

# Basic Stata

## OVERVIEW

### Conventions and general syntax

Stata commands follow a common syntax, which you can access by looking the *command* up:

```
help command
```

I use this syntax below and in the Stata exercises handout.

A command's syntax diagram shows how to type the command and indicates possible options (it also gives the minimal allowed abbreviations for items in the command).

- items in *courier* should be typed exactly as they appear
- items in *italics* represent arguments for which you are to substitute variable names, observation numbers, etc.
- options are specified at the end of the command, where a comma precedes the first option
- square brackets `[]` denote optional qualifiers
- underlining indicates the shortest abbreviations allowed

For instance, I can translate:

```
regress depvar [indepvars] [if] [, options]
```

in my data as:

```
reg logwage age age2 if male==1, robust
```

This is the general syntax (we will use much less!):

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

where as we said, the square brackets denote optional elements, *varlist* a list of variables, *command* a Stata command, *exp* an algebraic expression, *range* an observation range and *options* a list of options.

- Stata's syntax is case sensitive.

- varlist

If no variables are specified, the command is applied to all the variables in the dataset

*varB-varF* → all the variables stored in between (to see: desc; to change: order)

\* → 0 or more characters go here

edu\* → all variables whose names start with edu

\*78 → all variables whose names end with 78

- **if** (and **in**) allow to restrict the command to a specific subset of the data:

*if exp* selects observations based on specific variable values, which must satisfy the *if* condition(s)

==	!=	>	>=	<	<=	&		!
equal	unequal	larger than	larger or equal	smaller than	smaller or equal	and	or	not

Examples:

```
if wage<1000
```

```
if place=="Canada" & age!=.
```

- **by varlist: command**

repeat the command for each subset of the data for which the values of the variables in *varlist* are equal

If data not sorted by *varlist*:

```
bysort varlist
```

Example:

```
bysort foreign: summarize wage
```

- A dot (.) denotes a missing value.  
Note: a missing variable is always considered larger than any other value.
- commands but also variable names can be abbreviated, as long as no confusion arises
- to stop what Stata is doing: press the Break button
- to retrieve previous commands typed in: PgUp
- to delete a full line of a command: Esc
- Printing and preserving output

To save output in a log-file: → log using *myresults.log* [, replace]

To close the log-file: → log close

- clear

→ to start from a clean slate

Note: can also be used as an option in use *newdata*, clear

- Comments

\* this line is not executed (a line commencing with \* is ignored)

```
/* from here to below  
these lines  
are not executed */
```

- Long lines

One option is to comment out the line break:

```
quietly replace lnf = theta2- /*  
  */ ln(1+exp(theta1)+exp(theta2)+exp(theta3)+ /*  
  */ exp(theta4)) if treatment==3
```

- display

→ can be use as calculator, e.g.

```
di 3*6.8
```

→ can be used to re-display specific results, e.g.

```
summarize xvar  
di r(mean)*r(N)
```

## ACCESSING DATA

### Opening and saving Stata files

- opening a Stata file:  
`use filename [, clear]`
- saving a Stata file:  
`save filename [, replace]`  
`save, replace`

### Increasing memory

```
set memory #
```

### Looking at the dataset

- Describing the contents of the data  
`describe`
- Counting observations  
`count [if exp]`
- Listing data  
`list [varlist] [in range] [if exp]`

# DATA MANAGEMENT

## Dropping variables

```
drop varlist  
drop [varlist] if exp
```

Sometimes it's simpler to specify which ones to keep:

```
keep varlist [if exp]
```

## Generating and replacing variables

to modify the values of an existing variable:

```
replace oldvar = exp [if exp]
```

to create new variables:

```
generate newvar = exp [if exp]
```

Examples (note the abbreviations):

```
replace rate = rate*100  
replace age = 25 if age==250
```

```
g constant = 5  
g logw = log(wage)  
g age2 = age*age /* or: g age2 = age^2 */
```

Useful functions (see help functions):

```
log(), abs(), int(), round(), sqrt(), min(), max(), sum()
```

statistical functions: e.g.

```
normal(z) → returns the cumulative standard normal distribution at z  
normalden(z) → returns the standard normal density at z
```

## Accessing Stata output

From the last-run model:

`_b[varname]` → the coefficient of *varname*  
`_se[varname]` → the std error of the coefficient of *varname*

From the last-run command:

Estimation command: `e(name)`  
General command: `r(name)`

Such Stata output can be used as part of expressions in commands, in generating variables or in defining scalars, e.g.

```
summarize wage if male==1
g meanwage_male = r(mean) /* this generates a constant equal to the mean wage
                           for men */
g maxwage_male = r(max)

count if female==1
g number_fem = r(N)
```

A way to store scalars in Stata is:

```
scalar name = expression
```

```
summarize wage if male==1
scalar wmale = r(mean) /* this stores the mean wage for men in a scalar
                        called wmale */

regress wage male educ age age2
scalar returneduc = _b[educ] /* this stores the coefficient on education (i.e. the
                              wage return to education) in a scalar called returneduc */
```

Scalars can in turn be used in further expressions.

To see the value of all scalars defined in the session:

```
scalar list
```

# TAKING A FIRST LOOK AT THE DATA

## Summarising the data

```
summarize [varlist] [if exp]
```

→ no. of non-missing obs, mean, std deviation, min and max

```
summarize [varlist] [if exp] , detail
```

→ add quantiles , 4 smallest and largest values, variance, mean, skewness and kurtosis

## Exploratory data analysis

- Correlations

```
correlate [varlist] [in range] [if exp]
```

```
[, covariance] → instead of correlation coefficients
```

- Tables

1. One-way tables: frequencies

```
tabulate varname [if exp]
```

```
sum wage
```

```
tab age if wage>r(mean)
```

→ age distribution for above mean-wage earners

2. Two-way tables: frequencies and measures of association

```
tabulate var1 var2 [in range] [if exp]
```

```
[, row → relative frequency of that cell within its row
```

```
col → relative frequency of that cell within its column
```

```
nofreq → frequencies not displayed
```

```
all ] → display all measures of association:
```

Pearson chi2, likelihood-ratio chi2, Cramer's V, gamma, Kendall's tau-b  
(tests of the hyp that row and col variables are independent)

## Non-parametric analysis

Univariate kernel density estimation:

```
kdensity varname [if exp]
```

```
[, width (#)] → to specify the bandwidth
```

```
kdensity duration if duration<40, s(.) norm xlab
```

# ESTIMATION

## Overview

```
[by varlist:] command yvar xvarlist [if exp] [in range] [, options]
```

`if` and `in` define the estimation sub-sample

Note: in order not to clutter notation, in the following, they are omitted.

Useful options:

```
, robust          → robust standard errors (White correction for heteroskedasticity)
level(#)          → set significance level for confidence intervals. Default = 95
```

## OLS regression

```
regress yvar xvarlist [if exp] [, robust]
```

## Predicted values and residuals

```
predict newvar [, statistic: in particular
    xb          → default: predicted value of dependent variable
    residual] → the residuals
```

```
regr logw age age2 sex educat
predict fitted
predict resid, residual
```

## Hypothesis testing

### Testing differences in means

To perform a  $t$  test of the hypothesis that  $var1$  and  $var2$  have the same mean:

```
ttest var1 = var2
```

To perform a two-sample  $t$  test of the hypothesis that a variable  $var$  has the same mean within the two groups defined by  $groupvar$  (option `unequal` specifies not to assume equal variances in the two populations):

```
ttest var, by(groupvar) [unequal]
```

To perform a Hotelling T-squared test of the hypothesis that vectors of the means of variables  $varlist$  are equal within the two groups defined by  $groupvar$ :

```
hotelling varlist, by(groupvar)
```

## Linear hypothesis (Wald test) after regress

Note: `regress` already provides overall F test and individual t tests

```
test exp=exp
test coefficientlist
```

Note: for `test`, both *varname* and `_b[varname]` denote the coefficient on *varname*.

Examples:

```
regr logw age group sex edu2-edu4
test age group sex edu2-edu4
test group
test age=1
test 2*(age+sex) = -3*(edu2-(edu3+1))
test _b[x1]=0

test x1 x2 x3

test region1 region2 region3 region4    /* which is the same */
testparm region*                        /* or */
testparm region1-region4    /* testparm allows you to specify a varlist */
```

## **Instrumental Variables**

```
ivreg yvar [exogvarlist] (endogvarlist = IVvarlist)
```

→ estimates a linear regression model of *yvar* on *exogvarlist* and *endogvarlist* using *IVvarlist* (along with *exogvarlist*) as instruments for *endogvarlist*.

## **Binary qualitative outcome model**

```
probit yvar xvarlist [, robust]
```

→ estimate maximum-likelihood probit models of binary *yvar* on *xvarlist*

```
dprobit
```

→ same as `probit` but instead of reporting coefficients, it reports the change in the probability for an infinitesimal change in each independent, continuous variable and the discrete change in the probability for dummy variables