Short STATA Manual

Mario Larch
University of Bayreuth
Original version developed by Benedikt Heid

This version: December 14, 2016

What's the goal of this **(very short)** course?

- Get you to get to grips with STATA: Showing you what is possible.
- Enabling you to produce the empirical results for a research paper in STATA—from getting your data into STATA to getting your results out of STATA.

Content I

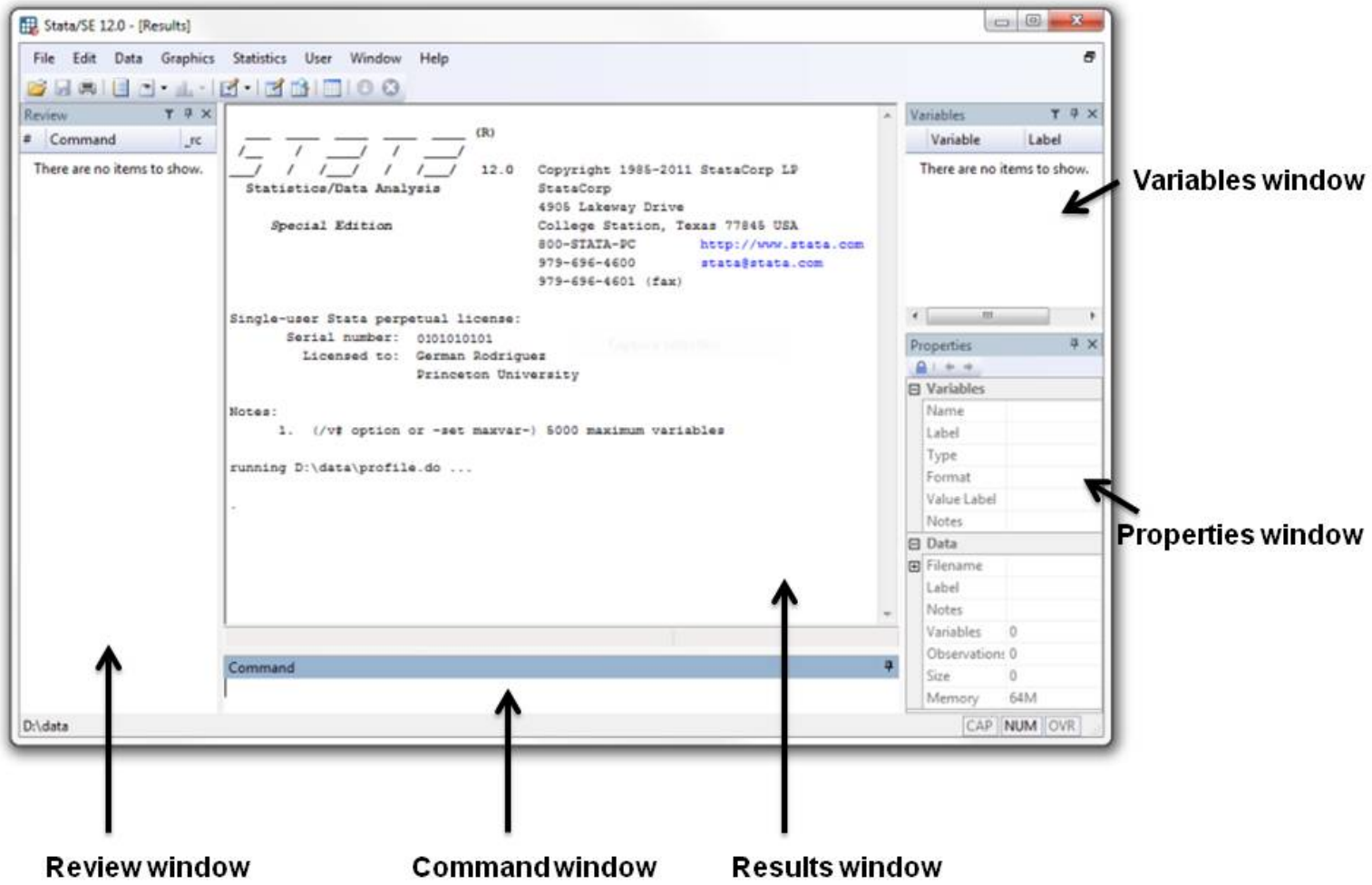
STATA – Data Analysis and Statistical Software I

- Proprietary all-purpose statistical software package.
- Proprietary $\Rightarrow$ expensive, you can only use it in the university PC pool (or you have to buy a license, student licenses are available, see for more information here).
- Allows analysis of cross-section, panel and time series data sets.
- Very strong in data handling and management, especially for surveys and panel data.
- Methods: all kinds of (non-)linear regression models.
- Also allows ANOVA, multivariate methods, structural equation modelling (SEM),...
- STATA is the de facto industry standard for statistical analysis of socio-economic data in economics.

STATA – Data Analysis and Statistical Software II

- Even though R, an open-source software, is getting more and more popular (https://www.r-project.org/).
- Allows to document your work using script-files (so-called do-files) $\Rightarrow$ Reproducibility ✓

STATA interface I



STATA interface II

- The *Command* window is where you type your commands.
- The *Results* window shows the results, immediately after typing the command or executing the do-file.
- The *Review* window shows the commands added to a list, so you can keep track of the commands you have used.
- The *Variables* window lists the variables in your dataset.
- The *Properties* window displays properties of your variables and dataset.

STATA interface III

- The toolbar contains icons that allow to *Open*  and *Save*  files.
- Once you open a dataset, a list of all variables appears in the window *Variables*.
- The Data Editor  and Data Browser  present you with a spread-sheet like view of the data.
- The Do-file Editor  allows you to construct a file of STATA commands and execute it in whole or in part from the editor.
- Watch out! STATA names are case-sensitive: $\texttt{X} \neq \texttt{x}$.
- In general, it is recommended to type all commands in a seperate do-file, instead of using the command window, to ensure **reproducibility** of your results.

STATA do-files

- A Do-file is a set of STATA commands typed in a plain text file.
- To access Stata's Do-file Editor click on  .
- Advantages:
  - Run your program directly from the editor.
  - Run just a few commands (marked section) or run the whole do-file.
  - Analysis is reproducible.
- One command per line. Use \ for line break.
- Use * for commenting out one line.
- Whole sections can be commented out by enclosing them in /* *section* */.
- Do-files can be saved  and executed  .

General command syntax

- General form:
  Command *varlist* [*if* exp], [*Option*]
- Example: **`summarize`** *wage* **`if`** *female==1*, **`detail`**

- Expressions in square brackets and options are optional.
- Note:
  `if` at the end of a command means the command is to use only the data specified. Equality tests are performed by two equal signs (==), **not** one (=).

Useful commands I

- Data and do-file editor
  - Open data editor: **`edit`**
  - Open data browser: **`browse`**
  - Open do-file editor: **`doedit`**
  - Open do-file: **`do`** *filename*
- Help function
  - Help menu: **`help contents`**
  - Look up commands in help menu: **`help`** [*command*]
  - **`search`** `topic` searches for `topic` in the help file

Content I

Reproducibility: Necessary for science!

- All the analyses you do have to be documented. Otherwise, no one can reproduce what you have done.
- Therefore: do-files! (documents the code)
- Second way for documentation: log-files (documents the code ***and*** the results)

Creating a log file

- **`capture log close`**
  - put this command before opening a log file to prevent errors.
- **`log using`** *`filename`* **`[, append replace [text|smcl]]`**
  - creates a log-file with *filename*.
- **`log close`**
  - closes the log-file.

Content I

Load and save data I

- **`clear`**
  - Removes dataset and clears memory.
- **`cd`** *`C:\folder\subfolder`*
  - Changes to the indicated working directory.
- **`pwd`**
  - Displays the path of the current working directory.
- **`ls`**
  - Displays the content of the current working directory.

Load and save data II

- **use** *filename.dta* [**,clear**]
  - Loads the indicated data set, while deleting the current data set.
- **save** *filename.dta* [**,replace**]
  - Saves the indicated data set, while replacing the old data set.

Describing the dataset

- **`describe`**
  - Produces a summary of the dataset in memory.
- **`list`**
  - Displays the values of variables.
- **`display`**
  - Displays all strings and values of scalar expressions.
  - Use **`display _N`** to display the number of lines in the data set.
  - Alternatively, **`display`** can be used as a substitute for a hand calculator, e.g. **`display`** 2+2.
  - You can abbreviate **`display`** to **`di`**.

Types of variables

- **Numeric** variables
  - All variables whose values are numbers.
  - Values are displayed in standard numeric format and are colored in `black`.
- **String** variables
  - All variables whose values are alphanumeric.
  - Values are displayed in `red` color.
  - To convert string varaibles to numeric variables and vice versa, use:

    `destring` *`varlist`*`,` **`replace`**
    `tostring` *`varlist`*`,` **`replace`**

Content I

Operators I

- STATA contains a number of basic mathematical and logical operators.
- Mathematical operators:
  - `+`: Summation, e.g. `2+3`
  - `-`: Subtraction, e.g. `7-3`
  - `*`: Multiplication, e.g. `25*3`
  - `/`: Division, e.g. `6/3`
  - `^`: Power, e.g. `2^3`
- The results of these operations can be displayed using **`di`** (the abbreviated command **`display`**).

Operators II

- Logical and relational operators:
  - `==`: EQUAL, e.g. `(2==3)`
  - `!=`: NOT EQUAL, e.g. `(11!=10)`
  - `<`: LESS THAN, e.g. `(8<8)`
  - `>`: GREATER THAN, e.g. `(8>9)`
  - `<=`: LESS OR EQUAL, e.g. `(8<=8)`
  - `>=`: GREATER OR EQUAL, e.g. `(10>=9)`
  - `&`: AND, e.g. `(3==3)&(2!=2)`
  - `|`: OR, e.g. `(3==3)|(2!=2)`
- Returns either 1 (true) or 0 (false).
- Operators are mainly used in combination with variables, e.g. `pareduc=meduc+feduc` or `urban=(population>100000)`.

How Does STATA treat missing values?

- Numeric missing values
  - STATA displays numeric missing values as " `.`"
  - Numeric missing values are represented by large positive values.
  - For instance, the expression **`age > 60`** is true if variable **`age`** is greater than 60 **or** missing.
  - To exclude missing values, ask whether the value is less than "`.`"
    Example: **`list if age > 60 & age < .`**
- String missing value
  - STATA has a string missing value, which is denoted by "(blank)".

Working with variables I

- **generate**
  - Creates a new variable, e.g. **generate** `NewVar=500`.
- **replace**
  - Changes the content of an existing variable, e.g. **replace** `VarName=ln(VarName)`.
- **drop** *Varlist*
  - Eliminates variables or observations.
- **keep** *Varlist*
  - Keeps variables ($\hat{=}$ columns) specified in *Varlist*, e.g. **keep** *var1*
- **keep if** *exp*
  - Keeps observations ($\hat{=}$ lines) that satisfy the specified condition *exp*, e.g. **keep if** *exp*

Working with variables II

- **rename** *Old_VarName New_VarName*
  - Changes the name of an existing variable.
- **sort** *Varlist*
  - Arranges the observations of the current data into ascending order based on the values of *Varlist*.
- **xtset** *Panelvar Timevar*
  - Declares data in memory to be a panel.
  - If you save your data after **xtset**, the data will be remembered to be a panel and you will not have to **xtset** again.

Working with variables III

- **generate** *DummyVar = exp*
  - Creates a Dummy variable,
    e.g. **generate** `male = sex ==1`.

Content I

Descriptive statistics

- **summarize** *Varlist*
  - Calculates and displays a variety of univariate summary statistics. If no *Varlist* is specified, summary statistics are calculated for **all** variables.
- **corr** *Varlist*
  - Displays the correlation matrix or covariance matrix for a group of variables.
- **tabulate** *Varname*
  - Produces one-way tables of frequency counts of *Varname*.
- **tabulate** *Var1 Var2*
  - Produces two-way tables of frequency counts of *Var1 Var2*. Use the option **cell** to also get relative frequencies.

Content I

Basic graphs of single variables

- **histogram** *Varname*
  - Draws a histogram of *Varname*. To set the number of bins and the width of bins use: **bin**(#) or **width**(#) as options.
- **kdensity** *Varname*
  - Draws a kernel density plot ($\approx$ smooth histogramm) of *Varname*. To set the bandwidth use the **bwidth**(#) option; you can set the kernel by **kernel**(kernelname).

Two-way graphs

- **[graph twoway] scatter** *yvar xvar*
  - Draws a scatter plot (twoway plottype).
- **[graph twoway] line** *yvar xvar*
  - Draws a line plot (twoway plottype).
- **twoway (scatter *yvar xvar*) (lfit *yvar xvar*)**
  - Draws a scatter plot and the according regression line (twoway plottype).

Exporting a graph

- **`graph export` *`graph1`*`.png, as(png) [replace]`**
  - exports the last created graph and saves it as *`graph1.png`*.

Content I

OLS regression analysis

- **reg** *yvar xvar*
  - Fits a model of *yvar* and *xvar* using simple **linear** regression.
- **reg** *yvar xvar*, *robust*
  - Fits a model of *yvar* and *xvar* using heteroscedasticity-robust standard errors.
- **reg** *yvar xvar*, vce(cluster *clustvar*)
  - Fits a model of *yvar* and *xvar* using standard errors clustered at *clustvar*.
- **reg** *yvar xvar1 xvar2 xvar3 ...*
  - For a linear regression on multiple independent variables use the same command and include all necessary x-variables.

Two-stage least-squares regression

- STATA also has implemented several estimators which use instrumental variables (IVs).
- A popular IV estimator is **two-stage least-squares (2SLS)**.
- Example: **`ivregress 2sls`** *`yvar(xvar=iv)`*`,`**`first`**
  - Fits a linear regression of *`yvar`* on *`xvar`* using the two-stage least-squares method and *`iv`* as instrumental variable. **`first`** displays the results from the first stage.

Time-series models

- STATA also has implemented standard time series models, e.g. ARMA and ARIMA. ARIMA are ARMA models on time series which are integrated (I) of order d. To estimate these models, you first have to declare your data-set to be a time-series data set. This is done by:
  tsset *timevar*
- **arima** *yvar, arima(p,d,q)*
  - Estimates an ARIMA(p, d, q) model
- **arima** *yvar, arima(1,0,2)*
  - Estimates an ARMA($1, 2$) model ($\equiv$ ARIMA(1,0,2)).

Hypothesis testing

- **test** *xvar*
  - Tests linear hypothesis $H_0 : \beta_{xvar} = 0$ **after** estimation.
- **test** *xvar==1*
  - Tests linear hypothesis $H_0 : \beta_{xvar} = 1$ **after** estimation.
- **test** *xvar1*
  test *xvar2*, accum
  - Tests linear hypothesis $H_0 : \beta_{xvar1} = \beta_{xvar2} = 0$ after estimation. *accum* allows to test a hypothesis jointly with a previously tested hypothesis.

t-distribution

- STATA can be used to calculate critical values and *p*-values manually.
- **`ttail`**`(n,t)`
  - returns the area under the right tail of the *t*-distribution with `n` degrees of freedom. In other words, it calculates `p` $= 1 - P(T \leq$ `t`$) \equiv 1 - F($`t`$)$. This can be used for calculating critical values manually.
- **`invttail`**`(n,p)`
  - returns the `p`-quantile of the *t*-distribution with `n` degrees of freedom. This can be used for calculating *p*-values manually.
- If `ttail(n,t)= p`, then `invttail(n,p) = t`.
- Remember: Display calculations using **`di`**.

Non-linear regression models and marginal effects I

- **probit** *yvar xvarvlist [, robust]*
  - Fits a Probit model for *yvar* and *xvarlist* (using heteroscedasticity-robust standard errors).

There are many more non-linear regression models: **logit, clogit, mlogit, poisson, nbreg, stcox, biprobit, ivprobit**, ...

Non-linear regression models and marginal effects II

- Most regressions are estimated to formulate a sentence like: "If x increases by 1 unit, y increases by β_k units"
  $\Rightarrow$ We are interested in the **marginal effect** of regressors.

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \varepsilon_i \\ \Rightarrow \text{marginal effect:} \frac{\partial y_i}{\partial x_{ki}} &= \beta_k \end{aligned}$$

- In the linear regression model, coefficients = marginal effects.

Non-linear regression models and marginal effects III

- In nonlinear regression models like e.g. the probit model, the coefficients are no longer the marginal effects! For linear index model it holds that

$$\begin{aligned} y_i &= g(x_i'\beta) \\ \Rightarrow \text{marginal effect:} \frac{\partial y_i}{\partial x_{ki}} &= \frac{\partial g(x_i'\beta)}{\partial (x_i'\beta)}\beta_k \neq \beta_k \end{aligned}$$

- In general, the marginal effect will depend on the value of the regressors! $\Rightarrow$ The marginal effect is no longer constant across all observations.
- In the probit model:

$$\begin{aligned} p_i \equiv P(y_i = 1) &= \Phi(x_i'\beta) \\ \Rightarrow \frac{\partial p_i}{\partial x_{ki}} &= \Phi'(x_i'\beta)\beta_k = \phi(x_i'\beta)\beta_k \end{aligned}$$

Non-linear regression models and marginal effects IV

- Therefore, every observation in the sample has its own marginal effects.
- Very often, the following summary measures are used:

marginal effect at the mean(s)	$\phi(\bar{x}'\beta)\beta_k$	`margins, dydx(*) atmeans`
average marginal effect	$\frac{1}{N}\sum_{i=1}^{N}\phi(x_i'\beta)\beta_k$	`margins, dydx(*)`
marginal effect at $x = x^*$	$\phi(x^{*\prime}\beta)\beta_k$	`margins, dydx(*) at(age=20 male=1)`

- Which measure should you use? It depends on the economic question you are interested in!

Content I

Programming STATA settings I

- **`set more [on off] [, permanently]`**
  - **`set more on`** tells STATA to wait until you press a key before continuing when a —more— message is displayed (default setting).
  - **`set more off`** tells STATA not to pause or display the —more— message.
  - **`permanently`** makes STATA remember the setting after you exit it.

Programming STATA settings II

- **quietly** *[command]*
  - Suppresses all terminal output for the duration of *command*.
  - **quietly {...}** surpresses terminal outputs for all commands enclosed in the braces.
- **noisily** *[command]*
  - Turns back on terminal output for the duration of *command*.
  - **noisily{...}** turns back on terminal outputs for all commands enclosed in the braces.
- **#delimit ;**
  very long
  command
  over several lines **; #delimit** cr
  - allows you to write a very long command in several lines, indicate the end of the command by ;.

Macros: local variables I

- One of the most powerful features of STATA is the possibility to use macros. **`local` *`localname`* *`localcontent`***
- You can reference the content of a local by **`` `localname' ``**.
- STATA will replace the string **`` `localname' ``** by the string **`localcontent`**.
- You can also calculate using locals. Then, you need the = sign.
  `local` *`localname`* `=` *`localcontent`*
  e.g. **`local i = `i' + 1`**
- If ***`localcontent`*** contains blank spaces, reference the local as "**`` `"`*`localname`*`"' ``**".

Macros: local variables II

- There are also global variables. Their usage is very similar to locals, however, they can be tricky—simply do not use them, everything can be solved by using locals as well.
- There are quite some subtleties concerning locals, and the usage of quotation marks ‘"’. For more information, please read the manual!!!

Loops

Very often, you want to execute the same operation on a set of variables, e.g. create log values for several variables. Then, loops can economize your code.

- **foreach** *c* **of local** *localname* **{**
  code using `c'
  }

Flow control

Very often, we want a program to perform a task when a certain condition is true, and perform another task when the condition is false. This can be done by an **`if/else`** statement.

```
if exp {
...
}
else {
...
}
```

If the expression is valid all the commands in the first braces are executed, if not then all the commands in the second braces are executed. The second set of braces is not necessary.

Accessing regression results for calculations

- **`r(...)`**
  - Returns the value of the saved result **`r(name)`**. Use the command **`return list`** to return results for general commands, stored in **`r(...)`**.
- **`e(...)`**
  - Returns the value of the saved estimation result **`e(name)`**. Use the command **`ereturn list`** to return results for estimation commands, stored in **`e(...)`**.
- **`_b[varname]`**
  - Accesses the estimated parameter value for the regressor *varname*.

Content I

Creating nice regression tables I

After having run a regression, you can save regression results using the following commands:

- **`est store name`**
  - Saves the results from the last regression as *name* (current session).

You can restore these results by

- **`est restore name`**

- **`est save filename [, append | replace]`**
  - Saves the results from the last regression as *name* (on hard drive in file *filename*, for new session).

You can reload these results in a new session by

- **`est use filename`**

Creating nice regression tables II

You can compare different regressions in one table by

- **`est table name1 name2, se stats(N r2 r2_a)`**
  - `se` also displays the standard errors, and `stats(N r2 r2_a)` returns the number of observations as well as the (adjusted) R^2. You can add any other result stored in `e()`.

While **`est table`** is nice, it does not help us with getting the table into LaTeX or WORD.
To solve this problem, we can for example use a package called **`outreg2`**.

Packages and how to install them

- STATA has implemented many estimators and useful commands. However, some problems cannot be solved without considerable programming effort.
- Solution: STATA packages $\Rightarrow$ Do not reinvent the wheel!
- **`search topic, all`**
  - allows you to search the online package repository.
- **`ssc install packagename`**
  - installs the package on your computer.

Creating nice regression tables III

```
# delimit ;
outreg2 [name1 name2] using "path/filename",
word excel tex(frag)
title("My regression table")
dec(3) replace ;
# delimit cr
```

Content I

- STATA comes with two matrix languages: `matrix` (old, few functions) and MATA (new, powerful).
- MATA is a fully developed matrix algebra environment.
- Similar to, but less powerful than, MATLAB or GAUSS.

- **`tomata` *`varlist`***
  - transfers STATA variables to MATA vectors.
- `mata`
  - enters MATA mode.
- `end`
  - ends MATA mode and returns to STATA.

OLS using matrix algebra and MATA

$$\hat{\beta}_{OLS} = (X'X)^{-1}X'y$$

```
mata
y = lwage
N = rows(y)
ones = J(N,1,1)
X = (ones,educ,age)
beta = luinv(X'*X)*X'*y
end
```

compare to results from

```
reg lwage educ age
```