—and an empirically based confidence interval:

```
. bs "ci mpg" macro S_4, reps(1000)
                         Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
    1000     .6725511    .0695833        .5361703    .8089319  (normal based)
(average)    .6613738                    .5329553    .8018642  (empirical)
```

`bs` calculated the empirical confidence interval because we specified more than 200 replications. In this case, the empirical confidence interval is virtually identical to the normal one, indicating that the distribution of the standard error of the mean of `mpg` is approximately normally distributed.

We followed the conventional wisdom of using at least 1,000 replications when obtaining an empirical confidence interval, although I would have been satisfied with fewer. Here are three runs in a row of the same command:

```
. bs "ci mpg" macro S_4, reps(250)
                         Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
     250     .6725511    .0694593        .5364134    .8086887  (normal based)
(average)    .6723161                    .5426767     .792872  (empirical)

. bs "ci mpg" macro S_4, reps(250)
```

```
                        Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
     250     .6725511    .0682854        .5387142     .806388  (normal based)
(average)    .6732489                    .5354373    .8017403  (empirical)

. bs "ci mpg" macro S_4, reps(250)
                        Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
     250     .6725511    .0687151        .5378719    .8072303  (normal based)
(average)     .658848                    .5442142    .8149669  (empirical)
```

Example 3.2

In another data set, we obtain a standard error of the 90th percentile (stored in `_result(12)` by `summarize, detail`):

```
. bs "summarize d_time, detail" _result(12), rep(1000)
                        Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
    1000     29.78806    16.98772       -3.507259    63.08338  (normal based)
(average)     30.6736                       6.6529    77.21201  (empirical)
```

`d_time` in this data is the time from the last treatment until death for 100 very sick patients and negative times are not possible; the empirical interval is clearly preferred.

Example 4

Problem: Obtain an estimate of the 95 percent confidence interval of the difference in medians of `mpg` between foreign and domestic cars in the auto data.

Solution: Stata has no command that presents difference in medians but we can write one. It is so short we will enter it interactively and then estimate our result:

```
. program define diffmed
  1.         summarize mpg if foreign, detail
  2.         local mpgfor = _result(10)
  3.         summarize mpg if ~foreign, detail
  4.         global S_1 = _result(10) - `mpgfor´
  5. end
. bs diffmed macro S_1, reps(100)
                        Bootstrap
    Reps      Pt. Est.   Std. Err.       [95% Conf. Interval]
-------------------------------------------------------------
     100         -5.5     1.37447       -8.193911   -2.806089  (normal based)
(average)      -5.335
```

Speeding execution

Consider the command:

```
. bs "sum mpg, det" _result(10)
```

`bs` has no way of knowing that only `mpg` plays a role in the summary and is thus forced to make bootstrap samples that include all the variables in the data set. `bs` will run faster if you keep only the variables relevant to the calculation:

```
. keep mpg
. bs "sum mpg, det" _result(10)
```

In practice, unless you have hundreds of variables in your data, keeping the relevant variables will make little difference.

Missing values

Data sets invariably have missing values for some variables. Since `bs` does not know which variables play a role in the specified command, it has no way of excluding the missing values. That causes no problem in one sense because all Stata commands deal with missing values gracefully.

It does, however, cause a statistical problem. Bootstrap sampling is defined as drawing, with replacement, resamples of size N from a sample of size N. `bs` determines N by counting the number of observations in the data set, not counting the number of nonmissing observations on the relevant variables. The result is that too many observations are resampled and, moreover, the resulting resamples, since drawn from a population with missing values, are of unequal sizes.

If the number of missing values relative to sample size is small, this will little difference. If you have a large number of missing values, however, you should first omit the missing values:

```
. drop if mpg==.
. bs "sum mpg, d" _result(10)
```

Saved Results

`bs` saves in the global `S_#` macros:

`S_1`	point estimate calculated over the entire data set
`S_2`	average estimate calculated over resamples
`S_3`	estimated standard error (standard deviation over resamples)
`S_4`	lower bound of empirical confidence interval
`S_5`	upper bound of empirical confidence interval

The bstrap command

`bstrap` is more complicated to use than `bs`, slightly slower, and more flexible.

To use `bstrap`, you must first write a program, a program that follows the outline of the program required by `simul` (see Gould 1994b):

```
program define progname
    if "`1'"=="?" {
        global S_1 "variable name(s)"
        exit      }
    perform estimation on sample in memory
    post results
end
```

There must be the same number of results following the `post` command as variable names following the `global S_1` command.

Example 5

As in example 1, we will obtain a bootstrap standard error for the median of `mpg` in the auto data. Recall that `summarize` with the `detail` option calculates, among other things, medians, and stores the median in `_result(10)`. First, we must write a program to calculate the statistic given a sample:

```
program define mpgmed
        version 3.1
        if "`1'"=="?" {
                global S_1 "median"
                exit
        }
        summarize mpg, detail
        post _result(10)
end
```

That done, we can

```
. bstrap mpgmed
Bootstrap:
         Program:            mpgmed
         Arguments:
         Replications:       50
         Data set size:      74
         Sample size:        _N
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+--------------------------------------------------------
  median |      50       20.12   .9066647         19         22
```

The reported standard deviation is the bootstrap estimate of the standard error of the median. The best estimate of the median is still, however, the median reported by running `summarize mpg, detail` on the entire sample, which is 20. The estimate of the standard error is .907.

After running `bstrap`, the data in memory contains the bootstrapped results:

```
. describe
Contains data
  Obs:    50 (max=  5040)                  mpgmed bootstrap
 Vars:     1 (max=    99)
Width:     4 (max=   200)
  1. median       float  Sorted by:

. list in 1/4
       median
  1.       20
  2.       21
  3.     20.5
  4.       20
```

Example 6

This example corresponds to example 2; we wish to obtain a bootstrap estimate of the standard error of the standard error of the mean of `mpg` using the auto data. Our solution below corresponds to the previous solution 2:

```
program define sesemean
        version 3.1
        if "`1'"=="?" {
                global S_1 "se"
                exit
        }
        ci mpg
post $S_4
end
```

Then,

```
. bstrap sesemean
  (output omitted)
```

Example 7

In our examples so far we have not exploited the power of `bstrap` and, as a result, by comparison to `bs`, `bstrap` seems merely inconvenient. Let us obtain bootstrap estimates of the standard errors of the coefficients in a regression of `mpg` on `weight` and `displ` using 75 replications:

```
program define myreg
        version 3.1
        if "`1'"=="?" {
                global S_1 "weight displ cons"
                exit
        }
        regress mpg weight displ
        post _b[weight] _b[displ] _b[_cons]
end
```

We now estimate the regression on the entire sample and run the bootstrap to obtain the new estimates of the standard errors:

```
. use auto, clear
. regress mpg weight displ
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  2,    71) =   66.79
   Model |  1595.40969     2  797.704846               Prob > F      =  0.0000
Residual |  848.049768    71  11.9443629               R-square      =  0.6529
---------+------------------------------               Adj R-square  =  0.6432
   Total |  2443.45946    73  33.4720474               Root MSE      =  3.4561
------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  weight |  -.0065671   .0011662     -5.631   0.000      -.0088925   -.0042417
   displ |   .0052808   .0098696      0.535   0.594      -.0143986    .0249602
   _cons |   40.08452    2.02011     19.843   0.000       36.05654    44.11251
------------------------------------------------------------------------------
```

```
. bstrap myreg, reps(75)
Bootstrap:
         Program:            myreg
         Arguments:
         Replications:       75
         Data set size:      74
         Sample size:        _N
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+--------------------------------------------------------
  weight |      75   -.0066732   .0010785  -.0115266  -.0043077
   displ |      75    .0059866   .0086184  -.0154117   .0358489
    cons |      75    40.26126    2.24061   36.15217   50.04395
```

We could have obtained standard errors from `bs`, but that would have required executing the `bs` command three times. `bstrap` is faster in this case.

Regardless of which we use, we obtain our best estimate of the coefficients from the regression on the entire sample. We obtain our standard errors from the standard deviations reported by `bstrap` or `bs`.

In this case, we obtained our bootstrap estimates by resampling the entire data, which is appropriate if we believe both the dependent and independent variables are random. In the classic case, however, the independent variables are considered fixed and one should resample the residuals. That is, in the model,

$$y_j = \mathbf{x_j}\mathcal{B} + \epsilon_j$$

it is the ϵ_j that is the random component. An estimate e_j of ϵ_j can be obtained by estimating the regression on the entire sample to obtain

$$y_j = \mathbf{x_j}\mathbf{b} + e_j$$

It is the e_j that one wants to resample. Letting e_j^* be a set of resampled residuals, one can then calculate

$$y_j^* = \mathbf{x_j}\mathbf{b} + e_j^*$$

One then re-estimates the model on y_j^* and collects the resulting parameter estimates $\mathbf{b}^*$ for the standard-error calculation.

We can do this with `bstrap`:

```
program define myreg2
        version 3.1
        if "`1´"=="?" {
                use auto, clear
                regress mpg weight displ
                predict yhat
                save base, replace
                predict e, resid
                keep e
                global S_1 "weight displ cons"
                exit
        }
        merge using base
        gen ystar = yhat + e
        regress ystar weight displ
        post _b[weight] _b[displ] _b[_cons]
end
```

`bstrap` calls our program once at the outset to obtain the names of the parameters, which we store in the macro `S_1`. We can use this initialization call for our own purposes as well. In `myreg2`, we load the data, estimate the overall regression, calculate the predicted values and save the data, and finally calculate the residual and leave just that in memory. It is the residual that `bstrap` will now resample. When we are subsequently called, we take the resampled residuals in memory and merge them back with our estimation sample, form y_j^*, and re-estimate the regression. The result of doing this is

```
. bstrap myreg2, reps(75)
Bootstrap:
         Program:            myreg2
         Arguments:
         Replications:       75
         Data set size:      74
         Sample size:        _N
```

```
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+--------------------------------------------------------
  weight |      75   -.0065262   .0011292  -.0096954  -.0042488
   displ |      75    .0063092   .0089959  -.0154251   .0325636
    cons |      75    39.69676    2.09894   35.60074   45.19046
```

Speeding execution and missing values

The same comments made above about `bs` concerning speeding execution and missing values apply to `bstrap`.

Performance

`bstrap` runs 36 percent faster and `bs` 45 percent faster than `boot`. `bs` is 15 percent faster than `bstrap`. The timings are

replications	`boot` (secs.)	`bstrap` (secs.)	`bs` (secs.)
20	4.56	4.01	3.41
100	22.30	16.92	14.51
500	128.25	87.78	74.87
1000	278.26	178.07	152.15

Timings were performed on a DOS 25MHz 486 computer running Intercooled Stata. Example 1, the standard error of the median of `mpg` in the auto data, was used. (20 and not 50 replications were included in the timings because, when these timings were performed, 20 was the planned default number of replications.)

There is evidence of nonlinearity in replications for all three commands, but only at small numbers of replications. Between 100 and 1,000 replications, all three functions are linear. To minimize the effects of fixed-cost differences, the percentage changes above were calculated using 1,000-replication timings. For a small number of replications, such as 20, these fixed costs are significant. `bstrap` is only 20 percent and `bs` 25 percent faster than `boot`. All three commands, however, are absolutely fast for small numbers of replications and performance comparisons are only important as the number of replications increases.

The increasing difference with number of replications is not unexpected. Both `bstrap` and `bs` are implemented in terms of `post` and `post` employs a more complicated setting up process to speed subsequent replications.

The nonlinearity (which is small) was unexpected. Experimentation revealed it had to do with the computer's buffering of I/O. With small numbers of replications, many I/Os could be avoided as requests were satisfied out of the computer's buffers.

In terms of further decreasing execution times, `simul` (see Gould 1994b) provides a theoretical lower bound on what is achievable with current Stata technology. `bstrap` is basically identical to `simul` except that it draws bootstrap samples before calling the user-supplied program. This includes saving at the outset the data in memory and then, after each replication, reloading the data and drawing a resample. The resampling is performed by `bootsamp` (see [5s] boot).

Thus, in order to get a timing of how long `bstrap` would take if drawing the resample took no time, I made timings substituting a void `bootsamp` routine. To obtain a timing exclusive of bootstrap sampling and reloading the data, I made timings with `simul` having it, over and over, recalculate the median of `mpg`. In the table below, columns (1) through (3) present the timings; column (1) is just a repeat of `bstrap`'s timings from the table above. These timings allow estimating the execution time of `bootsamp`, the time spent in reloading data, and the time spent doing everything else required to perform the bootstrap estimation. All timings are in seconds and were performed on the same DOS 25MHz computer running Intercooled Stata:

replications	(1) `bstrap` total	(2) `bstrap with` void `bootsamp`	(3) `simul`	(4) (1)-(2) `bootsamp`	(5) (2)-(3) I/O	(6) (1)-(4)-(5) residual
20	4.01	2.03	1.15	1.98	.88	1.15
100	16.92	8.29	4.56	8.63	3.73	4.56
500	86.78	38.88	21.36	47.90	17.52	21.36
1000	178.07	78.11	43.12	99.96	34.99	43.12
percentages of (1)						
20				49	22	29
100				51	22	27
500				55	20	25
1000				56	20	24

Columns (4), (5), and (6) present the decomposition of the total execution time (1). To wit: `bstrap` is spending over 50 percent of its time in `bootsamp` drawing the resample, spending roughly 20 percent of its time thrashing the disk with files, and 25 percent of its time in the machinations of `post` (and calculating the median of `mpg`, of course, but this takes virtually no time).

There is little that can be done as a Stata programmer about column (6); I believe `post` is about as efficient as possible given current Stata technology. This does suggest, however, that a new internal facility for posting results could speed `bstrap` by 25 percent. `post`'s syntax was designed with an eye toward internalizing it but, as of now, we have no plans to do so.

The 20 percent spent in reloading the underlying data is probably not avoidable. Many Stata commands do allow frequency weights and a `bootsamp` that left the data alone and merely set the weights would allow avoiding this cost, but this would make the user program more complicated and, anyway, writing a program to create weights (as opposed to rearranging the data) is difficult.

Attacking the 50-plus percent of time spent in `bootsamp` is likely to be the most profitable. `bootsamp` is an ado-file and there is more than one way it could be coded.

References

Efron, B. 1979. Bootstrap methods: Another look at the jackknife. *Annals of Statistics* 7: 1–26.

——. 1982. *The Jackknife, the Bootstrap and Other Resampling Plans.* Philadelphia: Society for Industrial and Applied Mathematics.

Efron, B. and C. Stein. 1981. The jackknife estimate of variance. *Annals of Statistics* 9: 586–596.

Efron, B. and R. Tibshirani. 1986. Bootstrap methods of standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science* 1: 54–77.

Gould, W. 1994a. ssi6: Routines to speed Monte-Carlo experiments. *Stata Technical Bulletin* 20: 18–22.

——. 1994b. ssi6.1: Simplified Monte-Carlo simulations. *Stata Technical Bulletin* 20: 22-24.

Mooney, C. Z. and R. D. Duval. 1993. *Bootstrapping: A Nonparametric Approach to Statistical Inference.* Newbury Park, CA: Sage publications, Sage university paper series, quantitative applications in the social sciences, no. 95.

Stine, R. 1990. An introduction to bootstrap methods: examples and ideas. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long, 353–373. Newbury Park, CA: Sage Publications.

sts9	Johansen's test for cointegration

Ken Heinecke and Charles Morris, Federal Reserve Bank of Kansas City, FAX 816-881-2199

Most time series techniques can be applied only to covariance stationary variables, that is, variables whose unconditional means and variances are constant over time. Most variables encountered in empirical research, though, are nonstationary. For example, both the mean and variance of gross domestic product (GDP) in the United States grow over time, driven by growth in population, capital stock, and productivity. Techniques have been developed for transforming variables such as GDP to stationarity for analysis, and for converting predictions of these stationary transformations back to the nonstationary form.

Cointegration is a relatively new and powerful approach to modeling nonstationary variables. Nonstationary variables like GDP may grow without bound, but, they may also maintain a relatively constant relationship to other nonstationary variables. For example, the relationship between GDP and its determinants, (population, capital, etc.) is fairly stable. If a linear combination of two or more nonstationary variables is stationary, the variables in the linear combination are said to be cointegrated.

Stata has for some time offered commands for analyzing nonstationary and cointegrated variables. The Stata time series library (sts7.4) offers graphical tests for nonstationarity (`ac` and `pac`), the Dickey–Fuller (`dickey`) and Phillips–Perron (`ppunit`) tests for nonstationarity, the Engle–Granger limited-information test for cointegration (`coint`), and Hakkio's implementation of MacKinnon's approximate asymptotic p-values for these tests (Hakkio 1994).

This insert describes `mlcoint`, an implementation of Johansen's method for determining the number of cointegrating relationships in a set of nonstationary variables. `mlcoint` also calculates Johansen's maximum-likelihood estimates of the cointegrating relationships (Johansen 1988; Johansen and Juselius 1990). Johansen's method is a full-information technique and is preferred to the Engle–Granger limited-information approach. This insert also describes `lrcotest` and `wcotest`, two commands for testing restrictions on the estimated cointegrating relationships. The Johansen procedure requires the estimation of vector autoregressions (VARs) and the calculation of eigenvalues of functions of residual matrices. As a consequence, our implementation draws heavily both on the Stata time series library and on Stata's matrix programming facility.

The next section briefly explains cointegration and the Johansen procedure. The following section describes `mlcoint`—our implementation of Johansen's maximum-likelihood test. The next section is an application of `mlcoint` in a recent empirical study. The final section illustrates `lrcotest` and `wcotest`, the likelihood ratio and Wald tests, respectively, of restrictions on the cointegrating relationships.

Cointegration and Johansen's procedure

Nonstationary variables can often be transformed to stationary variables by differencing them. Thus, for example, the log of GDP is nonstationary but the first difference of the log of GDP—that is, the growth rate of GDP—is stationary. In theory, it may require several differences to induce stationarity, and the number of differences required is called the order of integration of the original nonstationary variable, denoted by $I(k)$ where k is the order of integration. (Integration is the inverse of differencing.) In practice, most variables of interest are integrated of order 1.

When a linear combination of $I(1)$ variables is stationary, the variables are said to be *cointegrated*. Cointegrated variables are bound together in the long run by one or more equilibrium relationships. Each of the variables is unbounded—is $I(1)$—but the variables cannot drift indefinitely far apart.

It turns out (Engle and Granger, 1987) that variables are cointegrated if and only if they can be written in error correction form, a particularly tractable and interpretable form. As an example, suppose that two $I(1)$ variables, y_t and x_t, follow the autoregressive distributed lag process

$$y_t = \mu + \alpha(L)y_{t-1} + \beta(L)x_{t-1} + \epsilon_t,$$

where ϵ_t is an i.i.d. disturbance and $\alpha(L)$ and $\beta(L)$ are pth order polynomials in the lag operator, L, that is,

$$\begin{aligned} Ly_t &\equiv y_{t-1}, \\ L^k y_t &\equiv L(L^{k-1}y_t) \\ &\equiv y_{t-k} \end{aligned}$$

and

$$\begin{aligned} \alpha(L) &= 1 - \alpha_1 L - \alpha_2 L^2 - \cdots - \alpha_p L^p, \\ \beta(L) &= 1 - \beta_1 L - \beta_2 L^2 - \cdots - \beta_p L^p. \end{aligned}$$

This equation is a very general specification; for instance, it could arise as one equation in a bivariate vector autoregression. The terms in this equation can be rearranged to yield the error correction form

$$\Delta y_t = \mu + \tilde{\alpha}(L)\Delta y_{t-1} + \tilde{\beta}(L)\Delta x_{t-1} - \lambda(y_{t-1} - \gamma x_{t-1}) + \epsilon_t,$$

where $\tilde{\alpha}(L)$ and $\tilde{\beta}(L)$ are $(p-1)$-order polynomials in L. Δy_t, Δx_t, and ϵ_t are stationary by assumption; thus the term $(y_{t-1} - \gamma x_{t-1})$ must also be stationary or λ must be zero. If y_t and x_t are not cointegrated, then there is no stationary linear combination of the two variables, in which case $\lambda \equiv 0$. If, on the other hand, y_t and x_t are cointegrated, then $(y_{t-1} - \gamma x_{t-1})$ must be the unique stationary linear combination of these variables. $\lambda(y_{t-1} - \gamma x_{t-1})$ is called an *error correction mechanism* (ECM) and the vector $(1, -y)$ is called the cointegrating vector for y_t and x_t.

The ECM represents an equilibrium relationship between y_t and x_t. Nonzero values of $(y_{t-1} - \gamma x_{t-1})$ are errors—deviations from the equilibrium relationship—and λ is a speed-of-adjustment parameter that measures how rapidly these errors are corrected.

The notion of cointegration generalizes easily to a set of more than two $I(1)$ variables. Instead of a unique cointegrating relationship, a set of n variables can have as many as $n-1$ cointegrating vectors. Different cointegrating vectors may include different subsets of the n variables.

The Johansen procedure is designed to test for the number of cointegrating relationships among a set of variables and to estimate the error correction mechanisms. To explain the procedure, let y_t be an n-vector of $I(1)$ variables generated by the process

$$y_t = A(L)y_{t-1} + \epsilon_t,$$

where $A(L)$ is a pth-order matrix polynomial in L and ϵ_t is a vector of i.i.d. normal disturbances. As in our earlier example, this VAR can be rewritten as the vector error correction model

$$\Delta y_t = \tilde{A}(L)\Delta y_{t-1} + \Pi y_{t-1} + \epsilon_t,$$

where $\tilde{A}$ is a $(p-1)$-order polynomial in the lag operator. Johansen (1988) shows that the number of cointegrating vectors for the elements of y_t depends on the rank of Π. There are three possibilities:

1. If Π has full rank ($\mathrm{rank}(\Pi) = n$), there are no cointegrating vectors and the variables in y are stationary. To see this, rewrite the error correction model as

$$y_{t-1} = \Pi^{-1}(\Delta y_t - \tilde{A}(L)\Delta y_{t-1} - \epsilon_t).$$

   All the terms on the right-hand-side of this equation are stationary, thus y_{t-1} is stationary. Thus $\mathrm{rank}(\Pi) = n$ contradicts the assumption that the elements of y_t are $I(1)$, and we ignore this possibility in what follows.

2. If $\mathrm{rank}(\Pi) = 0$, then $\Pi = 0$ and, again, there are no cointegrating vectors. In this case, each element of y has an independent unit root, and the VAR can be consistently estimated in first differences.

3. If $\mathrm{rank}(\Pi) = k < n$, the number of cointegrating vectors equals k, the rank of Π. In this case, Π can be decomposed as

$$\alpha\beta' = \Pi,$$

   where β' is a $k \times n$ matrix of cointegrating vectors and α is an $n \times k$ matrix of weights (speed-of-adjustment parameters) on the cointegrating vectors (analogous to λ in the two-variable model). Specifically, the rows of β' are the k cointegrating vectors and the (i,j)-th element of α is the weight (speed-of-adjustment) of the jth cointegrating vector in the ith equation.

Johansen proposes two likelihood ratio tests for determining the rank of Π. The rank of Π is equal to the number of nonzero eigenvalues of Π. Thus, the Johansen tests amount to tests for the number of nonzero eigenvalues. Johansen's first test, the *maximal eigenvalue test*, is really a sequence of tests. The estimated eigenvalues of Π are sorted in descending order. (The eigenvalues are guaranteed to be nonnegative real numbers.) The maximal eigenvalue statistics are simple functions of the eigenvalues (see Johansen (1988) for details). The kth statistic provides a test of the null hypothesis that $\mathrm{rank}(\Pi) = k$ against the alternative that $\mathrm{rank}(\Pi) = k+1$. Johansen's second test, the *eigenvalue trace test*, is related to the first. The eigenvalue trace statistics are the running sums of the maximal eigenvalue statistics. The kth eigenvalue trace statistic provides a test of the null hypothesis that $\mathrm{rank}(\Pi) \leq k$ against the alternative that $\mathrm{rank}(\Pi) > k$. Critical values for these tests for systems of up to 11 variables can be found in Osterwald-Lenum (1992).

In order to conduct the Johansen tests, the number of lags in $\tilde{A}(L)$ must be known. The test is derived under the assumption that ϵ_t is i.i.d. normal. One practical strategy is to estimate the vector error correction model for a variety of lag lengths, then to select the smallest lag length that yields i.i.d normal residuals. In practice, it is simpler to estimate the original VAR of the nonstationary variables than the error correction model. This procedure works because the estimated residuals from the two forms are identical. Because the model is a VAR, the estimates and tests of the residuals can be conducted one equation at a time.

mlcoint: an implementation of the Johansen procedure

`mlcoint` uses Johansen's (1988) method to calculate maximum-likelihood estimates of the matrix of cointegrating vectors, β', and the matrix of weights, α, for a system of variables. `mlcoint` also reports the eigenvalues of Π and the two sequences of test statistics for the number of cointegrating vectors: the maximal eigenvalue statistics and the trace statistics. The syntax of `mlcoint` is

`mlcoint` *varlist* [`if` *exp*] [`in` *range*] [`,` `noconstant` `lags(#)` `nonormal` `regress` `standard` `static(`*varlist*`)`]

`noconstant` suppresses the constant.

`lags(#)` specifies the number of lags in the VAR relating the levels of the variables in the *varlist*. The default is one lag.

`normal` suppresses the display of α and β'.

`regress` displays the regressions in the VAR, which are normally suppressed.

`standard` displays the standardized versions of α and β'. By default, the normalized versions are displayed.

`static(`*varlist*`)` specifies variables that enter the VAR but that are not part of the system of cointegrated variables. These are typically static variables, such as dummy variables for particular time periods.

Example: The demand for and supply of bank loans

Becketti and Morris (1993) consider whether nonbank sources of credit have become better substitutes for bank loans in the United States. Because monetary policy operates, in part, by influencing the availability of bank credit, the answer to this question has important implications for the conduct of monetary policy. Becketti and Morris estimate a reduced-form model for bank loans and look for the shifts in the reduced-form parameters implied by an increase in the substitutability between bank and nonbank loans.

As is frequently the case, theory provides no guidance on the dynamic specification of the demand for and supply of bank loans. Becketti and Morris test for the cointegration of bank loans with related variables as part of their specification search. The variables in the model are the log of commercial and industrial (C&I) loans made by banks, the log of business fixed investment, the log of business inventories, the log of corporate net cash flow, the federal funds rate, a measure of the mortgage interest rate, and the yield on 3-month Treasury bills. The model includes a dummy variable that is equal to one in the first quarter of 1973, when credit controls temporarily inverted the normal relationship between the rate on bank loans and the commercial paper rate. The data in this example cover the period from the fourth quarter of 1954 through the fourth quarter of 1974. (The data used in the paper cover a longer period. See the paper for more details.)

```
. use test
(1954:Q4 to 1974:Q4)
. describe
Contains data from test.dta
  Obs:    81 (max= 13576)                  1954:Q4 to 1974:Q4
 Vars:    11 (max=   250)
Width:    37 (max=   502)
  1. year         int    %8.0g             Year
  2. quarter      int    %8.0g    quarter  Quarter
  3. date         float  %9.0g             Date
  4. lci          float  %9.0g             log of comm & industrial loans
  5. lfinr        float  %9.0g             log of bus. fixed investment
  6. linvb        float  %9.0g             log of business inventories
  7. lcash        float  %9.0g             log of corporate net cash flow
  8. rff          float  %9.0g             federal funds rate
  9. rmort        float  %9.0g             2ndary mkt yld FHA mortgages
 10. rtb3         float  %9.0g             3-month T-bill rate
 11. D731         byte   %8.0g             1 if 1973:Q1, else 0
Sorted by:  year  quarter
```

We use `mlcoint` to test for cointegrating vectors and to estimate the standardized values of α and β'. The decomposition $\alpha\beta' = \Pi$ does not produce unique estimates of α and β'; they are subject to normalization. By default, `mlcoint` displays the normalization used by Johansen and Juselius. The `standard` option renormalizes β' so that the elements along the main diagonal are set to 1. This standardization frequently makes the cointegrating vectors easier to interpret. (α and β' can be suppressed entirely by specifying the `nonormal` option.)

```
. mlcoint lci lfinr linvb lcash rff rmort rtb3, lags(3) static(D731)
              Maximal
             eigenvalue    Trace
Eigenvalues  statistics  statistics
-----------------------------------
 .49810389    53.770247   143.20193
 .35592582    34.315427   89.431685
 .22690901    20.073965   55.116258
 .20991048    18.377504   35.042293
  .1361805    11.418533   16.664789
 .05269449    4.2224233   5.2462557
 .01304029    1.0238324   1.0238324
Normalized Beta´
             lci       lfinr       linvb       lcash         rff       rmort
vec1   8.2111169   2.3373851   9.9509562  -11.020102    141.8105   -131.8916
vec2  -35.152846   19.561222   2.5833305   22.152547   69.711541  -5.5859522
vec3  -18.066601   4.1278469   13.575604   7.8640453  -387.84061   135.15789
vec4  -8.0394895   1.8778115  -32.428552    25.31537   84.649501   464.31987
vec5  -4.7748735   14.256283  -17.767234  -1.6379929   138.97698   199.25813
vec6  -21.506409   31.853869   8.7072734  -9.2866495  -47.326449  -63.608341
vec7   6.5176516  -22.299819   22.304887  -2.4118364  -36.796837   6.7258063
```

```
           rtb3
vec1  -246.38517
vec2   54.039207
vec3   479.71214
vec4  -342.99164
vec5  -157.53091
vec6   35.270378
vec7    28.17488

Normalized Alpha
             vec1        vec2        vec3        vec4        vec5        vec6
  lci  -.00103048   .00346921  -.00096933   .00109281  -.00289944   -.0001909
lfinr   .00522932   .00174119  -.00153536   .00072536  -.00378395  -.00274383
linvb   .00300533   .00176897    .0012646  -.00075802  -.00122996   .00022439
lcash   .01314884    .0005851  -.00232835  -.00249405     -.00692  -.00180451
  rff  -.00131544  -.00141552  -.00031901  -.00031189  -.00149853    .0001836
rmort  -.00050176   -.0001966   -.0004696  -.00053419  -.00022372   .00005604
 rtb3   .00027168  -.00090946  -.00103893   .00011416  -.00100533   .00026513

             vec7
  lci   .00016255
lfinr  -.00018434
linvb   .00011762
lcash   .00224957
  rff  -.00015801
rmort  -.00008605
 rtb3  -.00014028
```

The 5 percent critical values for the maximal eigenvalue and trace statistics (as reported by Osterwald-Lenum's Table 1) are

$n-rank(\Pi)$	Maximal eigenvalue statistic	Trace statistic
1	3.76	3.76
2	14.07	15.41
3	20.97	29.68
4	27.07	47.21
5	33.46	68.52
6	39.37	94.15
7	45.28	124.24
8	51.42	156.00
9	57.12	192.89
10	62.81	233.13
11	68.83	277.71

Comparing the maximal eigenvalue statistics and the trace statistics to these critical values suggests that this system has one cointegrating vector. This vector is the first row of β', that is

$$\beta'[1,.] = (8.21, 2.34, 9.95, -11.02, 141.81, -131.89, -246.39).$$

The corresponding weights are found in the first column of α. For example, the weight of the cointegrating vector (the first row of β') in the fourth equation (the cash flow equation) is $\alpha[4,1] = .01$.

Testing restrictions on elements of the cointegrating vectors

By assumption, Π, α, and β' have rank less than n. But the maximum-likelihood estimates reported by `mlcoint` are full-rank. (Note that all of the estimated eigenvalues are greater than zero.) The maximal eigenvalue and trace statistics indicate the true rank of Π and, hence, the number of cointegrating vectors in β'.

It is also useful to know which elements in a cointegrating vector are truly nonzero. Since each cointegrating vector represents a different equilibrium relationship among the variables in the system, theory sometimes suggests credible zero restrictions. For instance, one cointegrating vector may represent an equilibrium relationship among the quantity variables in the system. Another cointegrating vector may represent an equilibrium constraint across the prices in the system. Imposing credible zero restrictions can improve the estimates of the nonzero parameters.

Johansen and Juselius (1990) propose likelihood ratio and Wald tests for restrictions on the cointegration vectors. `lrcotest` and `wcotest` implement the likelihood ratio and Wald tests, respectively. These programs can be used to test the null hypothesis that one variable or several variables jointly do not enter the cointegrating vectors. `lrcotest` and `wcotest` are less general than the tests proposed by Johansen and Juselius because the programs cannot test restrictions on linear combinations of the variables. `lrcotest` and `wcotest` can only be run after `mlcoint`.

The syntax of these two programs is

```
lrcotest varlist [ , cirel(#) restrict ]
wcotest varlist [, cirel(#) ]
```

`cirel(#)` specifies the number of cointegrating relationships in the system. The default is one.

`restrict` requests the display of the restricted estimates. These include the restricted eigenvalues, normalized eigenvectors, and normalized weights.

The likelihood ratio test estimates the restricted model and compares the likelihood of the restricted to the unrestricted model. The Wald test does not require reestimation, hence `wcotest` cannot display the restricted estimates.

We illustrate `lrcotest` and `wcotest` by applying them to the data from the previous example. Looking back at the estimated β', note that the coefficients on the interest rate variables in the cointegrating vector are an order of magnitude larger than the other coefficients, which multiplied quantity variables. This difference in coefficient size could reflect a difference in the scaling or variability of the interest rate and quantity variables or it may indicate that this cointegrating vector captures an equilibrium relationship that involves only the interest rates.

The likelihood ratio and Wald tests both indicate that the interest rate variables are included in the cointegrating vector. The null hypothesis that the interest rate variables do not enter is rejected at the 1 percent level.

```
. wcotest rff rmort rtb3
Cointegration: Wald test              chi2(3) = 46.08
                                      Prob > chi2 = 0
. lrcotest rff rmort rtb3, restrict
Eigenvalues from restricted model
  .41480211
  .22592182
  .06388363
  .01543231
Normalized restricted Beta´
             lci       lfinr       linvb       lcash         rff       rmort
vec1  -18.156396   13.028547   10.753804   7.1382265           0           0

           rtb3
vec1          0
Normalized restricted Alpha
            vec1
  lci   .00029605
lfinr   .00401968
linvb   .00351447
lcash   .00871831
  rff  -.00197144
rmort  -.00072726
 rtb3  -.00074543
Cointegration: likelihood ratio test  chi2(3) = 11.98
                                      Prob > chi2 = .01
```

Each of the interest rate variables is individually significant as well. For example, here are the tests of the 3-month Treasury bill yield.

```
. lrcotest rtb3
Cointegration: likelihood ratio test  chi2(1) = 5.79
                                      Prob > chi2 = .02
. wcotest rtb3
Cointegration: Wald test              chi2(1) = 12.45
                                      Prob > chi2 = 0
```

The situation is less clear-cut with regard to the quantity variables. The four quantity variables—C&I loans, business fixed investment, business inventories, and corporate cash flow—are jointly significant.

```
. lrcotest lci lfinr linvb lcash, restrict
Eigenvalues from restricted model
   .2948431
  .21988827
  .11002065
Normalized restricted Beta´
             lci       lfinr       linvb       lcash         rff       rmort
vec1           0           0           0           0   -108.2608   234.01531
            rtb3
vec1  -14.264512
Normalized restricted Alpha
              vec1
  lci  -.00276866
lfinr   .00026379
linvb   .00124703
lcash    .0026014
  rff  -.00119857
rmort  -.00077027
 rtb3  -.00060523
Cointegration: likelihood ratio test   chi2(4) = 26.52
```

Individually, however, all the variables except cash flow and possibly inventories are insignificant.

```
. wcotest lci
Cointegration: Wald test              chi2(1) = 2.42
                                      Prob > chi2 = .12
. wcotest lfinr
Cointegration: Wald test              chi2(1) = .2
                                      Prob > chi2 = .66
. wcotest linvb
Cointegration: Wald test              chi2(1) = 3.6
                                      Prob > chi2 = .06
. wcotest lcash
Cointegration: Wald test              chi2(1) = 7.3
```

These results are analyzed in detail in Becketti and Morris.

Notes

The matrix formulas for $\widehat{\Pi}$, the maximum-likelihood estimate of Π, can be found in Johansen and Juselius. You must have the Stata time series library installed in order to use `mlcoint`, `lrcotest`, and `wcotest`. You can install this library either by copying it to your personal `ado` directory or by storing it in a separate directory that you add to your `adopath`. `mlcoint` calls `tsfit` to estimate the VARs used to calculate $\widehat{\Pi}$. The lagged variables created in this process are left in your data set. Finally, as noted above, `mlcoint` must be run before `lrcotest` or `wcotest` can be used.

References

Becketti, S. 1994. sts7.4: A library of time series programs for Stata. *Stata Technical Bulletin*, this issue.

Becketti, S. and C. Morris. 1993. Reduced Form Evidence on the Substitutability between Bank and Nonbank Loans. Research Working Paper RWP 93–18. Federal Reserve Bank of Kansas City.

Engle, R. F. and C. W. J. Granger. 1987. Co-Integration and error correction: representation, estimation, and testing. *Econometrica* 55: 251–276.

Hakkio, C. 1994. sts6: Approximate p-values for unit root and cointegration tests. *Stata Technical Bulletin* 17: 25–28.

Johansen, S. 1988. Statistical analysis of cointegration vectors. *Journal of Economic Dynamics and Control* 12: 231–254.

Johansen, S. and K. Juselius. 1990. Maximum likelihood estimation and inference on cointegration—with applications to the demand for money. *Oxford Bulletin of Economics and Statistics* 52: 169–210.

Osterwald-Lenum, M. 1992. A note with quantiles of the asymptotic distribution of the maximum likelihood cointegration rank test statistics. *Oxford Bulletin of Economics and Statistics* 54: 461–472.

zz3.5	Computerized index for the STB	(Update)

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

The STBinformer is a computerized index to every article and program published in the STB. The command (and entire syntax) to run the STBinformer is `stb`. Once the program is running, you can get complete instructions for searching the index by typing `?` for help or `??` for more detailed help.

The STBinformer appeared for the first time on the STB-16 distribution diskette and included indices for the first fifteen issues of the STB. The STB-21 distribution diskette contains an updated version of the STBinformer that includes indices for the first *twenty* issues of the STB. As the original insert stated, I intend to include an updated copy of this computerized index on every STB diskette. I encourage you to contact me with suggestions for changes and improvements in the program.

Reference

Gould W. 1993. Computerized index for the STB. *Stata Technical Bulletin* 16: 27–32.