EEGenerating Skills – Introduction to Stata

Miguel Portela[1,2]

[1]NIPE – UMinho
[2]IZA, Bonn

April, 2021

Outline

1. Introduction to Stata: from menus to 'do' files
2. Handling data with different data types: Stata, ASCII, Excel, CSV, Web Data and ODBC
3. Data Reshaping
4. Exploratory data analysis: descriptive statistics and graphic manipulation
5. Exploratory data analysis: descriptive statistics and graphic manipulation
6. From Stata to LaTex, Word and Excel: efficient procedures to export descriptive statistics, graphs and regression tables

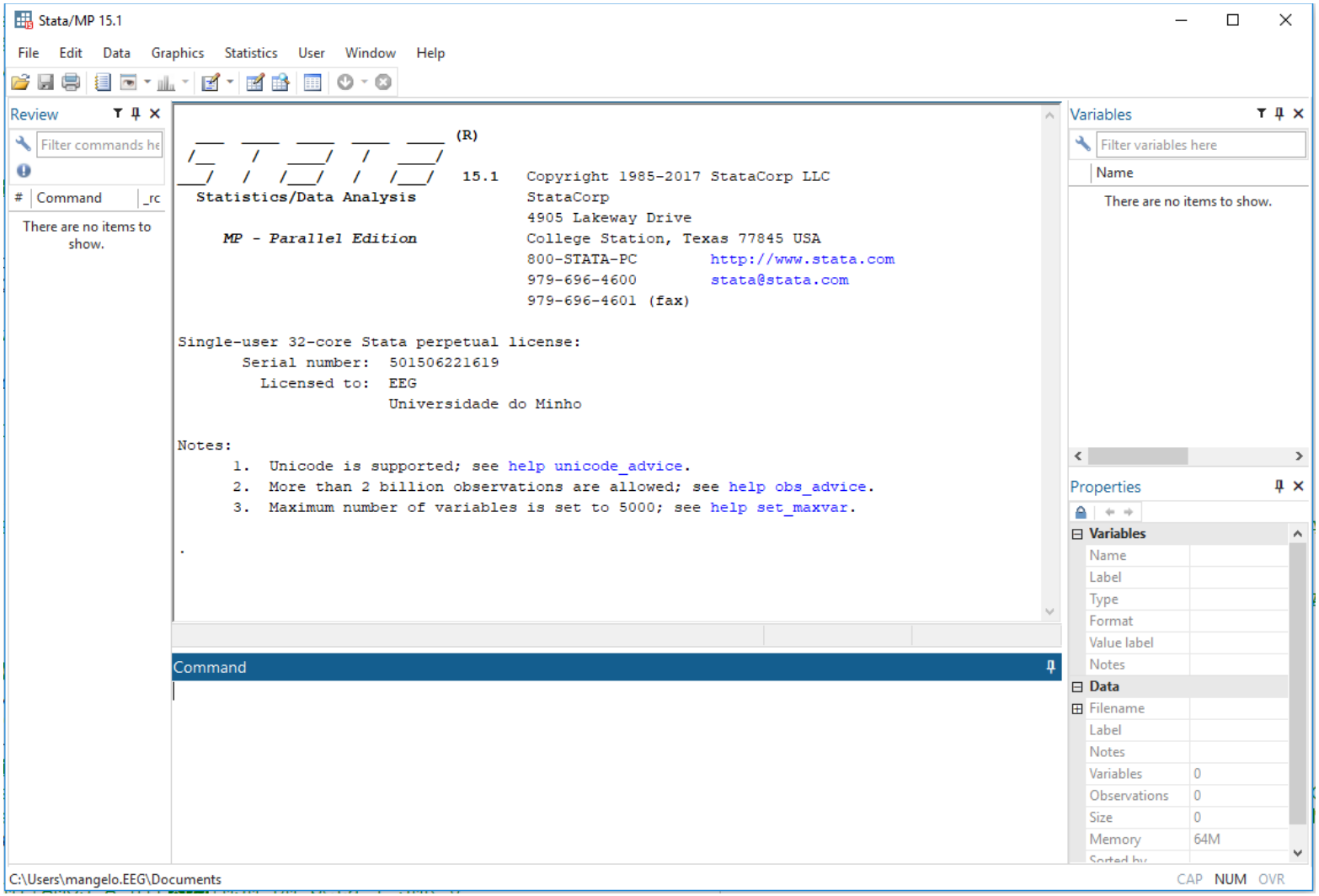
Introduction

- "Stata is a complete, integrated statistical package that provides everything you need for data analysis, data management, and graphics.", Stata website
- It is command-based software, and is available for Windows, Macintosh, and Unix systems
- Stata provides a highly flexible interactive mode that makes it easier for beginners to learn and use
- Stata supports features for programming and matrix manipulation

Key issues about Stata

- Starting Stata: **Start** > **All Programs** > **Stata 16**
- In order to be able to replicate your analysis use a 'do' file (`doedit`)
- Define a working directory (`cd`; `pwd`)
- Define a log file for your output (`log using`)
- You can learn 'command line' *commands* by using menus
- The data you want to use can be in ASCII (American Standard Code for Information Interchange) format, in Excel, CSV (comma-separated values), Access, SPSS, Gauss, Ox, ...
- Look to the data using both the menus and the command line (`browse`, `list`)
- Getting help: `help`, `search`, `findit`, `net search`
- Exiting Stata: `exit` or `exit, clear`

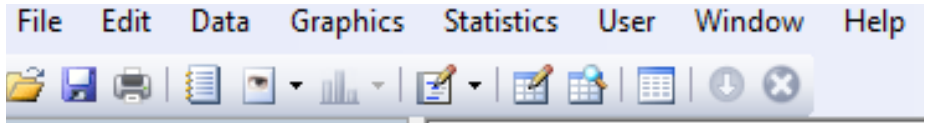
Looks


Stata/MP 15.1
File Edit Data Graphics Statistics User Window Help
Review
Filter commands he
Command _rc
There are no items to show.
(R)
15.1 Copyright 1985-2017 StataCorp LLC
Statistics/Data Analysis StataCorp
4905 Lakeway Drive
MP - Parallel Edition College Station, Texas 77845 USA
800-STATA-PC http://www.stata.com
979-696-4600 stata@stata.com
979-696-4601 (fax)
Single-user 32-core Stata perpetual license:
Serial number: 501506221619
Licensed to: EEG
Universidade do Minho
Notes:
1. Unicode is supported; see help unicode_advice.
2. More than 2 billion observations are allowed; see help obs_advice.
3. Maximum number of variables is set to 5000; see help set_maxvar.
.
Command
Variables
Filter variables here
Name
There are no items to show.
Properties
Variables
Name
Label
Type
Format
Value label
Notes
Data
Filename
Label
Notes
Variables 0
Observations 0
Size 0
Memory 64M
C:\Users\mangelo.EEG\Documents
CAP NUM OVR

Stata has several windows

1. **Command**: you can insert here specific commands that request tasks to Stata
2. **Results**: the main window at the center of the screen where Stata replies to your commands
3. **Review**: typically on the left side, keeps track of all commands you have issued – you replay the commands by double-clicking, or click once to edit them (previous commands can be retrieved also by using the 'PageUp' key in your keyboard in the commands window)
4. **Variables**: lists the variables in Stata's memory, together with its properties
5. **Properties**: provides information on the data uploaded
6. **Browse**: view the data (recommendation, do not use the Edit window)
7. **Do-file Editor**: use it to write/edit your Stata programs (.do files)

Menus: File

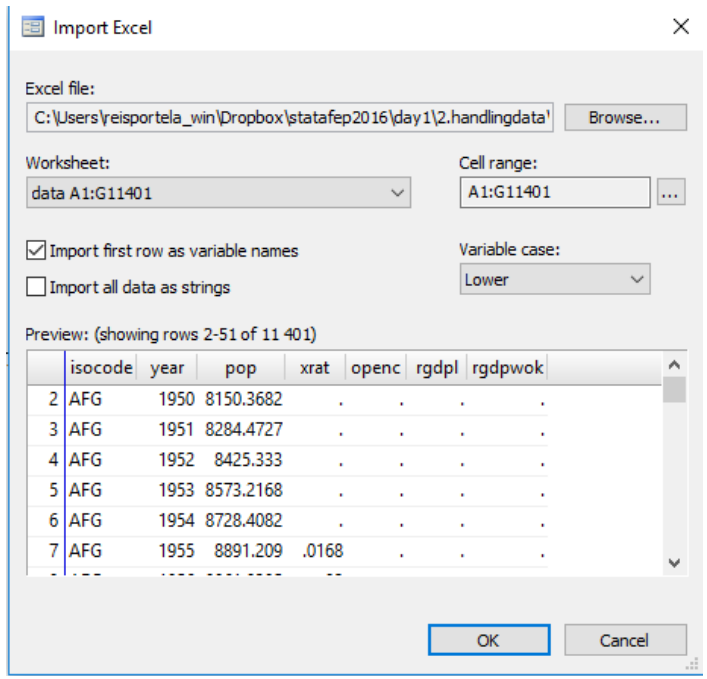


Menu 'File': Open..., Import, Example Datasets...

1. Change Working Directory: `cd "C:/tmp/stata/day1"`
2. Open a Stata data file from the internet: `use http://www.stata-press.com/data/r16/apple.dta`
3. Save in Stata native binary format: `save apple, replace`
4. Regular file open in Stata: `use apple`
5. Open a system example data file: `sysuse auto`

Menus: File, import an Excel file

Import data from Penn World Table (https://www.rug.nl/ggdc/productivity/pwt/): a sample file is available as 'pwt71_short.xlsx' under folder 'data'

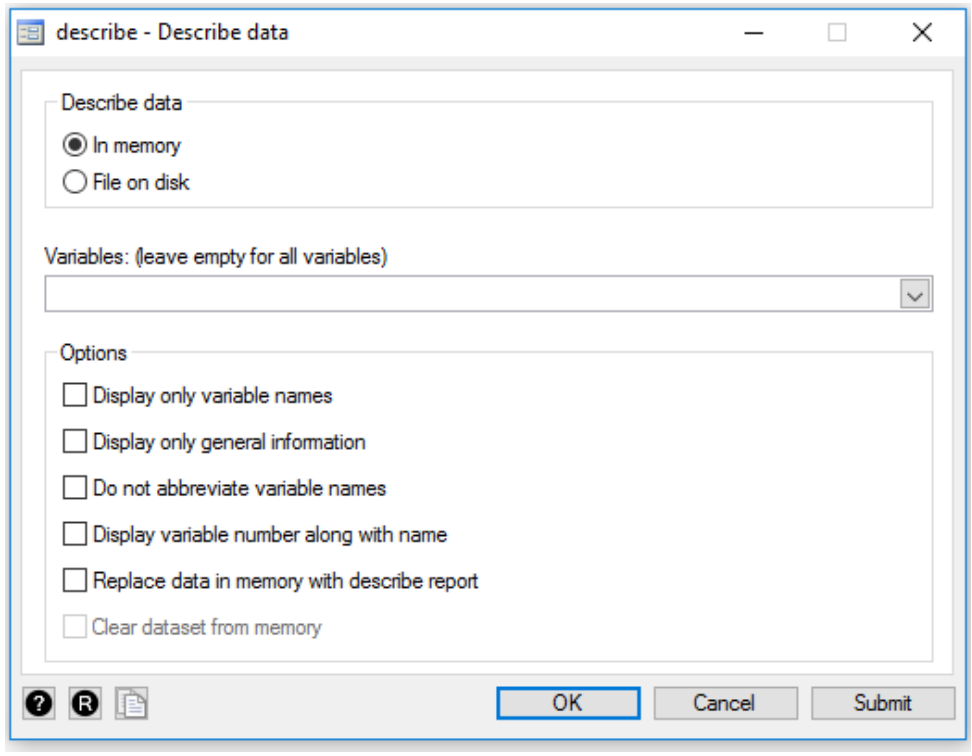


Menus: other menus

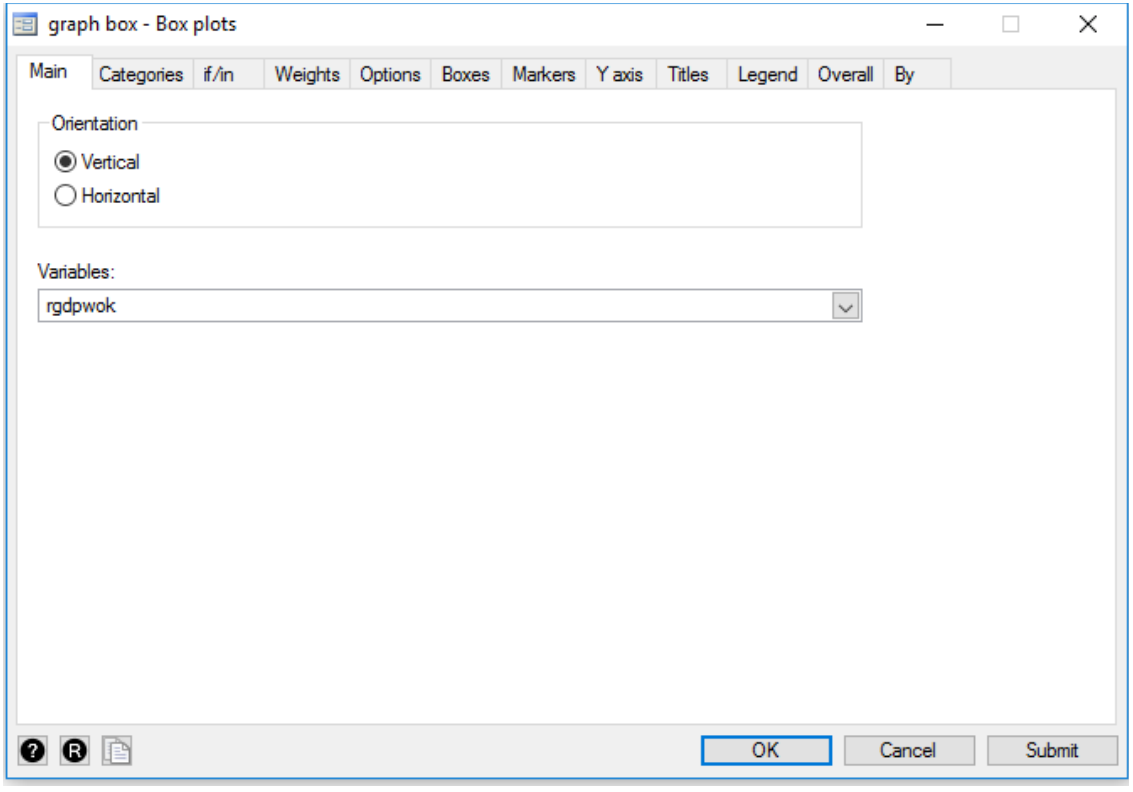
Explore the other menus: to explore them use, for example, the data 'auto.dta' by issuing the command 'sysuse auto'

- Edit : Preferences - proxy.uminho.pt:3128; colors
- Data : Describe data, Describe data in memory or in file; Summary statistics
- Graphics : Box plot; Smoothing and densities, Kernel density estimation
- Statistics : Summaries, tables, and tests; Summary and descriptive statistics; Frequency tables; One-way-table; Other tables; Compact table of summary statistics
- User : Users can share menus
- Window : Command, Results, etc.
- Help : PDF Documentation

Menus: Data, describe

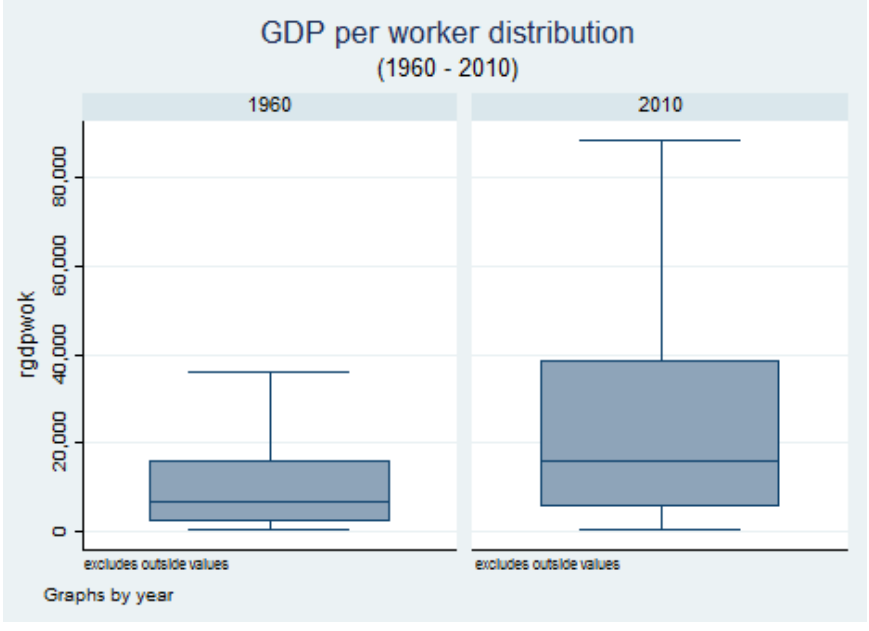


Menus: Graph, box plot

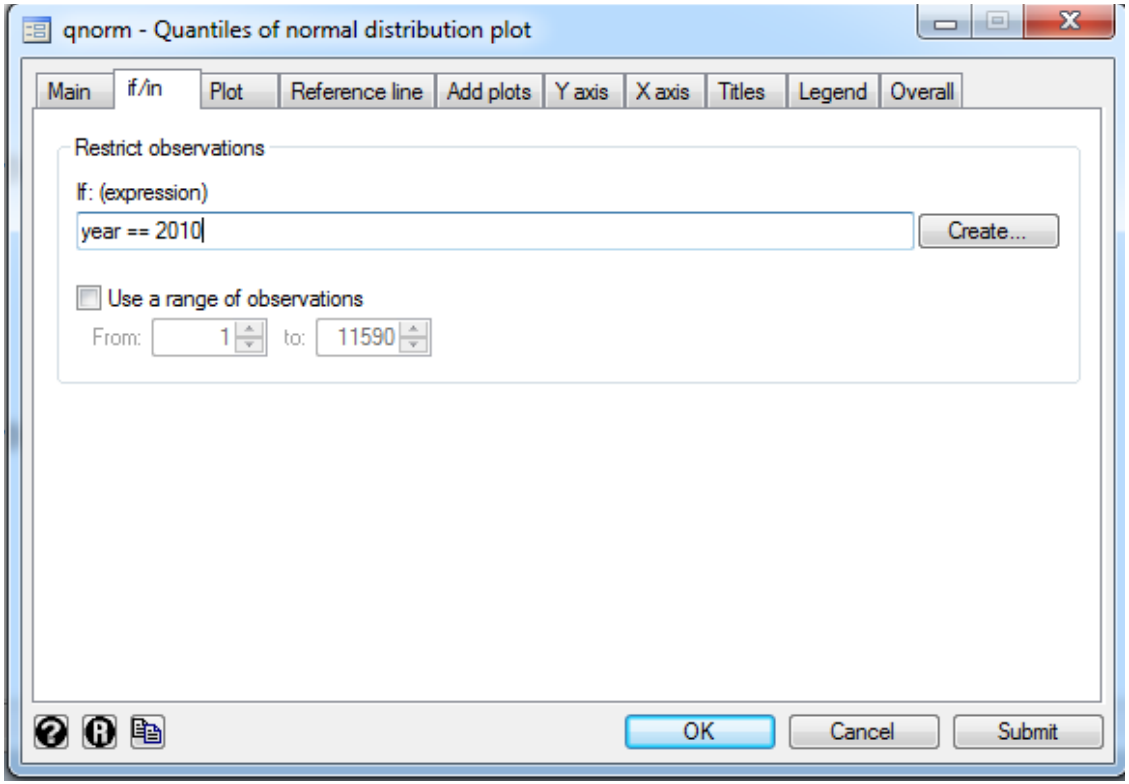


Menus: Graph, box plot (cont.)

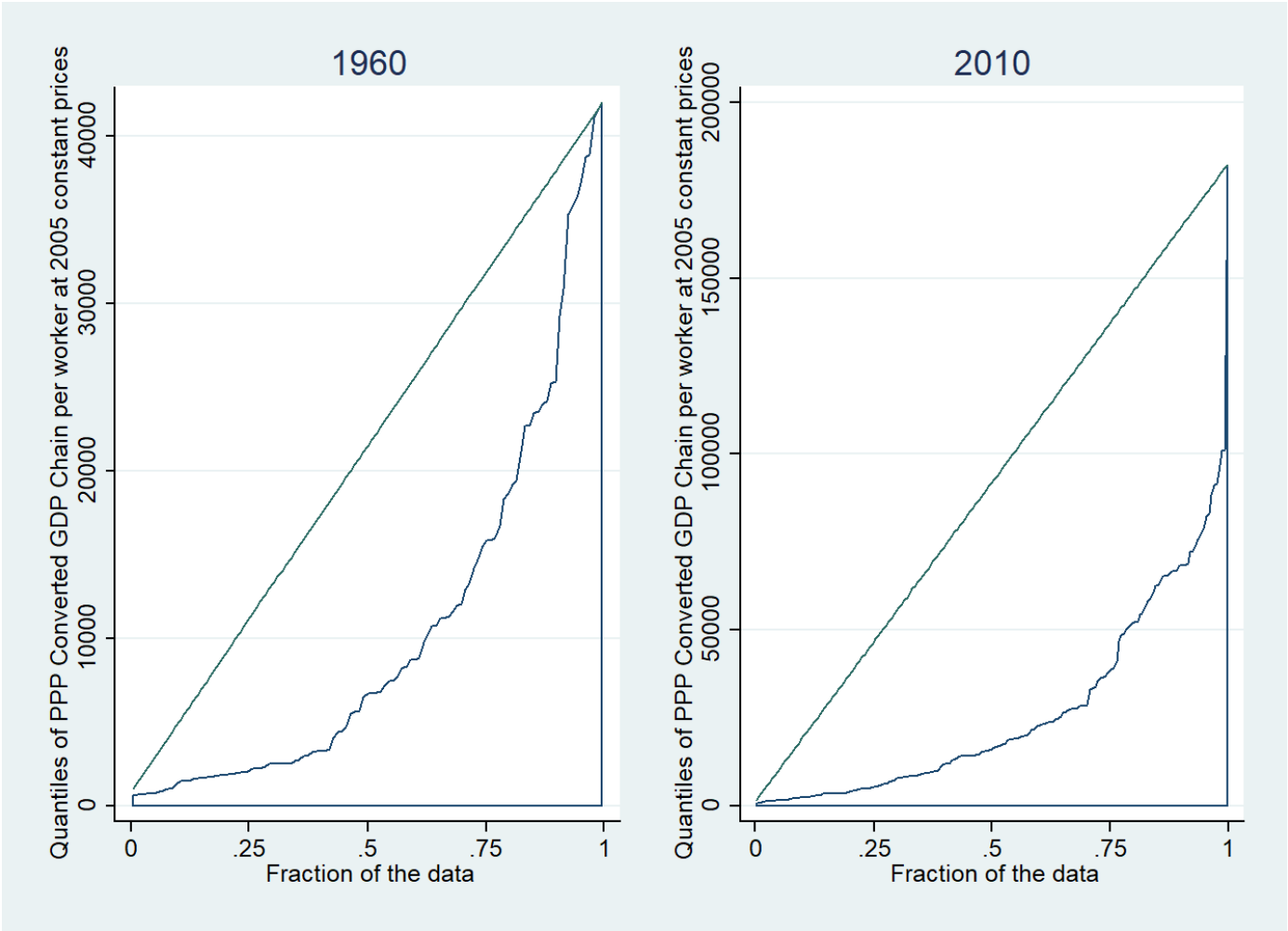
```
graph box rgdpwok, nooutsides by(year)
```



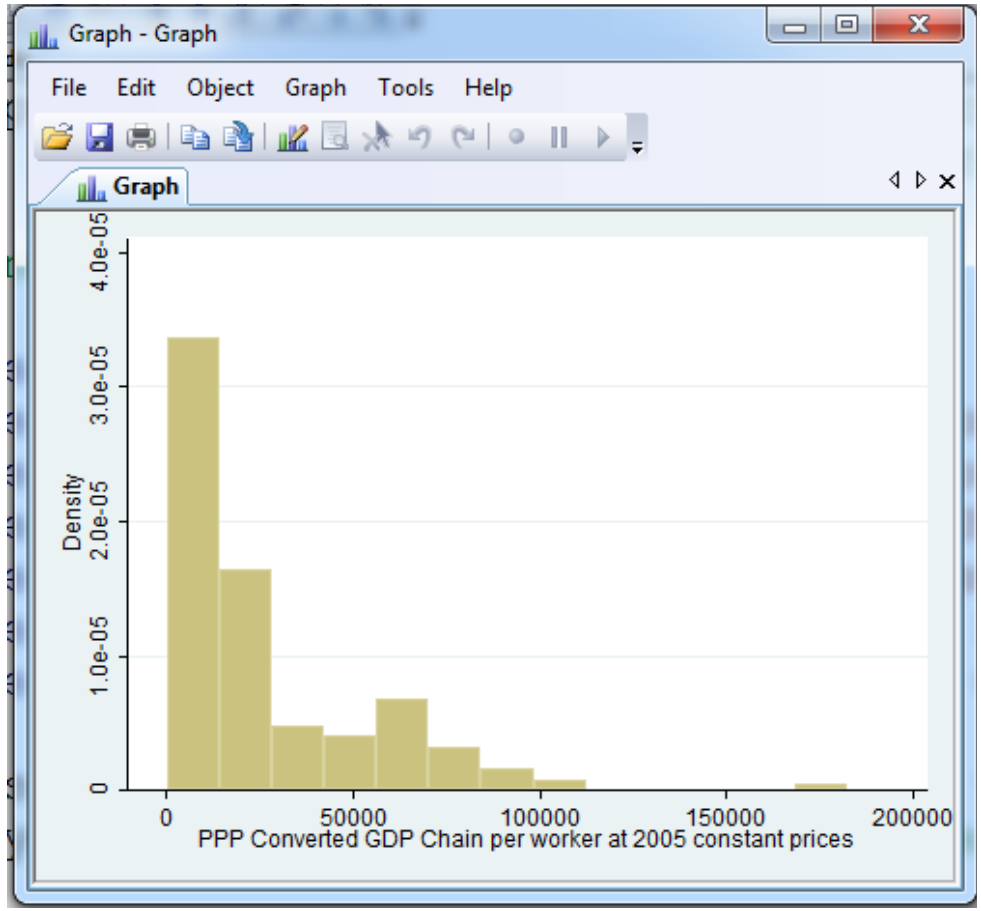
Menus: Graph, quantiles distribution plot



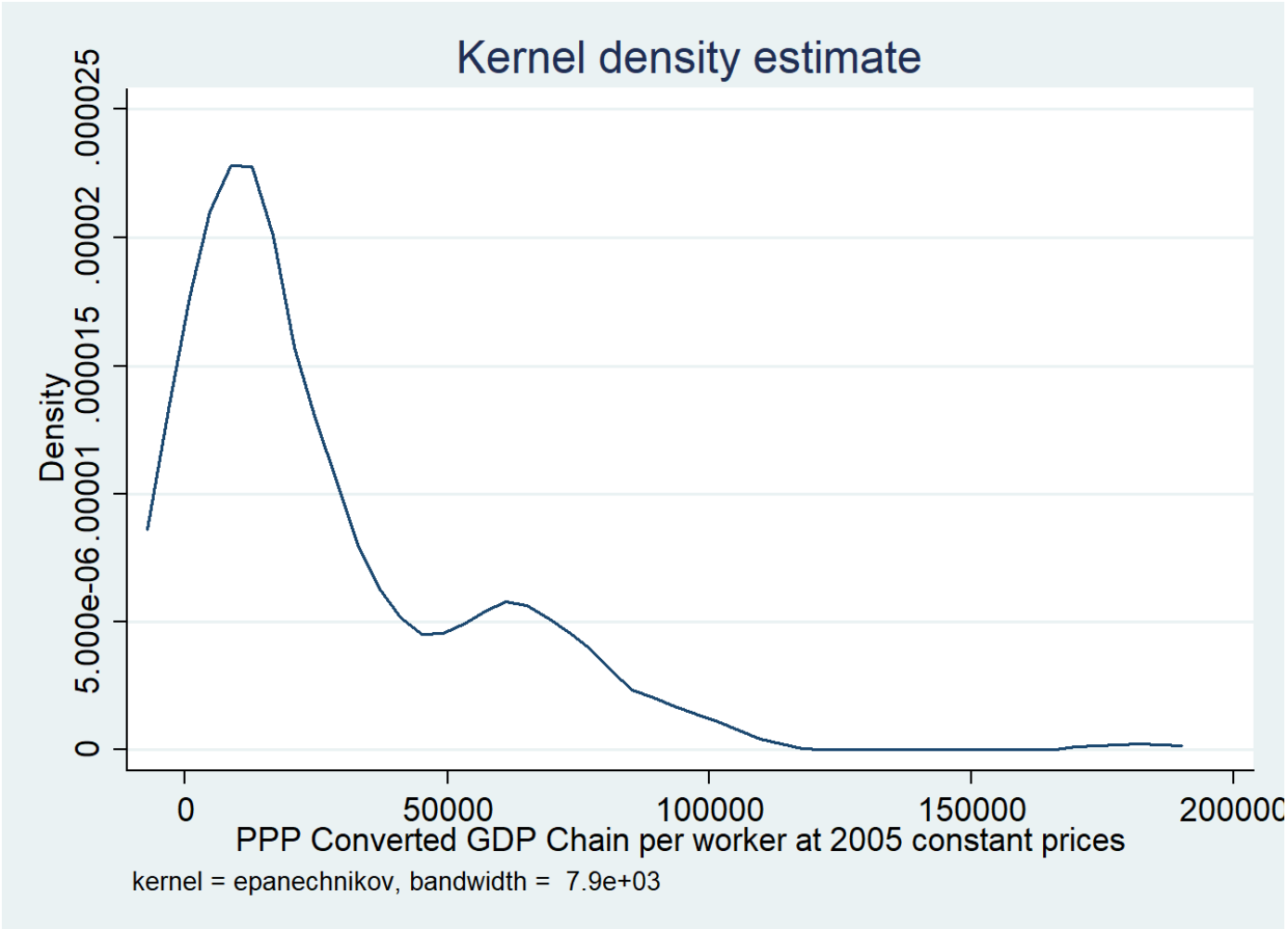
Menus: Graph, quantiles distribution plot (cont.)



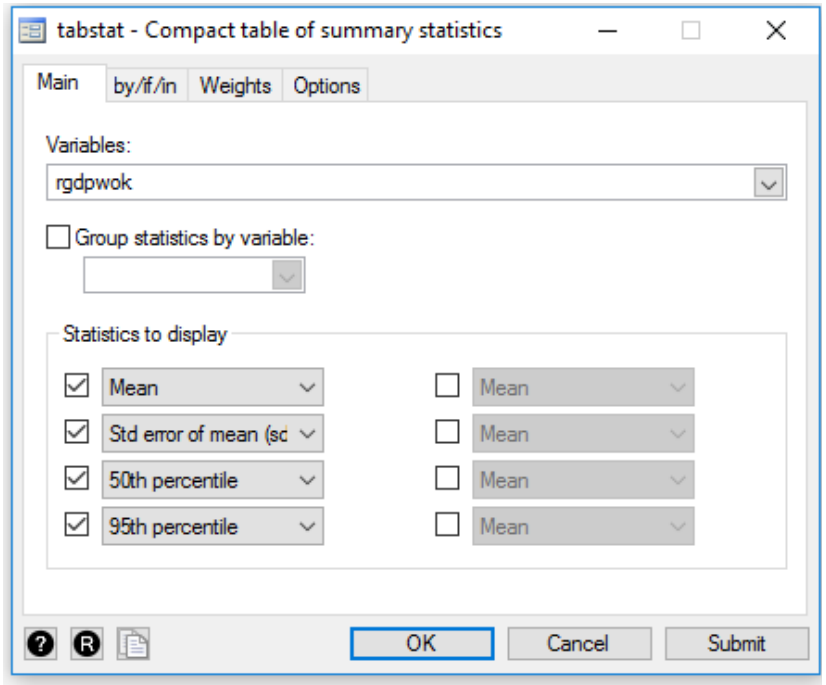
Menus: Graph, histogram



Menus: Graph, kernel density

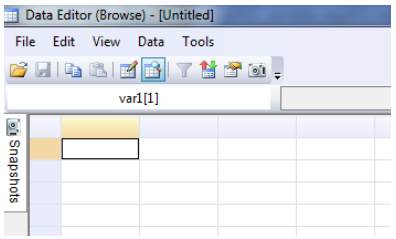


Menus: Statistics, tabulation of statistics

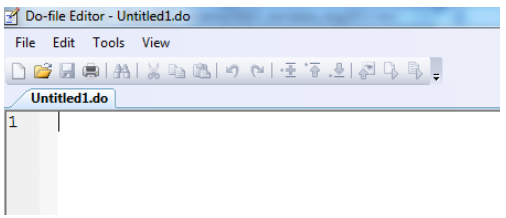


Further components

- `Browse the data`



- `Do-file Editor`



Typical Stata files

- “do” file
- “ado” file
- “dta” (data) file
- “log” file
- “smcl” file

Stata Command Syntax

[prefix :] command [varlist] [=exp] [if] [in] [weight] [using filename] [, options]

- Elements in square brackets are optional.
- "varlist" is a list of variable names,
- "command" is a Stata command,
- "exp" is an algebraic expression,
- "if" and "in" define a range of observations,
- "weight" is a weighting expression,
- "options" are options allowed by the particular command.
- The optional "prefix" invokes Stata prefix commands (more on this later).
- Most commands use the above syntax.
- Type `help language` at the command prompt for more information about Stata's language syntax

The grammar of Stata

- Explore the help feature: '`help summarize`' & pdf Stata's manual
- Descriptive statistics for a list of variables: `sum y16-y27`
- Abbreviation rules: `tab1 y~h`, `des x31?`, `br at*`
- The *in* qualifier: `list y~h y16-y27 in 1/10`
- `list y~h y16-y27 in -10/-1`
- The *if* qualifier: `tab z if x1 == 0`
- Operators: `help operators`
- `display ln(2) == log(2)`
- Functions: `help functions`
- The *by* prefix
- `sort x1`
- `by x1:  sum z`
- `bysort x1:  sum z`

The grammar of Stata (cont.)

LOOPS

The foreach loop: `h foreach`

```
foreach v of varlist x45-x53 {

    tab ‘v’ x1
}
```

The forvalues loop: `h forvalues`

```
forvalues n = 1/10 {
    gen r‘n’ = runiform()
}
```

Stata's resources

- Type `help language` and explore the syntax features
- You can look up the Stata Manuals by accessing the item "PDF Documentation" in the help menu. The manuals are pdf files that you are free to copy to any device that supports pdf files (the pdf files are usually in a folder named "docs" inside the Stata main folder).
- Stata is case-sensitive: the variables "price", "Price" and "PRICE" are all different
- Stata has several reserved names such as: "_all", "_b", "_cons"," _n", "_N", "_pi", "_pred", "_rc", "_skip", "_coef", "byte", "double", "float", "if", "in", "int", "long", "strL", "using", "with", and "str# "
- `help resources`
- Explore the commands: "`edit`", "`browse`", "`list`" , "`doedit`", "`clear`"} or "`cls`"

Organizing and handling data

- # 1. READING DATA FROM STATA FORMAT: use
- To find out the current "working directory"type on the command window `pwd`
- Move to the working directory '`cd C:/tmp/stata/day1/1.menus`'
- To read the file "auto.dta"from your current "working directory"do `use auto`
- Files that come with your Stata distribution (such as the auto.dta) can be read by typing `sysuse auto`
- To save a Stata file to the current "working directory"type `save mydata`, where "mydata"is the name of the file
- Stata will not let you write over an existing file with the same file name. In that case you need to add the "replace"option: `save mydata, replace`

Organizing and handling data (cont.)

- # 1. READING DATA FROM ASCII FORMAT: infile using fixed-format files (infile with a dictionary)
- Open the course directory "C:/tmp/stata"
- Move to folder 'data\handlingdata': `cd day1 1.menus`
- View the do file 'handlingdata.do': `doedit 1.1.1.menus.do`
- View with Stata doedit the files 'ascii_example.txt' and 'ascii_example.dct' under folder 'day1\2.handlingdata\ascii':
- `doedit ascii_example.txt`
- `doedit ascii_example.dct`

Organizing and handling data (cont.)

- # 2. READING DATA FROM CSV (comma-separated values) FORMAT (can be saved from different software)
- The data is available at Penn World Table (PWT): http://cid.econ.ucdavis.edu/pwt.html
- Read text data created by a spreadsheet; indicate that variables names are in the first row of the data file
- Label variables
- Type: `lookfor GDP`

Organizing and handling data (cont.)

- # 3. RETRIEVE A TABLE FROM A WEBSITE AND CLEAN THE OUTPUT
- [view http://youtu.be/NTFvh0Pk_OA]
- Go to the website using excel
- Save the file in Excel format
- In Stata 16 import the data from xlsx using Stata's menu File > Import ...
- How to do this task using `edit` and Copy/Paste?
- Solution using the command `insheet`
- Possible problem
- Filter for some characters in your variables
- Explore further functions, in particular string functions

Organizing and handling data (cont.)

- # 4. COMBINE INFORMATION FROM DIFFERENT SOURCES: CROSS PWT WITH BARRO & LEE DATA
- Source: http://www.cid.harvard.edu/ciddata/ciddata.html
- Exercise
- build an excel file with data on education by country and year
- transport the data to Stata
- combine the information with PWT used above
- # 5. RESHAPING THE DATA
- Source: http://www.cid.harvard.edu/ciddata/ciddata.html
- Download the data in Excel format
- Import it to Stata and save the file 'cap_inv.dta'
- `import excel cap_inv.xls, sheet("CAP_INV")`

Organizing and handling data (cont.)

- `merge` - merges a Stata data set in memory (the master dataset) to a data set in disk (the using data set) using one or more key variables: `merge 1:1 var1 using data2`
- `append` - appends a Stata data set on disk (say data2) to a Stata data set in memory (say data1): `append using data2`
- `contract`
- `reshape`
- `fillin`
- `expand` - duplicate observations
- `collapse`
- Explore `xpose`, `cross`, `joinby` and `stack`

Key Stata commands to manage your data

- The easiest way to create a variable is with the `generate` command: `generate myvar=1`
- To change the name of a variable do: `rename myvar newvar`
- To delete a variable do: `drop myvar1`
- Or you can specify the variables to keep in the data set: `keep myvar2 myvar3`
- The '`drop`' and '`keep`' commands may also be used to drop observations
- For example, `drop if myvar2==3`, will delete all observations for which myvar2 is equal to 3

Key Stata commands to manage your data (cont.)

- Stata uses two equal signs for whenever it wants to check for an equality. On the other hand, one equal sign is used when we want to assign values (eg. `generate myvar=1`)
- The command "`keep`" can be used to retain a subset of the observations: `keep if x==3` will do the opposite of "`drop if x==3`"
- You can modify selected values of a variables using the replace command: `replace var1=0 if var2==3`
- Or you can recode the values of a variable, `recode var1 2/3=2 4=3`, and this changes the values 2 and 3 to 2 and the value 4 to 3.
- The commands '`preserve`' and '`restore`' can be used to implement temporary changes to your data (example will be shown later on)

Expressions

- There are logical, algebraic and string expressions in Stata
- Expressions involving variables are evaluated for each observation
- Logical expressions evaluate to 1 (true) or 0 (false) and use the operators < and $\leq$ for "less than" and "less than or equal to" to respectively and similarly > and >= are used for "greater than" and "greater than or equal to".
- The symbols == and $!=$ stand for "equal to" and "not equal to", and the characters !, & and | represent "not", "and" and "or" respectively. For example, the expression

if $(y != 2 \& z > x) | x == 1$
means "if y is not equal to two and z is greater than x or if x equals one"

Expressions (cont.)

- Algebraic expressions use the usual operators +, -, *, / and ^ for addition, subtraction, multiplication, division, and powers, respectively.
- String expressions mainly use special string functions such as substr(myvar,3,4) to extract a substring of myvar starting at position 3 for a length of 4 characters.
- The logical operators == and ~= are also allowed with string variables and the operator + concatenates two strings. For example, the combined logical and string expression

$$(\text{"}moon\text{"} + substr(\text{"}sunlight\text{"}, 4, 5)) == \text{"}moonlight\text{"}$$

returns the value 1 for "true"

Functions

- There are several functions that can be used in expressions.
- These are mathematical, statistical, string, programming, date and time and matrix functions.
- Some examples are:
- abs(x) computes the absolute value of x.
- For example, The command `gen amyvar=abs(myvar)` computes the variable "amyvar" which is the absolute value of "myvar"
- exp(x) provides the exponentiation of x.
- ln(x) computes the natural logarithm of x.
- For a complete list of functions just type `help functions`

Formatting

- Formatting controls how the variable is displayed. For example:

$$format\ varname\ \%7.2g$$

- The characters immediately after "% " specify the format options. Some examples are:
- % #.#g - general numeric format. Eg. %9.0g
- % #.#f - fixed numeric format. Eg. %9.2f
- %d - default numeric elapsed date format. Eg. %d
- To see a complete list of formats just type: `help format`
- The easiest way to manage labels, notes and formats is by using the "Variables Manager" window (type "`varmanage`") or select (DATA->Variables Manager) from the menu

Sorting

- The sort command sorts your data in ascending order
- `sort var1`
- or
- `sort var1 var2`
- note that missing values are treated as very large numbers
- To sort variables in descending order you must use gsort
- `gsort +var1 -var2`
- The "+" before the name of the variable means ascending while the "–" is for descending order. If you omit the sign it will assume "+"

Further commands

- The describe command gives a general overview of the data set
- The compress command automatically converts the variables to their most adequate storage type
- The display command is used to show strings and values. It is also a powerful calculator. Strings must be enclosed in quotes. Some examples of display are:
- `display 2`
- `display` “Hello”
- `display 2+2`
- `display 2+log(5)`
- `display "20 times 10 is " 20*10`

Further commands: indices

- Each observation has associated with it an index starting from 1 to the last observation (the row number in the Data Editor)
- For example, the value of the third observation of variable `x` may be referred to as `x[3]`: `display x[3]`, which will display the 3rd observation of x.
- The (hidden) system variables `_n` takes on the value of the running index and `_N` is equal to the total number of observations. For example, if we wish
- to replace `x[3]` by 2, we can do this using the syntax: `replace x=2 if _n==3`
- We can refer to a range of observations using either if with a logical expression involving `_n`

Further commands: indices

- An easier alternative makes use of the in range, where range is a range of indices specified using the syntax f/l (for "first to last") where f and/or l may be replaced by numerical values if required. Here 5/12 means "fifth to twelfth" and f/10 means "first to tenth" etc. Negative numbers are used to count from the end, for example: `list x in -10/l`, lists the last 10 observations
- An easy way to construct lags and leads is by using `_n`, `gen lagx = x[_n-1]`, creates a variable which is the lag of `x`

Further commands: prefix commands

- Most commands allow a prefix command. A prefix command is separated by a ":" sign. Most popular are `by` and `quietly`
- The `by` command executes the command on subsets of data defined by the `by` variables (data must be sorted on by variables – if not type `bys` instead of `by`, `bys firm: gen nobs=_N`, or `bys firm: summarize sales`
- quietly suppresses all terminal output for the duration of command: `quietly: replace` $x = 2$ `if` $z > 0$
- For a list of prefix commands type `help prefix`

Further commands: egen function

- More complex functions are available through the egen command (extensions to generate).
- These include functions designed to deal with groups of observations or across groups of variables
- For a list of functions type `help egen`

Further commands: dates

- Stata stores all time variables as numeric
- Time variables may be year, months, days, milliseconds. The numeric value is the number of periods (years, days, etc) since 1960
- Time variables are then formatted to display the dates in a friendly manner
- Stata has functions to convert strings to Stata time variables, to change from one type of time variable to another. See `help datetime functions`
- A time variable defined in days corresponds to the number of days since 1/1/1960. For example, the number 14976 represents the 1st of January of 2001. If the variable is formatted as "%d" it will be displayed as "01jan2001"
- For more information about dates type `help datetime`
- Explore type `varmanage`

Further commands: categorical variables

- In Stata it is generally advisable to represent categorical (factor) variables as numeric variables with value labels
- A categorical string variable can be converted to a numeric variable using the command '`encode`' which replaces each unique string by an integer and uses that string as the label for the corresponding integer value: `encode make, generate(maken)`
- The command '`decode`' converts the labeled numeric variable back to a string variable: `decode foreign, generate(fors)`
- If you create a categorical variable in Stata you never need to create additional variables based on those categories. Stata is very efficient at dealing with categorical variables. See `help fvvarlist`

Further commands: missing values

- Missing values are represented by ".".
- Missing values assume the largest possible number. Be careful when using logical conditions in variables with missing values: `keep if` $x > 0$
- will drop the negative values of x but retain the missing values. If you want to also drop missing values do: `keep if` $x > 0$ & x<., or, `keep if` $x > 0$ `& !missing(x)`
- You can code different types of missing values using Stata's 27 numeric missing values (eg, ".a", ".b", ".c", etc...)

Data analysis

- Move to folder 'data\dataanalysis': `cd day1 3.dataanalysis`
- `doedit 1.3.1.statistics.do`
- To obtain a general description of the data in memory use: `describe`
- You can obtain a more detailed analysis with: `codebook` or `codebook, compact`
- Or you can use the inspect command to learn about a specific variable: `inspect rep78`
- You can use the command lookfor to search for a string on a variable name or label: `lookfor finance`
- To obtain basic descriptive statistics use: `summarize`

Data analysis (cont.)

- A useful command is '`count`' which simply counts the number of observations. Combined with '`if`' it is a simple way to describe your data. For example: `count if rep78==1& foreign==0`
- Categorical variables can be tabulated using the '`tabulate`' command: `tabulate rep78`, or even '`tabulate rep78 foreign`' to produce a two-way table of frequencies
- If you want to produce a series of one-way tables for a list of variables you can do: `tab1 rep78 foreign trunk`

Data analysis (cont.)

- To get a table of basic summary statistics for each level of a categorical variable do: `tabulate foreign, summarize (mpg)`
- Explore the following options: `missing`, `nofreq`, `nolabel`, `plot` and `sort`
- You can also use `tabstat` to produce tables of summary statistics. For example,

$$tabstat\ mpg,\ by(foreign)\ statistics(meanmax)$$

calculates the mean and maximum of `mpg` for each class of `foreign`

Data analysis: graph syntax

- `help graph`
- `graph box education if year == 2000`
- `help scheme`
- `help graph other`
- Graph Editor: **File** > **Start Graph Editor**
- Explore 'Start recording': it allows you to replay the same changes in other graphs