EViews Quick Reference

inspired by the R Reference Card . . .

This 'Quick Reference' shows only a small selection of EViews commands and functions, for a more extensive summary see "Quick Help Reference" in the EViews Help menu. The complete manual is available as pdf in the EViews Help menu.

Workfile, Pages, Input & Output

`cd` change default directory, e.g. `cd` *"c:\mydata"*;

`wfcreate` Create a new workfile;
  Cross section:
  `wfcreate u` *num_observations*
  e.g.: `wfcreate`(wf=mywf) u 60
  Time series and panel data:
  `wfcreate` *frequency start end [num_cross_sect]*
  frequencies: `a` annual, `q` quarterly, `m` monthly, . . . ;

`wfopen` opens EViews workfile; can also open foreign file formats, use *options* `type` = [stata, spss, excel, . . .]

`wfsave` save workfile

`pagecreate` creates new page within existing workfile, e.g.

`pageappend`

`pagestruct` assign a structure to the active workfile page;

`pageload`

`pagecopy`

`pagecontract` *smpl_spec*

`pagedelete`

`pagesave`

`smpl` sets the workfile sample to use for statistical operations and series assignment expressions;
  Special keywords: `@all`, `@first`, `@last`;
  Examples:
  `smpl` 1986 2003; or `smpl` 1986 @last;
  `smpl` @all `if` *x1* > 0
  Resetting sample: `smpl` @all

`show` show object window

`print` print view

`freeze` create view object (graph or table)

`output` redirect printer output

`read`

`write`

Auxiliary Commands

`rename` guess what

`delete` guess what

`copy` also for frequency conversions or merging data

`@expand` may be added in estimation to indicate the use of one or more automatically created dummy variables.

Objects

For extensive help on objects see EViews Help → Quick Help Reference → Object Reference

`series` Series of numeric observations; Accessing individual values: ser`(i)` *i*-th element of the series *ser* from the beginning of the workfile.
  `@elem`(*ser, j*) function to access the *j*-th observation of the series *ser*, where *j* identifies the date or observation.
  Example: `series` *x2* = *x1* − `@mean`(*x1*)

  Creating dummy varables:
  `series` *D1* = *condition* produces dummy variable which is 1 if condition is fullfilled and else 0, e.g.
  `series` *D1* = @all if x1 >= @mean(*y*)

  `frml` create numeric series object with a formula for auto-updating.

`group` create a group object.
  Example: `group` *mygroup1 ser1 ser2 . . .*
  groups can also be used as regressor list, e.g.
  `ls` *y c mygroup1*

`equation` Equations are used for single equation estimation, testing, and forecasting.

  Equation Methods: `ls`, `tsls` (2-stage ls), `binary` (logit, probit), `ordered`, `arch`, `censored`, `gmm`, . . .

  Data Members: Scalar Values: `@aic` Akaike information criterion; `@coefcov`(*i,j*) covariance of coefficient estimates *i* and *j*; `@coefs`(*i*) *i*-th coefficient value; `@dw` Durbin-Watson statistic; `@f` F-statistic; `@meandep` mean of the dependent variable; `@ncoef` number of estimated coefficients; `@r2` R-squared statistic; `@rbar2` adjusted R-squared statistic; `@regobs` number of observations in regression; `@schwarz` Schwarz information criterion; `@sddep` standard deviation of the dependent variable; `@se` standard error of the regression; `@ssr` sum of squared residuals; `@stderrs`(*i*) standard error for coefficient *i*; `@tstats`(*i*) t-statistic value for coefficient *i*; `c`(*i*) *i*-th element of default coefficient vector for equation (if applicable).
  Vectors and Matrices: `@coefcov` covariance matrix for coefficient estimates; `@coefs` coefficient vector; `@stderrs` vector of standard errors for coefficients; `@tstats` vector of t-statistic values for coefficients.

`graph` Specialized object used to hold graphical output. Declaration either with `graph` or `freeze`
  Examples: `graph` *myline*`.line` *ser1*, or
  `freeze`(*myline*) *ser1*`.line`;
  `graph` *myscat*`.scat` *ser1 ser2*, or
  `group` *grp2 ser1 ser2*, `freeze`(*myscat*) *grp2*`.scat`

`table` Formatted two-dimensional table for output display. Declaration either with `table` or `freeze`
  Example:
  `table`(*2,2*) *mytable*   ' creates 2 × 2 table
  mytable(1,1) = "First row, first column"
  mytable(2,1) = "R2 = "
  mytable(2,1) = *eq01*.`@r2`

`sample` description of a set of observations to be used in operations.

`text` Object for holding arbitrary text information.

`scalar` a scalar holds a single numeric value. Scalar values may be used in standard EViews expressions in place of numeric values.

`vector` declare vector object (one dimensional array of numbers). *Example:* `vector`(10) *myvect1* ' creates myvect1 with 10 rows.

`rowvector` declare rowvector object. *Example:* `rowvector`(10) *myvect2* ' creates myvect2 with 10 columns.

`coef` coefficient vector. Coefficients are used to represent the parameters of equations and systems.

`matrix` declare matrix object (two-dimensional array). Accessing elements: Simply append "(i, j)" to the matrix name (without a '.')
Examples: `matrix`(*10,3*) *results*;
`matrix` *covmat1*=*eq1*.`@coefcov`

`sym` symmetric matrix (symmetric two-dimensional array).

`alpha` Alpha (alphanumeric) series. An EViews alpha series contains a set of observations on a variable containing string values.

`system` System of equations for estimation. *System Methods:* `3sls`, `3sls`, `arch`, `fiml`, `gmm`, `ls`, `sur`, `tsls`, `wls`, `wtsls`.

`var` Vector autoregression and error correction object.

`model` declare model object, a set of simultaneous equations used for forecasting and simulation

`logl` Likelihood object. Used for performing maximum likelihood estimation of user-specified likelihood functions.

Matrix Commands and Functions

`stom` series to matrix; converts a group to a matrix; `stom`(*groupname*, matrixname)

`mtos` matrix to series; converts a vector or matrix to a series or group; `mtos`(*matrixname*, *groupname*)

`@det` Calculates the determinant of a square matrix or sym.

`@transpose` Transposes matrix object.

`@inverse` Returns the inverse of a square matrix object or sym.; e.g.: let X be a $N \times K$ matrix and y a $N \times 1$ vector, then the coefficient vector b of the OLS regression (for $K = 3$) can be calculated by: `coef`(*3*) b = `@inverse`(`@transpose`(X)*X)*`@transpose`(X)*y

`@inner` Computes the inner product of two vectors or series, or the inner product of a matrix object; e.g. OLS estimator: `coef`(*3*) b = `@inner`(X)*`@transpose`(X)*y

Tests

`auto` [eq] Breusch-Godfrey LM tests for serial correlation in the estimation residuals.

`hettest` [eq] test for heteroskedasticity, can be a Breusch-Pagan-Godfrey (the default option), Harvey, Glejser, ARCH or White style test;

`white` [eq] White's test for heteroskedasticity of residuals;

`chow` [eq] Chow breakpoint or Chow forecast tests for parameter constancy;

`facbreak` [eq] factor breakpoint test for stability, using Dummy;

`reset` [eq] Ramsey's regression specification error test;

`statby` [series] statistics by classification;

`stats` [series, group] descriptive statistics table, e.g.: *wage*.`statby`(max,min) *sex race*;

`testadd, testadrop` [eq, panel] likelihood ratio test whether to add (drop) regressors to (from) an estimated equation;

`testby` [series] equality test by classification, e.g.: *wage*.`testby`(med) *race*;

`testbtw` [group] tests of equality for mean, median, or variance, between series in group

`teststat` [series] simple hypothesis tests;

`ubreak` [eq] Andrews-Quandt test for parameter stability at some unknown breakpoint;

`uroot` [series, group, panel] unit root test;

`wald` [eq] Wald test of coefficient restrictions for an equation object;

Panel & Pool

`auto` [eq] Breusch-Godfrey LM tests for serial

`pagestruct` assign a structure to the active workfile page;

`pagestack`

`pageunstack`

Descriptive Statistics

`stats` [series, group, vector, ...] Computes and displays a table of means, medians, maximum and minimum values, standard deviations, and other descriptive statistics of one or more series or a group of series.

`freq` [series, alpha] performs a one-way frequency tabulation. The options allow you to control binning (grouping) of observations.

`hist` computes descriptive statistics and displays a histogram for the series.

`boxplot` [series, group, vector, ...] display boxplots for each series or column.

Some Functions

`@abs`(x) absolute value of x.

`@obs`(x) the number of non-missing observations for x in the current sample.

`@mean`(x) average of the values in x.

`@median`(x) median of the values in x.

`@quantile`(x, q) the q-th quantile of the series x.

`@min`(x) minimum of the values in x.

`@max`(x) maximum of the values in x.

`@d`(x) First difference, equals $x - x(-1)$

`@dlog`(x) First difference of the logarithm

`@exp`(x) exponential, e^x

`@log`(x) Natural logarithm

`@cor`(x, y) the correlation between x and y.

`@cov`(x, y) the covariance between x and y (division by N).

`@stdevp`(x) square root of the population variance

`@stdevs`(x) square root of the unbiased sample variance. Note this is the same calculation as `@stdev`

`@sum`(x) the sum of x

`@trend` trend variable.

`@iff`(s, x, y) returns x if condition s is true; otherwise returns y. Note this is the same as `@recode`.

Statistical Distribution Functions: There are four functions associated with each distribution. The first character of each function name identifies the type of function:

Function Type:	*Beginning of Name:*
Cumulative distribution (CDF)	@c
Density or probability	@d
Quantile (inverse CDF)	@q
Random number generator	@r

The remainder of the function name identifies the distribution, e.g. chisq, fdist, tdist, norm, unif, ...

Examples:

@runif(1,10) ... random number from uniform distribution between 1 and 10; @cfdist(x,v1,v1) ... cumulated F-distribution for x with v1 nominator degrees of freedom and v2 denominator df; @dchisq(x,v) ... density for χ^2 distribution for x with v df.

Programming: Commands & Control Structures

' (single apostrophe) comment character, instructs EViews to ignore all text following the apostrophe until the end of the line;

_ (underscore), continuation character, allows – used as last (!) character of a line – to continue a command on the next line;

`!` starting character of control variable (numeric!), e.g. `!pi = 3.14`

`%` starting character of string variable (enclosed in double quotes), e.g. `%name = "Herbert Stocker"`

`%0` – `%9` program arguments, special string variables that are passed to a program when the program is run.

{} for replacement variables; e.g. from `%x = "GDP"` and `equation eq1.ls {%x} c {%x}(-1)` EViews produces `equation eq1.ls GDP.ls c GDP(-1)`.

`if` **statement in a program** The if statement marks the beginning of a condition and commands to be executed if the statement is true. The statement must be terminated with the beginning of an ELSE clause, or an ENDIF.

```
if [condition] then
  [commands to be executed if condition is true]
else
  [commands to be executed if condition is false]
endif
```

`for` **loop in a program** The FOR loop allows you to repeat a set of commands for different values of a control or string variable.

```
for counter=start to end [step stepsize]
  [ commands ]
next
```

e.g. for numerical variables:

```
for !i = 1 to 15
  series scalex{!i} = x{!i}/!i
next
```

For string variables:

```
for %y gdp gnp ndp nnp
  equation {%y}trend.ls %y c {%y}(-1) @trend
next
```

`open` opens a program file, or text (ASCII) file.

`run` executes a program.

`exit` exit the EViews program.