Economic analysis with Stata 6.0

Christopher F Baum
Department of Economics and
Faculty Microcomputer Resource Center
Boston College

Faculty Microcomputer Resource Center

September 4, 1999

Introduction

- **Stata**: a statistical programming language with particular strengths in data manipulation and the analysis of cross-sectional and panel data
- Most similar in capabilities to SAS, but much simpler, more concisely documented, and far less expensive
- More programmable and cross-platform than SPSS
- Narrower features for time series analysis than RATS, but rapidly adding these capabilities

Portability

- One of the most portable, cross-platform-capable languages available for econometric analysis
- Versions available for Mac OS, Windows 3.1/95/98/NT, flavors of UNIX and Linux
- Binary datafiles transportable without translation across all platforms
- Stata programs run without modification across all platforms
- Full versions available at low cost to academics

Extensibility

- Much of Stata is written in its own language, and may be studied and extended
- Stata procedures (.ado files), when placed on the ADOPATH, are automatically available to your copy of Stata as a new command
- Bimonthly issues of the Stata Technical Bulletin (STB) contain new procedures in a peer-reviewed, tested context; these are linked to online help and freely available for download over the Web

User Support

- Stata itself is Internet-upgradeable, as are user-contributed components
- Users and StataCorp participate vigorously in StataList, a moderated LISTSERV mailing list, responding rapidly to users' enquiries
- All StataList ado-files are posted to the Boston College Statistical Software Components Archive on IDEAS, from which they may be freely downloaded. The archive is searchable, and items are Internet-installable.

On-line help

- Full help for all Stata commands is available from the ‘help’ command in a ‘man page’ format
- ‘lookup topic name’ will locate instances of that string anywhere within the help system
- A series of on-line tutorials exhibits the major features of the program
- The four-volume hardcopy *Reference Manual* and one-volume *User’s Guide* provide full documentation and present the underlying formulas and references

Dataset concepts

- Stata's speed results from its holding the entire dataset in memory
- Only one dataset may be used at a time; sophisticated techniques for merging allow the combining of several files into one
- Binary datasets created by Stata usually are given the filetype `.dta` on all platforms
- Access to binary datasets is much faster than access to text files

Memory allocation

- Stata starts with a default memory allocation which may not be sufficient for working with a large dataset
- UNIX Stata may be invoked with the `-kNNNN` switch (`stata -k50000` would allocate 50 Mb of memory) or the command `'set mem 50m'` may be given within the program. No more than 130 Mb is available under AIX.
- Macintosh Stata memory allocation may be adjusted via the Get Info box on the File menu.

Command-line mode

- Stata in its standard form is a command-line program on all platforms, with a ‘dot prompt’
- The StataQuest additions (freely downloadable from www.stata.com) turn any desktop copy of Stata into a limited-feature, menu-driven program suitable for a beginning statistics course, in which the student may point and click to select all available features of the program
- A ‘shell escape’ (!) may be used to access UNIX during a Stata session

‘Batch’ mode

- Any version of Stata may run a program without user intervention.
- In UNIX Stata, run the program ‘myjob.do’ with
  `stata < myjob.do > myjob.out`
  which will place the output in myjob.out.
- To run this as a true batch job, give the UNIX command ‘batch’ first, type the Stata command line, and use CTRL-D to submit the batch job.
- In Mac Stata, launch the do-file to execute the program within.

Desktop Stata

- In a desktop version of Stata, an additional window contains a log of all commands given. You may select any command to bring it into the command window, edit it, and execute it without retyping. A second window lists all the variables in the current dataset with their variable labels.
- Logging of commands and results to a text file may be started and stopped during the session
- Graphs may be generated and viewed; in the UNIX command-line version, they may not be viewed

Case sensitivity

- Like UNIX, Stata is case-sensitive. It expects that commands will be given in lower case. The variables price, PRICE, and Price are different variables. To avoid problems, stick with lower case throughout.
- UNIX file names and directory names must be given in the same case in which they appear in the operating system.

Getting your data in

- Comma-delimited (CSV) or tab-delimited data may usually be read very easily with the insheet command (which does not read spreadsheets!)
- ‘`insheet using filename`’ will expect that your data are in the ASCII text file ‘`filename.raw`’. If the filetype is not `.raw`, it must be specified. You need not specify comma- or tab-delimiting.
- If the first line of the file contains variable names, they will be automatically used within the program.

Getting your data in

- Variables may contain either numeric or string data. Functions exist to create numeric codes from a set of string values, or to convert string values with purely numeric contents to their numeric equivalents.
- The `insheet` command cannot read space-delimited data (even if it is purely numeric). Space-delimited data may be read with the infile or infix commands.

Getting your data in

- A free-format text file with space- (or tab- or comma-) delimited numeric data may be read with infile; i.e.
  `'infile price mpg displ using auto'`
  will read those three variables from the ASCII file `auto.raw`. The number of observations will be determined from the available data.
- The common missing-data indicator '.' (period) may be used to flag values as missing. This eases importation of text files written by SAS.

Getting your data in

- Infile may also be used with fixed-format data, including data containing undelimited string variables, by creating a 'dictionary' file (`.dct`) which describes the format of each variable and specifies where the data are to be found.
- If data are packed tightly, with no delimiters, a dictionary must be used to define the variables' locations.
- The `_column()` directive allows contents of a data record to be retrieved selectively.

Getting your data in

- The `byvariable()` option to infile allows a 'variable-wise' data set to be read; the number of observations must be specified as the value of the option. This is often useful when working with time-series data which may have been retrieved variable by variable, rather than in a columnar format.
- The '`infix`' command presents a syntax similar to that used by SAS for the definition of variables' types and locations in a fixed-format ASCII data set.

Getting your data in

- A logical condition may be applied on the infile or infix commands to read only those records for which certain conditions are satisfied; i.e.

```
infix using employee if sex==‘M’
```

will read only male employees records from the external file, while

```
infile price mpg using auto in 1/20
```

would read only the first 20 observations of the external file.

Getting your data in

- If your data are already in the internal format of SAS, SPSS, Excel, GAUSS, Lotus, or several other programs, you should use Stat/Transfer: the Swiss Army knife of data converters. It is available for Windows and on fmrisc.bc.edu (`stattransfer`).
- Use of Stat/Transfer will preserve variable labels, value labels, and other aspects of the data that might be lost if the data were converted to a pure ASCII file.

Working with stored data

- If you have saved your data in Stata binary format (in a `.dta` file), you employ the `use` *`filename`* command to make it the currently active datafile (or launch the file on a desktop version of Stata).
- If you want to `merge` files, both must be in `.dta` format, and you must use `sort` to arrange each dataset according to the order of the merge variable(s). Stata can handle one-to-one, one-to-many, and many-to-one merges.

Language syntax

- Stata commands follow a common syntax:

  `[by varlist]` *`command`* `[`*`varlist`*`] [=`*`exp`*`]`
  `[if exp] [in range] [weight]`
  `[,`*`options`*`]`

  where [•] indicate optional qualifiers.
- *command* is a Stata command
- *varlist* is a list of variable names
- *exp* is an algebraic expression
- *options* denotes a list of options

Language syntax

- *varlist* may be optional; if none is given, `_all` is assumed. Commands that alter or destroy data require an explicit *varlist*.
- For instance, the command `'summarize'` without additional arguments gives descriptive statistics for all currently defined variables.
- The command `'drop price mpg'` will remove those variables; `'drop _all'` is required to remove all currently defined variables.

Language syntax

- The `by` *`varlist:`* prefix may be used with many commands to instruct Stata to repeat the command for each value of the *varlist* (which will sensibly be comprised of integer-valued numeric variables and/or string variables). This is very powerful; e.g.
  `by race sex : summ income`
  will generate descriptive statistics for each combination of the two categorical variables.
- The dataset must be `sort`ed by the variables of the *varlist* prior to use of the `by` *`varlist:`* prefix.

Language syntax

- The `if` *exp* qualifier restricts the scope of the command to that of the logical expression. This can be used to evaluate a subset, e.g., to run a regression on only black males, or to construct a dummy variable conditional on certain features of the data.
- Note that logical expressions make use of '==' for equality, '&' for the AND operator, '|' for the OR operator, '!=' for the NOT operator. The words AND, OR, NOT are not used in Stata syntax.

Language syntax

- The `in` *`range`* qualifier restricts the scope of the command to a specific observation range:
  `1/10` denotes observations 1 through 10;
  `-5/-1` denotes the last five observations.
- This qualifier may be used to examine a few observations, or to pick out the top *n* observations after sorting the data.
- The `in` *`range`* qualifier may not be used in conjunction with the `by` *`varlist`*`:` prefix.

Language syntax

- The `=exp` expression is most often used in creating new variables, via the generate (gen) command.
- If an existing variable is to be modified, the replace command must be used, and replace cannot be abbreviated.
- Creating a dummy variable often requires both:

```
gen down = 1 if gdpgro < 0
replace down = 0 if gdpgro >= 0
```

- The `egen` command provides an additional and user-expandable set of functions

Language syntax

- Weights may also be applied in the context of many commands. Several different weighting schemes (analytic weights, frequency weights, sampling weights, importance weights) are available.
- Weighting is most commonly applied when working with survey data or cross-sectional data that represent groupings of microdata.

Language syntax

- Most commands take command-specific options. All options appear at the end of the command after a single comma.
- Options generally have default values. Many are toggles, with values of opt or noopt, such as
  `summarize price mpg, detail`
  which will generate extended descriptive statistics. The default choice is `nodetail`, which thus need not be given.
- Some options take numeric or string arguments.

File handling

- File extensions usually employed (but not required) include:
- `.ado` automatic do-file (procedure)
- `.dct` data dictionary
- `.do` do-file (user program)
- `.dta` Stata binary dataset
- `.gph` graph output file (binary)
- `.log` log file (text)
- `.raw` ASCII data file

File handling

- If you employ the common file extensions, you need not use them explicitly in Stata commands;
- `infile using mydat`
  presumes `mydat.raw`
- `save myfile, replace`
  presumes `myfile.dta`
- `do myprog`
  presumes `myprog.do`
- Exception: ado-files must have filetype `ado`.

File handling

- For ease of use, Macintosh Stata users should employ menu commands such as File->Open and File->Filename… to access files on the hard disk.
- The default Macintosh hard disk name–Macintosh HD–is problematic. Give the hard disk a name that does not contain embedded spaces.
- All desktop users should make use of an `ado` directory to store ado-files that are not distributed by Stata or STB. The `adopath` command specifies the expected location of this directory.

Data characteristics

- Stata stores data in three integer datatypes: `byte`, `int`, `long` and two floating point formats: `float` and `double`. There is also a `date` datatype.
- Stata handles `string` data, with varying-length strings (max 80 bytes), and the missing string “”.
- To use categorical variables in statistical routines, the encode command may be used to transform, e.g., sex={‘male’,’female’} via
  `encode sex, gen(gender)`
  where gender will now be a dummy variable.

Data characteristics

- If a string variable `'myvar'` contains the character representation of a number, it may be converted to a numeric variable via
  `gen newvar = real(myvar)`
  or via the more powerful `'conv2num'` command (an STB addition).
- A numeric variable may be converted to its character string equivalent via
  `gen str10 svar = string(newvar)`
  which should reproduce myvar.

Data characteristics

- Each variable may have its own default display format. This does not govern the contents of the variable, but affects how it is displayed. For instance, `%9.2f` would generate ‘dollars and cents’, like the FORTRAN format element F9.2. The command
  format varname formatspec e.g.
  `format gdp %9.1f`
  will store that display format with the variable in the Stata dataset.

Data characteristics

- Each variable may have its own variable label: a 31-character string which describes the variable, associated with the variable via `label variable varname "text".`
- Variable labels, where defined, will be used to identify the variable in printed output, space permitting.

Data characteristics

- Value labels associate numeric values with character strings; if a mapping is defined as

```
label define sexlbl
          0 “male” 1 “female”
```

  then a numeric (dummy) variable sex may be given value labels via

```
label values sex sexlbl
```

- If value labels are present, they will appear on printed output rather than their numeric equivalents.

Data characteristics

- Value labels may be generated when reading data if string variables take on specific values:

```
infile empno sex:sexlbl salary using
   empfile, automatic
```

The `sexlbl` qualifier indicates, in conjunction with the `automatic` option, that a set of value labels are to be defined as `'sexlbl'` from the discrete character values read from the `'sex'` variable. In this case `sex` becomes a numeric variable, with associated value labels as defined by `sexlbl`.

Functions

- Functions appear in expressions in the `generate`, `replace`, and `egen` statements in which variables are created.
- Functions also appear in the `if` *`exp`* qualifiers of many commands in which logical expressions are used to constrain the command.
- `+, -, *, /` have their usual meaning; ^ denotes exponentiation.
- + is also used as the concatenation operator for strings.

Functions

- Relational operators include >, >=, <, <=, == for equality, and ~= for inequality. '!=' may also be used for inequality.
- The most common error in constructing `if` *exp* qualifiers is the use of '=' where '==' is appropriate.
- Logical operators include & for AND, | for OR, and ~ for NOT. The words are not used. There is no exclusive OR (XOR) operator.

Functions

- Mathematical functions include the usual: `abs(), exp(), ln(), log()` [both referring to natural log], `mod(x,y), sqrt(),` and the trigonometric functions (with arguments in radians).
- The `lnfact(n)` function is the natural log of *n!.*
- The `mod(x,y)` is *x* modulo *y*.
- Statistical functions include those for binomial, χ^2, F, gamma, beta, normal, t, and uniform distributions and their inverses, as well as cumulative distribution functions for many distributions.

Functions

- Date functions allow manipulation of the portions of a date variable, and the calculation of elapsed time.
- String functions permit detection of a character or substring within a string, extraction of first, last, or specified substring, trimming, and case conversion.
- Special functions include coding (creating brackets of a numeric variable), integer truncation/floating point conversion, min, max, round, sign, and sum (the running sum of a variable).

Functions

- The `egen` command (extensions to generate) provides a number of special functions, including those which operate on a set of variables: for instance, row sums (`rsum()`), row means (`rmean()`), row standard deviations (`rsd()`). The `egen` command also computes percentiles, medians, and ranks.
- The `egen` command may be user-extended, as all `egen` functions are implemented as *`_gfunc`*`.ado` files.

Estimation commands

- All estimation commands share the same syntax.
- Multiple equation estimation commands use an *eqlist* rather than a *varlist*, where equations are defined prior to estimation via the `eq` command.
- Estimation commands display confidence intervals for the coefficients; the `level()` option controls the width of the intervals.
- The variance-covariance matrix of the estimators may be retrieved with the `vce` command.

Estimation commands

- Predicted values and residuals may be obtained after estimation with the `predict` command. The fit command may be used as an alternative to `'regress'` to generate a number of influence measures.
- After estimation, coefficients and standard errors may be used in expressions; `_b[income]` is the estimated coefficient on income in the last regression, while `_se[income]` refers to its estimated standard error.

Estimation commands

- Linear hypothesis (Wald) tests on the estimated parameters may be performed with the `test` command. The `'nltest'` command provides Wald tests of nonlinear hypotheses, while the `'lrtest'` command performs likelihood ratio tests.
- Robust (Newey/West, Huber/White) 'sandwich' estimates of the variance-covariance matrix are available for many estimation commands by specifying the `robust` option.

Estimation commands

- OLS estimates may be generated by the `regress` command, where the *varlist* contains the dependent variable followed by independent variables. A constant term is supplied by default; `nocons` suppresses the constant term.
- Following regression, `predict` *`yhat`* will generate the (in-sample) predicted values as variable *yhat*. Out-of-sample predictions may be generated by using an `if` *`exp`* or an `in` *`range`* qualifier to select observations not used in the estimation.

Estimation commands

- The `predict` command can also be used to generate estimated residuals (in- and out-of-sample), as well as standardized residuals, the standard error of the prediction, and the standard error of forecast.
- `predict` may be used following many estimation commands–not merely OLS regression. For instance, it may be used to predict estimated probabilities following estimation of a binomial logit model.

Estimation commands

- Basic statistical commands:
- `summarize`: descriptive statistics
- `table, tabsum, tabulate`: tables of summary statistics and frequencies
- `anova`: analysis of variance and covariance
- `oneway`: one-way analysis of variance
- `correlate`: correlations, covariances
- `ttest`: mean comparison tests

Estimation commands

- Regression commands:
- `regress`: OLS, IV, 2SLS regression
- `predict, fit`: predictions, fit diagnostics
- `cnreg`: censored-normal and Tobit models
- `nl`: nonlinear least squares
- `xtreg, xtgls, xtgee`: panel data models
- `glm`: general linear models
- `heckman`: Heckman's selection model

```
. use  ":Keewaydin:Stata:auto.dta"
(1978 Automobile Data)

. summ
```

Variable	Obs	Mean	Std. Dev.	Min	Max
make	0				
price	74	6165.257	2949.496	3291	15906
mpg	74	21.2973	5.785503	12	41
rep78	69	3.405797	.9899323	1	5
hdroom	74	2.993243	.8459948	1.5	5
trunk	74	13.75676	4.277404	5	23
weight	74	3019.459	777.1936	1760	4840
length	74	187.9324	22.26634	142	233
turn	74	39.64865	4.399354	31	51
displ	74	197.2973	91.83722	79	425
gratio	74	3.014865	.4562871	2.19	3.89
foreign	74	.2972973	.4601885	0	1

```
. ttest mpg,by(foreign)

Two-sample t test with equal variances      Domestic: Number of obs =       52
                                             Foreign: Number of obs =       22

------------------------------------------------------------------------------
Variable |      Mean    Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
Domestic |  19.82692     .657777   30.1423   0.0000       18.50638    21.14747
 Foreign |  24.77273     1.40951   17.5754   0.0000       21.84149    27.70396
---------+--------------------------------------------------------------------
    diff | -4.945804    1.362162  -3.63085   0.0005      -7.661225   -2.230384
------------------------------------------------------------------------------
Degrees of freedom: 72

                Ho: mean(Domestic) - mean(Foreign) = diff = 0

      Ha: diff < 0                 Ha: diff ~= 0                 Ha: diff > 0
        t =  -3.6308                 t =  -3.6308                 t =  -3.6308
    P < t =   0.0003         P > |t| =   0.0005         P > t =   0.9997

.
```

```
. regress price mpg hdroom displ foreign

  Source |       SS       df       MS                   Number of obs =      74
---------+------------------------------                 F(  4,    69) =   16.36
   Model |   309109608     4  77277401.9                 Prob > F      =  0.0000
Residual |   325955789    69  4723996.94                 R-squared     =  0.4867
---------+------------------------------                 Adj R-squared =  0.4570
   Total |   635065396    73  8699525.97                 Root MSE      =  2173.5

------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -113.1064   62.72059     -1.803   0.076      -238.2305    12.01778
  hdroom |  -613.7925   344.3983     -1.782   0.079      -1300.848     73.2633
   displ |    24.4052   4.699919      5.193   0.000       15.02911    33.78128
 foreign |   3529.337   702.0066      5.027   0.000       2128.872    4929.801
   _cons |   4547.006   2323.137      1.957   0.054      -87.52626    9181.537
------------------------------------------------------------------------------
```

```
. regress price mpg hdroom displ if foreign==0

  Source |       SS       df       MS                  Number of obs =      52
---------+------------------------------                F(  3,    48) =   16.06
   Model |   245079940     3  81693313.3                Prob > F      =  0.0000
Residual |   244114861    48  5085726.27                R-squared     =  0.5010
---------+------------------------------                Adj R-squared =  0.4698
   Total |   489194801    51  9592054.92                Root MSE      =  2255.2

------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -35.37898   104.8593     -0.337   0.737      -246.2128    175.4548
  hdroom |  -645.0291    397.923     -1.621   0.112      -1445.107    155.0487
   displ |   26.44172   5.597903      4.724   0.000       15.18638    37.69706
   _cons |   2628.467   3558.628      0.739   0.464      -4526.634    9783.567
------------------------------------------------------------------------------
```

Estimation commands

- A sampling of other estimation commands:
- `sureg, reg3`: Zellner's SUR/3SLS
- `qreg`: quantile (including median) regression
- `logit, probit`: binomial logit/probit
- `ologit, oprobit`: ordered logit/probit
- `mlogit`: multinomial logit
- survival analysis models
- `bstrap`: bootstrap sampling
- `ml`: maximum likelihood estimation

```
. tab rep78 foreign

Repair     | Car type
Record 1978|  Domestic    Foreign |     Total
-----------+----------------------+----------
          1 |         2          0 |         2
          2 |         8          0 |         8
          3 |        27          3 |        30
          4 |         9          9 |        18
          5 |         2          9 |        11
-----------+----------------------+----------
      Total |        48         21 |        69

. gen bestrep = rep78==5

. tab bestrep foreign

           | Car type
   bestrep |  Domestic    Foreign |     Total
-----------+----------------------+----------
         0 |        50         13 |        63
         1 |         2          9 |        11
-----------+----------------------+----------
     Total |        52         22 |        74
```

```
. logit bestrep price mpg foreign

Iteration 0:  Log Likelihood =-31.106481
Iteration 1:  Log Likelihood =-22.171647
Iteration 2:  Log Likelihood =-20.124399
Iteration 3:  Log Likelihood =-20.031529
Iteration 4:  Log Likelihood = -20.03006
Iteration 5:  Log Likelihood =-20.030059

Logit Estimates                                   Number of obs =     74
                                                  chi2(3)       =  22.15
                                                  Prob > chi2   = 0.0001
Log Likelihood = -20.030059                       Pseudo R2     = 0.3561

------------------------------------------------------------------------------
 bestrep |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   price |   .0001738   .0001754      0.991   0.322        -.00017    .0005175
     mpg |   .2077759   .0921829      2.254   0.024       .0271006    .3884511
 foreign |   2.155065   .8933987      2.412   0.016       .4040353    3.906094
   _cons |  -8.792186   3.065257     -2.868   0.004      -14.79998   -2.784393
------------------------------------------------------------------------------
```

```
. ologit rep78 price mpg foreign

Iteration 0:  Log Likelihood =-93.692061
Iteration 1:  Log Likelihood =-78.391154
Iteration 2:  Log Likelihood =-77.587155
Iteration 3:  Log Likelihood =-77.567314
Iteration 4:  Log Likelihood =-77.567278

Ordered Logit Estimates                           Number of obs =     69
                                                  chi2(3)       =  32.25
                                                  Prob > chi2   = 0.0000
Log Likelihood = -77.567278                       Pseudo R2     = 0.1721

------------------------------------------------------------------------------
       rep78 |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
       price |   .0000931   .0000921      1.011   0.312      -.0000874    .0002735
         mpg |   .0983342   .0588177      1.672   0.095      -.0169465    .2136148
     foreign |   2.455968   .6850463      3.585   0.000       1.113301    3.798634
-------------+----------------------------------------------------------------
       _cut1 |  -.7119252   1.656026            (Ancillary parameters)
       _cut2 |   1.090715   1.540291
       _cut3 |   3.719323   1.580065
       _cut4 |    5.80092   1.688265
------------------------------------------------------------------------------
```

Timeseries

- In Version 6, Stata has added a broad set of timeseries capabilities.
- The `tsset` command allows specification of the timeseries calendar associated with a dataset. Dates may be displayed in many formats. See `tsmktim` on SSC-IDEAS to set up a timeseries dataset.
- Annual, half yearly, quarterly, monthly, weekly, and daily frequencies are supported.

Timeseries

- Functions `tin(d1,d2)` and `twithin(d1,d2)` permit specification of date ranges for transformations and analysis.
- Lagged (and led) values or differences of timeseries data may be specified 'on the fly': e.g.

```
regress gdp L(1/4).gdp
regress D.gdp gdp
```

will run an AR(4) model and the Dickey-Fuller regression, respectively.

Timeseries

- Timeseries estimation commands include `arch`, `arima`, `dfuller` / `pperron`, `corrgram` / `xcorr`, and `wntestq` (Q-test).
- Timeseries commands available from SSC-IDEAS include `arimafit`, `durbinh`, `bgtest` (Breusch-Godfrey test for autocorrelation), and `gphudak` (long memory estimator), with more under development.

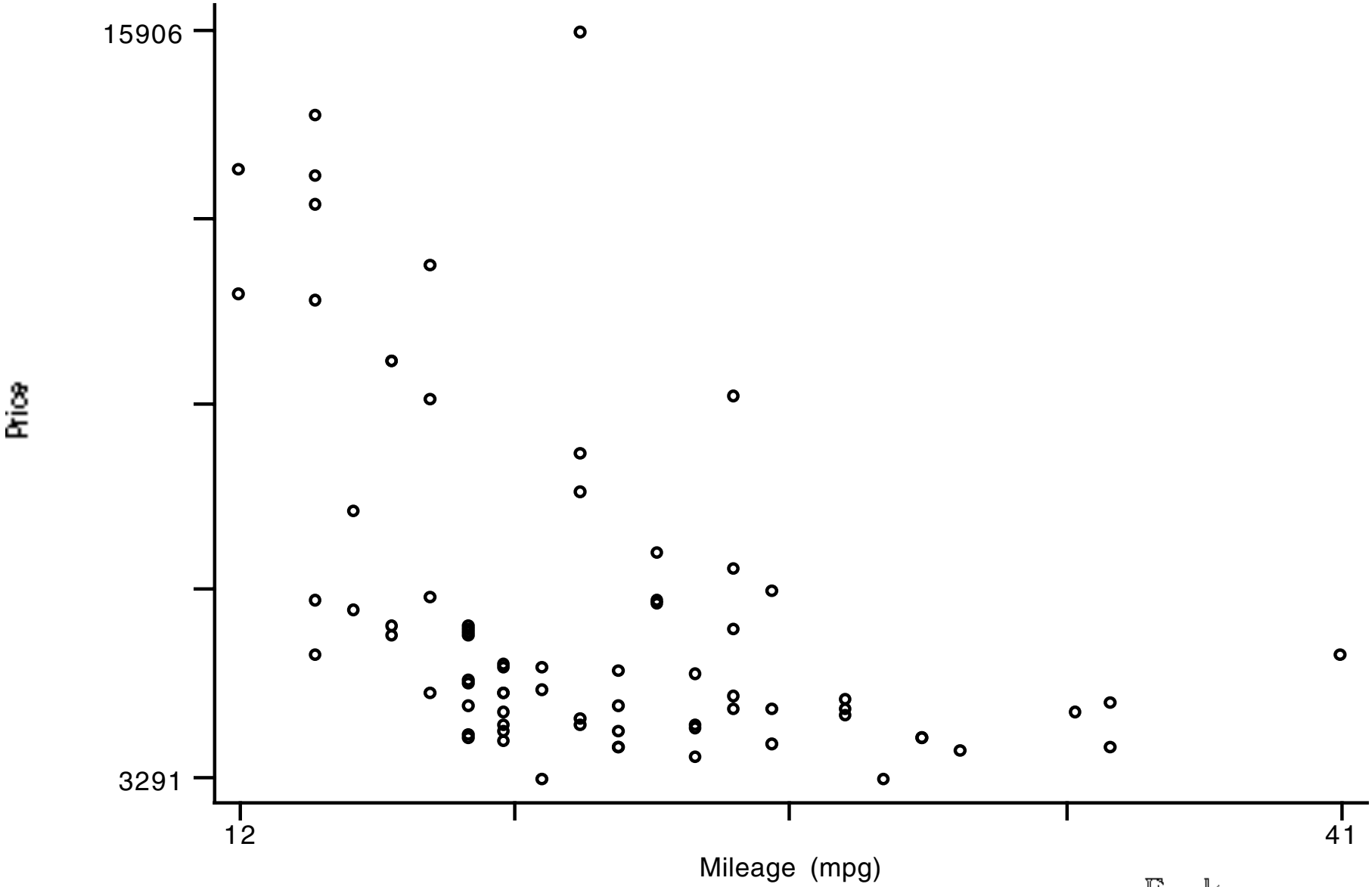
Graphics

- Stata contains extensive graphics capabilities for the generation of bar, box, pie, and star charts, as well as histograms, one-way scatterplots, and two-way scatterplots.
- Stata's capability to juxtapose many graphs on the same output screen is often helpful in exploratory data analysis.
- Graphs may be customized extensively to produce camera-ready output for inclusion in research papers.

Graphics

- Although the character-mode UNIX Stata is capable of generating the same graphics as the desktop (Mac or Windows) versions, they cannot be viewed on screen, but only saved to a file in PostScript format and sent to the printer from a telnet session. To view graphics, Xwindows must be used to run Stata. For details, see the UNIX Stata logon screen.

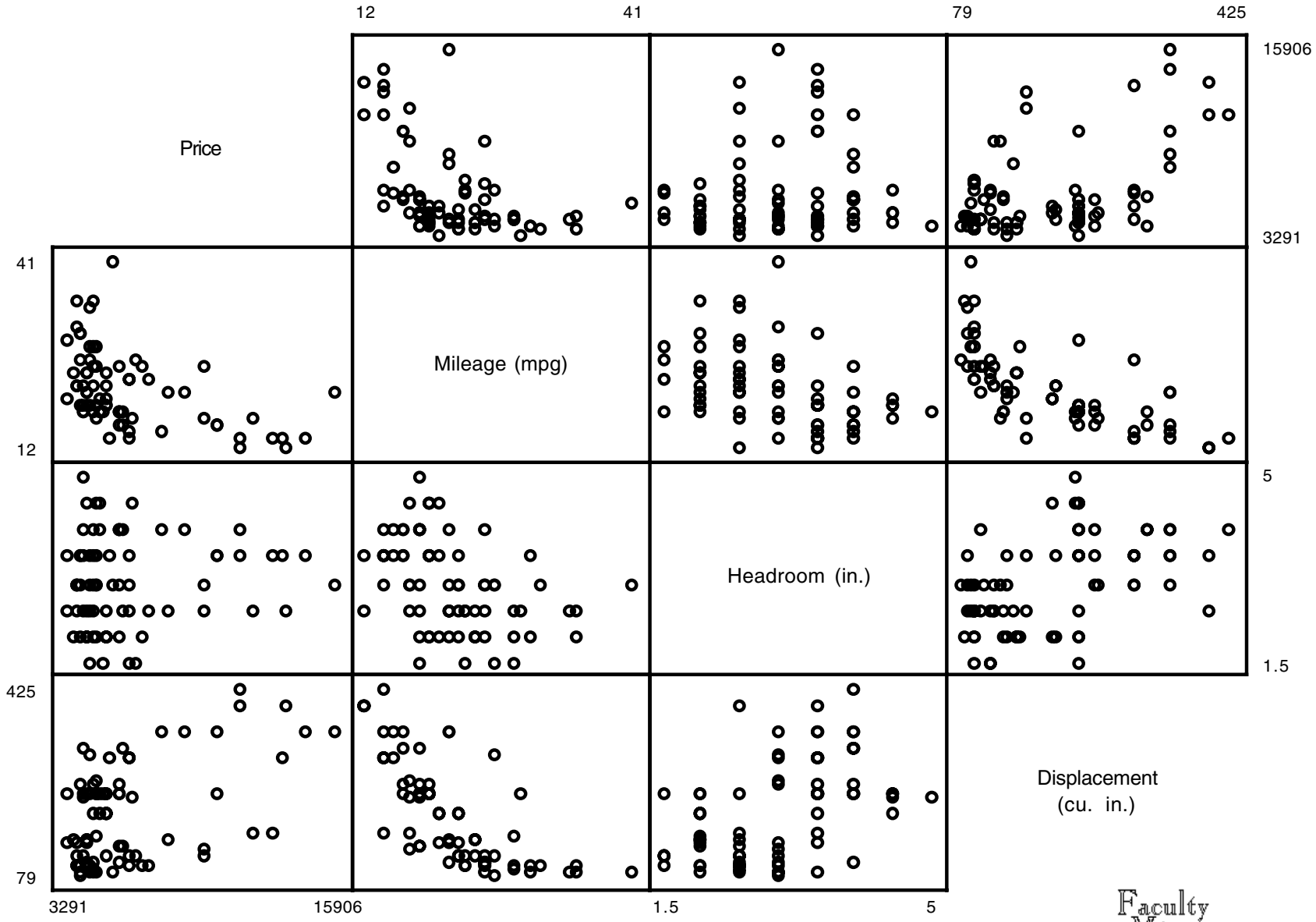
```
graph price mpg
```



September 4, 1999

Faculty Microcomputer Resource Center

```
graph price mpg hdroom displ, matrix
```



Matrix commands

- Stata contains a full-featured matrix language, with one limitation: matrices cannot be larger than a maximum dimension.
- The matrix language allows any estimation results to be stored in matrices and manipulated.
- Matrix functions include the singular value decomposition (SVD) and the eigensystem of a symmetric matrix.

Programming

- The Stata programming language allows users to readily incorporate new features into Stata. Stata 'procedures' are ASCII text, contained in `.ado` files and documented in associated `.hlp` files, and may be obtained from the STB, from StataList, and the SSC-IDEAS Archive maintained at B.C.
- The investment required to write Stata procedures to perform useful tasks is quite modest. High-level elements permit the handling of variable lists, error conditions, and options on user-written commands.

```
. program define fracrow
  1. /* expresses matrix elements as fraction of its row sums, in place */
. version 5.0
  2. local em "`1'"
  3. local a=rowsof(`em')
  4. local b=colsof(`em')
  5. tempname ones rowsum
  6. mat `ones'=J(`b',1,1.0)
  7. mat `rowsum'=`em'*`ones'
  8. local i=1
  9. while `i'<=`a' {
 10.   local j=1
 11.   while `j'<=`b' {
 12.      matrix `em'[`i',`j'] = 100.0 * `em'[`i',`j'] / `rowsum'[`i',1]
 13.      local j=`j'+1
 14.   }
 15.   local i=`i'+1
 16. }
 17. end

.
```

```
. matrix testb=(1,2,3,1\4,5,6,4\7,8,9,7)

. mat list testb

testb[3,4]
    c1  c2  c3  c4
r1   1   2   3   1
r2   4   5   6   4
r3   7   8   9   7

. fracrow testb

. mat list testb

testb[3,4]
           c1         c2         c3         c4
r1  14.285714  28.571429  42.857143  14.285714
r2  21.052632  26.315789  31.578947  21.052632
r3  22.580645  25.806452  29.032258  22.580645
```

In this case, a user has a datafile with several hundred variables which have been mistakenly characterized by 'insheet' as string variables rather than numeric variables. Encode or the STB-provided conv2num could be used to correct this, but each will only handle one variable at a time. This program allows for the syntax 'makenum firstvar-lastvar', as it will parse that list of variables, apply conv2num to each, replace '-1' with the missing data indicator, and summarize the resulting numeric variable. A straightforward use of Stata programming and its high-level functionality to handle arguments to user procedures.

```
prog def makenum
local varlist "req ex min(1)"
parse "`*'"
parse "`varlist'",parse(" ")
while "`1'" ~="" {
conv2num `1'
replace `1'=. if `1'==-1
summ `1'
mac shift
}
end
```

Getting Stata

- For students, faculty and staff at Boston College, desktop Stata (Mac/PowerMac, Win9x/NT, or Linux) is available through the Stata GradPlan at a substantial discount from the standard academic price, with shipping costs waived. Ask me for a Stata GradPlan brochure for details of the options available for software and documentation. Orders are usually fulfilled within two business days after you fax your order with payment to StataCorp.