An Introduction to Stata for Econometric Analysis

- Lecture 1: Basics and Descriptive Statistics -

Filippo Pavanello

University of Bologna

a.y 2022/2023

Outline

Basic Information

- The aim of this course is to provide you with a basic training to Stata software package
- The course is based on the Stata Manual and will be focused on data management and will provide you with all necessary information on building datasets
- Slides and other teaching material will be share with you after each lecture
- Thanks to **Nektaria Glynia** for sharing their slides on previous Stata introductory courses
- I am always available via email at filippo.pavanello2@unibo.it for any clarification or hint
- **Office hours:** by appointment on Microsoft Teams or in PhD Room (Piazza Scaravilli, First floor)

Outline

Manuals and Books

- A.C. Cameron & P.K. Trivedi "Microeconometrics using Stata" Revised Edition, 2010 StataCorp LP
- Stata has an excellent website with many step-by-step command description (http://www.stata.com/)
- Stata List is an independent list server maintained by Marcello Pagano at the Harvard School of Public Health, where you can post questions and receive prompt and knowledgeable answers from other users (http://www.stata.com/support/statalist/)
- UCLA maintains an excellent Stata portal, with many useful links, including a list of resources to help you learn and stay up-to-date with Stata (http://www.ats.ucla.edu/stat/stata/)
- Princeton University Data and Statistical Services (DSS) also host tutorials and other resources to learn Stata (`https://dss.princeton.edu/training/`)

Outline

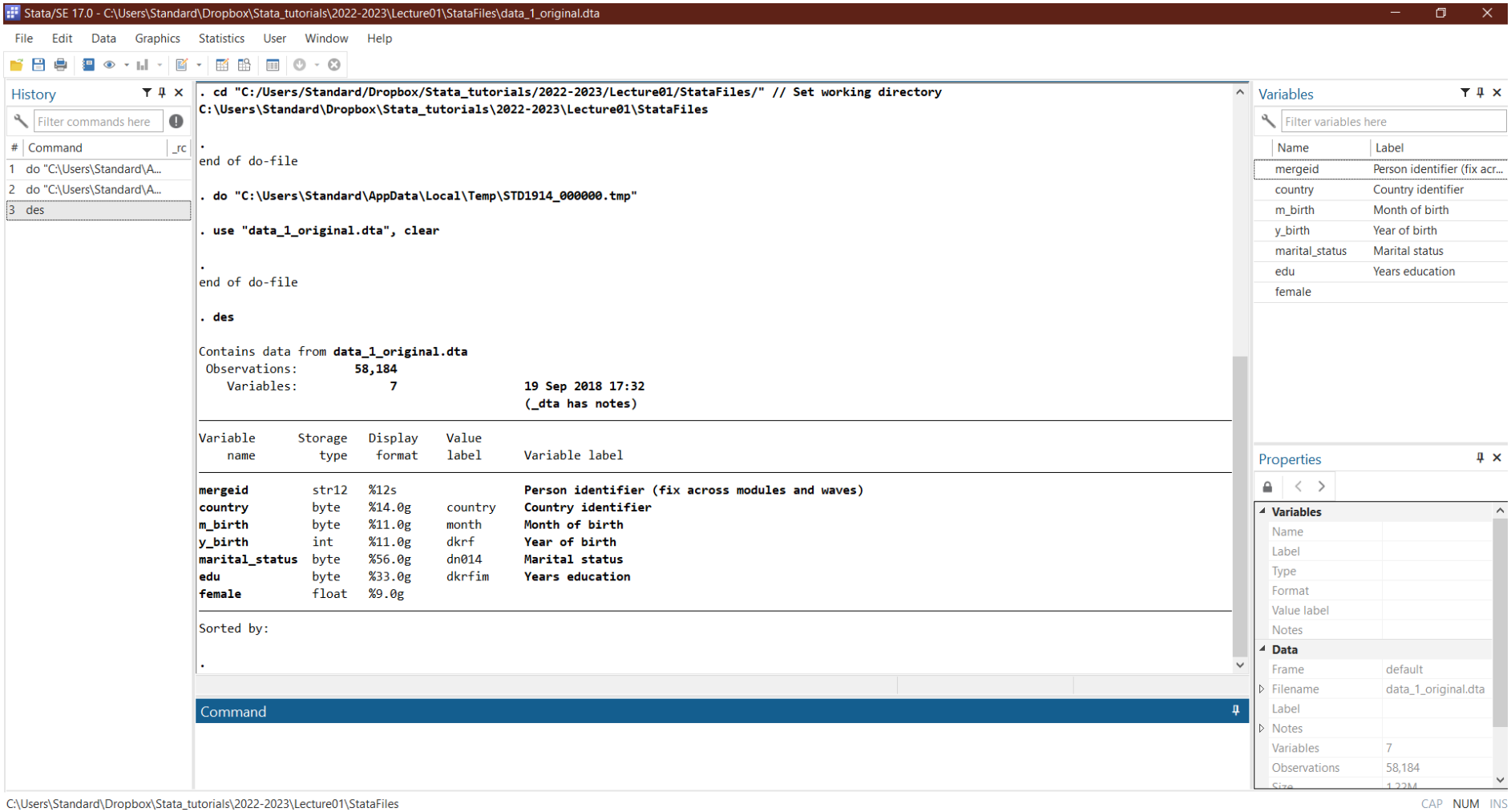
Structure of the Course

This course is structured in four lectures and will cover:

1. Introduction to the program and data management features
2. Tools for graphical analysis and descriptive statistics
3. OLS estimation and postestimation tools
4. Introduction to programming with macros

Introduction

The Stata Interface



Introduction

The Environments

Stata handles information conveyed on 4 linked environments:

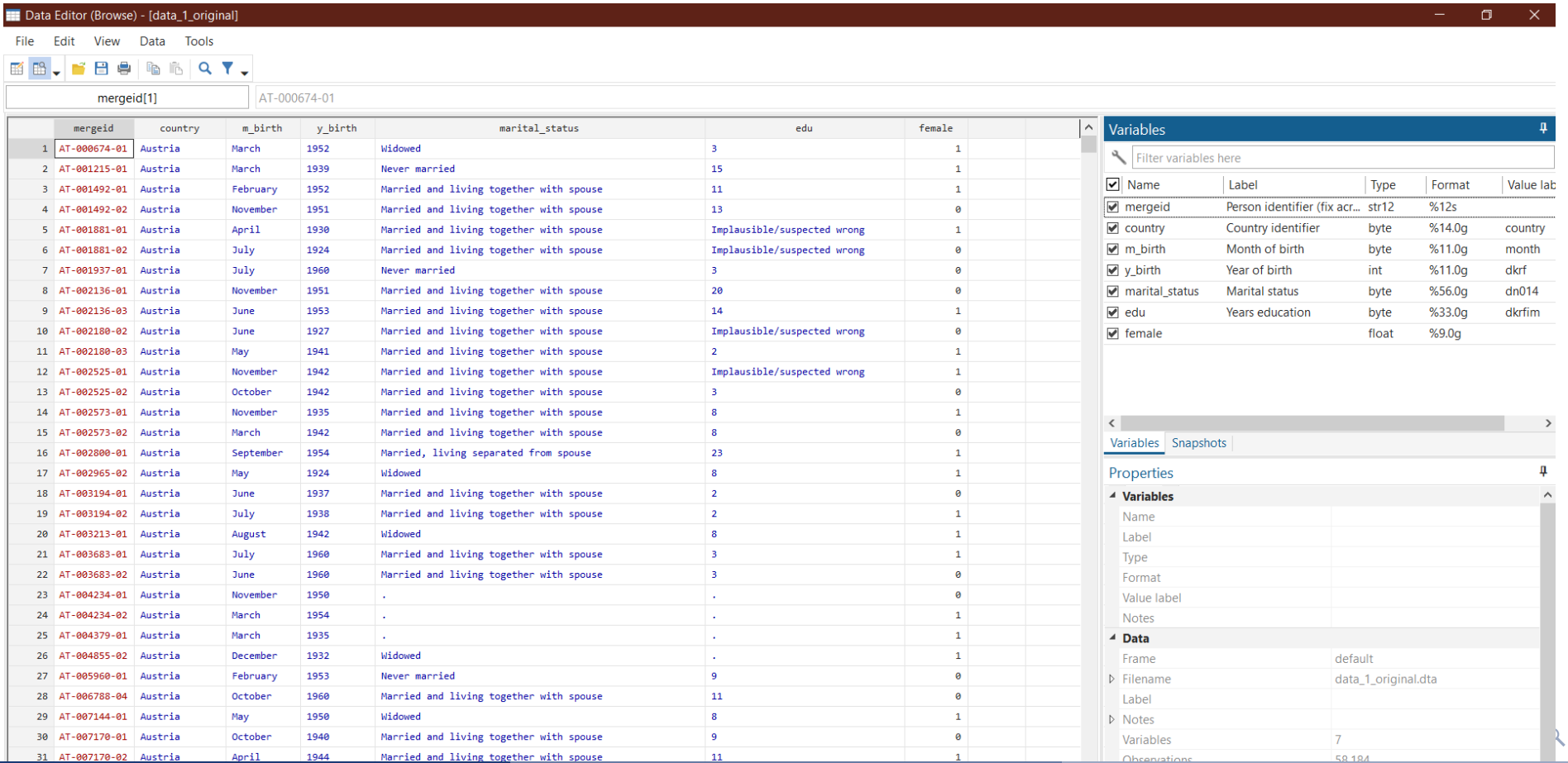
1. The **Dataset** is a collection of data relative to a sample of the population we want to study, i.e. the data we download
2. The **Macros** are temporary variables that contains a limited amount of strings/numbers and are useful to program a do-file
3. **Stata matrix programming language** performs operations with matrices defined as entities
4. **Mata matrix programming language** is an updated and faster version of Stata matrix programming available from Version 10

Introduction

The Dataset

The Dataset is organized in rows and columns:

- Each **column** represents a **variable**, i.e. testscore, str, district...
- Each **row** corresponds to a **sample unit** of the phenomenon that we want to study, i.e. schools, individuals, firms, households, countries...



	mergeid	country	m_birth	y_birth	marital_status	edu	female
1	AT-000674-01	Austria	March	1952	Widowed	3	1
2	AT-001215-01	Austria	March	1939	Never married	15	1
3	AT-001492-01	Austria	February	1952	Married and living together with spouse	11	1
4	AT-001492-02	Austria	November	1951	Married and living together with spouse	13	0
5	AT-001881-01	Austria	April	1930	Married and living together with spouse	Implausible/suspected wrong	1
6	AT-001881-02	Austria	July	1924	Married and living together with spouse	Implausible/suspected wrong	0
7	AT-001937-01	Austria	July	1960	Never married	3	0
8	AT-002136-01	Austria	November	1951	Married and living together with spouse	20	0
9	AT-002136-03	Austria	June	1953	Married and living together with spouse	14	1
10	AT-002180-02	Austria	June	1927	Married and living together with spouse	Implausible/suspected wrong	0
11	AT-002180-03	Austria	May	1941	Married and living together with spouse	2	1
12	AT-002525-01	Austria	November	1942	Married and living together with spouse	Implausible/suspected wrong	1
13	AT-002525-02	Austria	October	1942	Married and living together with spouse	3	0
14	AT-002573-01	Austria	November	1935	Married and living together with spouse	8	1
15	AT-002573-02	Austria	March	1942	Married and living together with spouse	8	0
16	AT-002800-01	Austria	September	1954	Married, living separated from spouse	23	1
17	AT-002965-02	Austria	May	1924	Widowed	8	1
18	AT-003194-01	Austria	June	1937	Married and living together with spouse	2	0
19	AT-003194-02	Austria	July	1938	Married and living together with spouse	2	1
20	AT-003213-01	Austria	August	1942	Widowed	8	1
21	AT-003683-01	Austria	July	1960	Married and living together with spouse	3	1
22	AT-003683-02	Austria	June	1960	Married and living together with spouse	3	0
23	AT-004234-01	Austria	November	1950	.	.	0
24	AT-004234-02	Austria	March	1954	.	.	1
25	AT-004379-01	Austria	March	1935	.	.	1
26	AT-004855-02	Austria	December	1932	Widowed	.	1
27	AT-005960-01	Austria	February	1953	Never married	9	0
28	AT-006788-04	Austria	October	1960	Married and living together with spouse	11	0
29	AT-007144-01	Austria	May	1950	Widowed	8	1
30	AT-007170-01	Austria	October	1940	Married and living together with spouse	9	0
31	AT-007170-02	Austria	April	1944	Married and living together with spouse	11	1

Introduction

Macros

- Macros are temporary variables containing **limited** qualitative or quantitative information that you can use in subsequent commands
- Macros are used **to generate loops** to save time and make your work neater
- Macros come in two types:
  1. **Global macros**, once defined, are available anywhere in Stata;
  2. **Local macros** exist solely within the do-file in which they are defined. When a do-file ends to run, its local macros are permanently deleted;
- Commands are different for global or local macros

File Types

Dta files

When working in Stata, you will manage 4 types of files:

- **.dta**: they contain data in Stata format and they cannot be opened with other softwares
- **.do**: they contain a list of Stata commands to replicate outputs. They can be opened and edited with Notepad
- **.log**: they contain Stata outputs and they can be opened with Notepad
- **.gph**: they contain Stata graphs that can be converted in .png, .tif, .eps

File Types

Types of variables

Variables in a dataset file (.dta) can be:

`float`	floating point numbers with 8 decimal digits in $\pm 1.70141173319 * 1038$ (**default**)
`byte`	integers in [-127; 100]
`int`	integers in [-32767; 32740]
`long`	integers in [-2147483647; 2147483620]
`double`	floating point numbers with 17 decimal digits in $\pm 8.9884656743 * 10307$
`str#`	Strings that contain text. The maximum number of characters is 244
`date`	Stata stores dates and times numerically as durations from some sentinel date (1 January 1960) in specified units. Stata internal form (SIF) can be converted to human readable form (HRF) using some specific commands The date 19feb2013 is stored in Stata as 19408

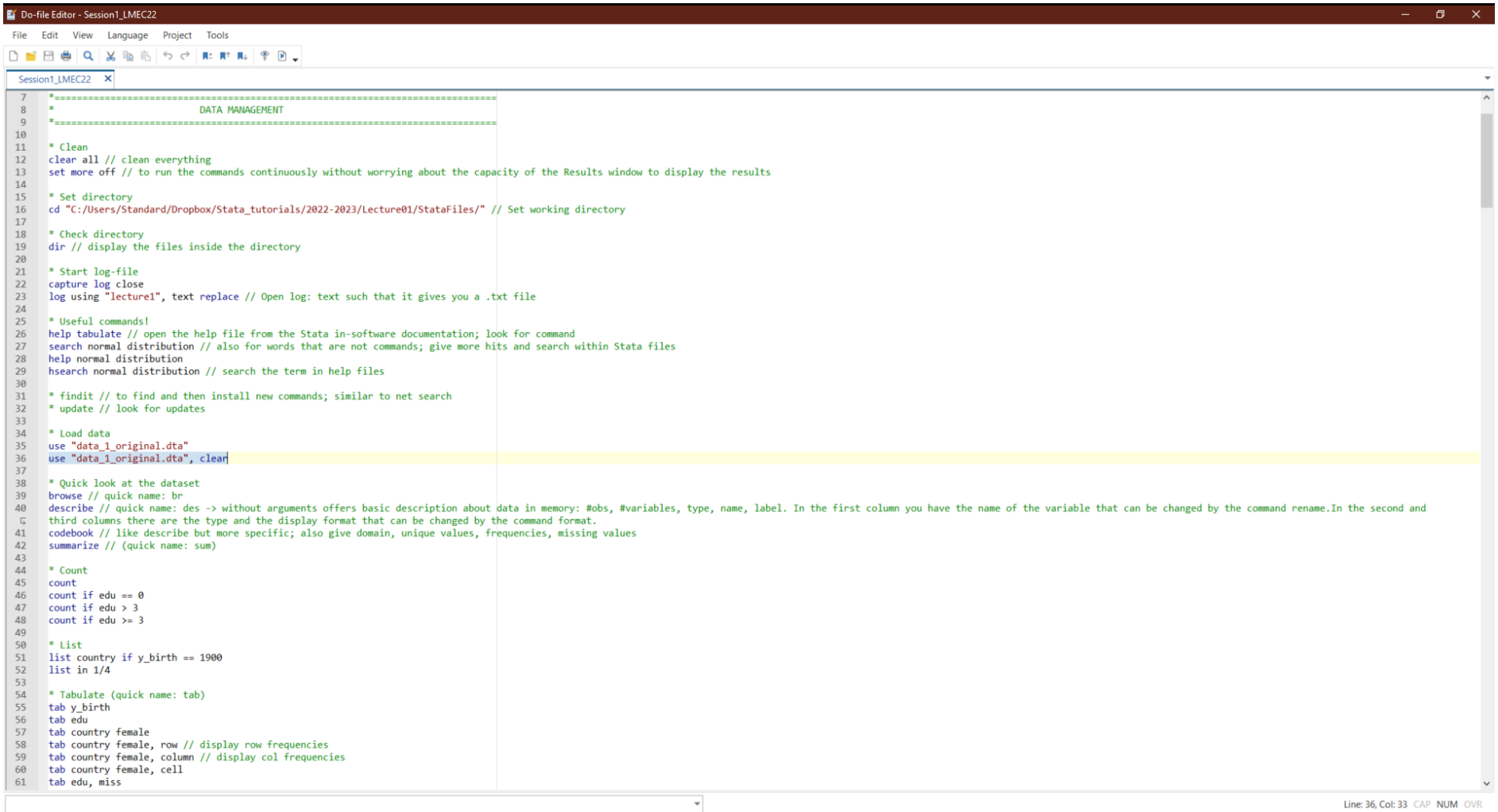
File Types

Do Files

- A do file is just a set of Stata commands typed in a plain text file
- You can use Stata's own built-in do-file Editor, which can run your program directly from the editor
- You can also select just a few commands and run them by selecting few lines of each command
- In theory, you could use any word processor to write your do file, but you would have to remember to save the file in plain text format and you will not be able to run directly
- **Always use a do file!**

File Types

Do Files



File Types

Log Files

- A log file saves permanently what appears in your output window in a file
- By default the log is written using SMCL, Stata Markup and Control Language (pronounced "smicle"), but can only be viewed using Stata's Viewer
- There is a **text option** to create logs in plain text format, which can be viewed in an editor such as Notepad or Word
- Log files will be the first output of your regressions
- **Always log your results!**

Example

`log using filename, text replace` – To open the log
`log close` – To close the log

File Types

Log Files

Do-file Editor - lecture1

File Edit View Language Project Tools

lecture1 Session1_LMEC22

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
      name:  <unnamed>
       log:  C:\Users\Standard\Dropbox\Stata_tutorials\2022-2023\Lecture01\StataFiles\lecture1.log
  log type:  text
 opened on:  27 Sep 2022, 18:19:47

.
end of do-file

. do "C:\Users\Standard\AppData\Local\Temp\STD1914_000000.tmp"

. help tabulate // open the help file from the Stata in-software documentation; look for command

. search normal distribution // also for words that are not commands; give more hits and search within Stata files

. help normal distribution

. hsearch normal distribution // search the term in help files
(building index ....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%)

.
end of do-file

. do "C:\Users\Standard\AppData\Local\Temp\STD1914_000000.tmp"

. use "data_1_original.dta"

. use "data_1_original.dta", clear

.
end of do-file

. do "C:\Users\Standard\AppData\Local\Temp\STD1914_000000.tmp"

. browse // quick name: br

. describe // quick name: des -> without arguments offers basic description about data in memory: #obs, #variables, type, name, label. In the firs
> t column you have the name of the variable that can be changed by the command rename.In the second and third columns there are the type and the
> display format that can be changed by the command format.

Contains data from data_1_original.dta
 Observations:        58,184
    Variables:             7                  19 Sep 2018 17:32
                                              ( dta has notes)
```

File Types

Ado Files

- Ado files are automatically loaded do-files
- '.ado' files contain the code for Stata commands
- Stata maintains a path that is a list of the directories over which it will search to find a command. In order to execute a command, Stata first looks to see if the command name is an internal command. If so, it executes it
- If not, Stata searches the path for the command that was issued by appending ".ado" to the end of the command and looking for the file named command.ado in the directories
- **You can add new commands for your research purpose**

File Types

Directories

- If you type `sysdir` or `adopath` you will know where Ado files are stored
- You know where to store your additional commands and it is a very useful check when some commands/new commands are not recognized by Stata

Example

`sysdir` – Query and set system directories

Commands Syntax

Commands

Stata commands follow a common syntax

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

see	language element	description
help prefix	prefix:	prefix command
help command	command	Stata command
help varlist	varlist	variable list
help exp	=exp	expression
help if	if	if exp qualifier
help in	in	in range qualifier
help weight	weight	weight
help using	using filename	using filename modifier
help options	options	options

Square brackets distinguish optional from required options. Items presented like this should be typed **exactly** as they appear in the diagram

Commands

Syntax details

`prefix:`	it precedes the command. Example: `quietly` run command and suppress the output or `by` run command on subsets of data. Example: `by group: egen mean = mean(age)`
`command`	is the only required element is the command itself, which is usually (but not always) an action verb. Stata commands are **case-sensitive**. Underlining is used to indicate abbreviations where possible. Example: `regress y x` or `reg y x`
`varlist`	indicates the name of one or more variables. Variable names are **case sensitive** and name can be abbreviated to the minimum number of letters that makes it unique in a dataset, but be careful when you create new variables. You can also select variables with similar names. Example: `v*` or v1-v15
`=exp`	indicates arithmetic expressions, functions, and parentheses. Example: `loggdp = log(gdp)` or `educsq =` $educ^2$
`weight`	Some commands allow the use of weights, such as frequency weights indicating the number of duplicated observations. Example: `fweights=pop`
`using file`	it uses a file stored in your computer, network or on the internet. Example: `using "census.dta"`

Commands

Syntax details

`options`	Options are specified by placing a comma at the end of the command and then listing the options one after another with intervening spaces: `import excel caschool.xls, sheet("caschool") firstrow clear` In this example: `import excel` is the command, `caschool.xls` is the file name `firstrow` and `clear` are the options

Commands

Operators

Algebraic and string expressions are specified in a natural way using the standard rules of hierarchy.

Arithmetic		Logical		Relational (numeric and string)	
+	addition	&	and	>	greater than
-	subtraction	|	or	<	less than
*	multiplication	!	not	>=	> or equal
/	division	~	not	<=	< or equal
^	power			==	equal
-	negation			!=	not equal
+	string concatenation			~=	not equal

A double equal sign (==) is used for equality testing, i.e. use it after specifying `if`

Commands everyone should know

Commands everyone should know

Help

Stata has very exhaustive help and manual. **Learn to use it!**

`help`	enables you to obtain help on a command (or function), which displays the help on a separate window called the Viewer. Every help page is linked directly to the related section on the manual
`search`	searches a keyword database and the Internet
`findit`	searches a word in the online help, the FAQs at the Stata website, the Stata Journal, and all Stata-related Internet sources including user-written additions
`hsearch`	searches help files
`net search`	searches the Internet for installable packages and you can use it to directly install additional ado-files
`update`	searches the Internet and updates Stata

Commands everyone should know

Working directory, loading, saving the data

`clear`	clears the memory: without cleared memory Stata does not load any new dataset
`set`	sets system parameters. Example: `set more off` tells Stata not to pause the display of the output
`cd`	sets the working directory, a very useful tool when working in groups. Example: `cd "D:\Project"`
`use`	loads Stata-format dataset. Example: `use dataset.dta, replace`
`sysuse`	use example dataset installed with Stata sysuse. Example: `sysuse auto, replace`
`webuse`	loads data from Stata website. Example: `webuse citytemp2, replace`
`save`	saves data in memory to file. Example: `save dataset.dta, replace`
`compress`	reduces the amount of memory used by your data. Example: `compress price`

Commands everyone should know

Inputting data into Stata and editing them

`import excel`	it reads worksheets from Microsoft Excel (.xls and .xlsx) files. Entire worksheets can be read, or custom cell ranges can be read. Example: `import excel auto.xls, clear`
`insheet`	reads text files (.csv files) created by a spreadsheet or database program. The data must be tab-separated or comma-separated, but not both simultaneously. A custom delimiter may also be specified. An observation must be on only one line and the first line of the file can optionally contain the names of the variables. Example: `insheet auto.csv, comma clear`
`edit`	edit brings up a spreadsheet-style data editor for entering new data and editing existing data. edit is a better alternative to input; see [D] input. Example: `edit`
`browse`	is similar to edit, except that modifications to the data by editing in the grid are not permitted. browse is a convenient alternative to list. Example: `browse`

Commands everyone should know

EXAMPLE: California School

See *Session1_LMEC21.do* do file:

```
clear all

capture log close

log using caschool.log, text replace

cd "C:Desktop¨

import excel caschool.xls, sheet("caschool") firstrow clear

browse

save caschool.dta , replace
```

Commands everyone should know

Basic data reporting

`describe`	describe contents of data in memory or on disk. Example: `describe`
`codebook`	produce a codebook describing the contents of data, so you know whether variables are string or numeric. Example: `codebook`
`count`	count number of observations satisfying a specified condition. Example: `count if price<5000`
`list`	list values of variables. Example: `list` (Don't type it if you have a lot of observations!)
`table`	tables of summary statistics. Example: `table make`
`tabulate`	one and two-way tables of frequencies. Example: `webuse citytemp2` `tab region agecat, row` includes row percentages `tabulate region agecat, col` includes col percentages

Commands everyone should know

Data manipulation

`generate, replace`	create or replace contents of variable. Example: `gen priceth=`*price*`/1000`
`egen`	extensions to generate. There is a list of egen functions that can be used with this command, check the help. Depending on the function, arguments are `varlist` or `numlist`, and the options depends on the funcion chosen. Example `egen mean = mean(price)`
`rename`	rename variable. Example: `ren make brand`
`drop and keep`	eliminate or keep variables or observations. Example: `drop price if price>7000`
`sort`	sort data alphabetically or ascending. Example: `sort price`
`order`	reorder variables in a dataset. Example: `order make foreign price`

Commands everyone should know

Data manipulation

`destring, tostring`	convert variables from string to numeric and it is useful when importing data from spreadsheets. Example: `tostring price, replace`
`encode, decode`	encode string into numeric and vice versa. It can be useful when using panel data. Example: `encode make, gen(id)`
`split`	splits the contents of a string variable into one or more parts using spaces by default Example: `split make`

`by & bysort`	this is a **prefix** and repeat Stata command on subsets of data, `bysort` sorts subsets ascending or alphabetically. Example:
	
	`by foreign: egen meanprice=mean(price)`
	
	`split make`
	`bysort make1: egen meanprice=mean(price)`

Commands everyone should know

EXAMPLE: California School

See *Session1_LMEC21.do* do file:

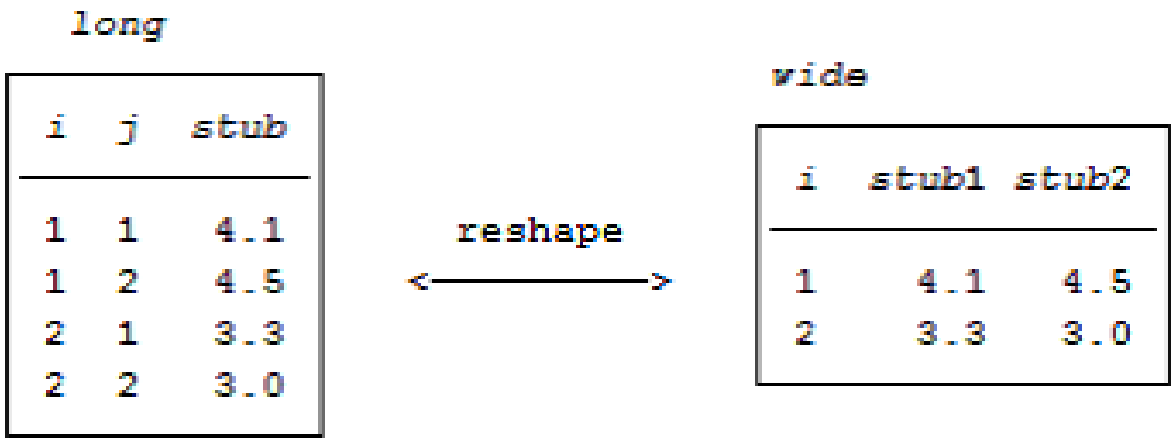
```
use caschool.dta , clear

label var read_scr "Avg reading score"

label var math_scr "Avg math score"

label var str "Student Teacher ratio"

count

describe read_scr math_scr str

codebook read_scr math_scr str

bys county:  egen cscore = mean(testscr)

label var cscore "County average test score"
```

Data management

Data management

Reshape

This convert data from wide to long and vice versa



To go from long to wide:
`reshape wide stub, i(i) j(j)`, where j is the existing variable

To go from wide to long:

`reshape long stub, i(i) j(j)`, where j is the new variable

Data management

Reshape

Example: From long to wide

	id	year	sex	inc	ue
1	1	80	0	5000	0
2	1	81	0	5500	1
3	1	82	0	6000	0
4	2	80	1	2000	1
5	2	81	1	2200	0
6	2	82	1	3300	0
7	3	80	0	3000	0
8	3	81	0	2000	0
9	3	82	0	1000	1

	id	sex	inc80	inc81	inc82	ue80	ue81	ue82
1	1	0	5000	5500	6000	0	1	0
2	2	1	2000	2200	3300	1	0	0
3	3	0	3000	2000	1000	0	0	1

Data management

Merge

Merge joins two datasets (master and using dataset) according to one (or more) key variables included in both datasets

```
merge [1:1, 1:m, m:1, m:m] varlist using filename
```

`merge 1:1`	One-to-one merge on specified key variables
`merge m:1`	Many-to-one merge on specified key variables
`merge 1:m`	One-to-many merge on specified key variables
`merge m:m`	Many-to-many merge on specified key variables
`merge 1:1 _n`	One-to-one merge by observation

Data management

Merge

When merging datasets, Stata creates a new variable `merge` that takes 5 values:

1. when the observation appears in the master dataset only
2. when the observation appears in the using dataset only
3. when the observation appears in both datasets and there are no variables in common
4. as in 3, but there is at least one variable in common, which has missing values in one of the two datasets
5. as in 4, but the common variables do not have missing values in both datasets

Data Management

EXAMPLE: California School

See *Session1_LMEC21.do* do file:

```
merge 1:1 id dist_cod using caschool_expenditure.dta

tab _merge

drop _merge

label var computer "Number of computers"

label var comp_stu "Computers per student"

label var expn_stu "Expenditures per student"
```

Data management

Append

Append adds observations at the end of the dataset in memory

```
append using <filename> [,options]
```

Main options:

`generate(newvar)`	it generates a variable that keeps track of the source of observations (0 for the master dataset, 1 for the first using dataset, 2 for the second using dataset...)
`keep(varlist)`	it selects the variables to be kept from the using dataset

Data management

Collapse and duplicates

- Collapse converts the dataset in memory into a dataset of means, sums, medians, etc. and clist must refer to numeric variables exclusively

```
collapse clist [if] [in] [weight] [, options]
```

1. You can set the statistic you want in the new dataset (mean is the default)
2. You can specify only one statistic per variable of observations
3. Stata will drop all the variables not specified in clist
4. You can specify `by(varlist)` in the options to aggregate statistics by levels of variables

- Duplicates drop duplicate observations, but also report them

```
duplicates drop varlist [if] [in] , force
```

Data management

Labels

Label data attaches a label (up to 80 characters) to the dataset in memory. Dataset labels are displayed when you use the dataset and when you describe it.

`label var`	This is explain or recall the meaning of a variable you can attach a label to it. Example: `label var price "Price in USD"`

`label define`	You may also want to assign a label to specific values of a variable. In order to do it, you first have to create a label for each value; then you have to attach the list of labels to the list of values.
	Example:
	`label define labelforeign 0 "Domestic" 1 "Foreign"`
	`label values foreign labelforeign`

`rename`	Finally, you may want to change the name of a variable in the dataset. Example: `rename make brand`

Stata Tricks

Stata Tricks

Continuation lines, comments and delimiters

`[space] ///`	is used to indicate a continuation line during the execution of a command. Example: `twoway (scatter medage popurban) /// (lfit medage popurban)`
`//`	is used to comment code lines. Example: `graph twoway(scatter medage popurban) // Graph`
`/* */`	indicates that all the text between the opening `/*` and the closing `*/`, ignored by Stata.
`display`	Turns Stata into a powerful calculator. Example: `di 50*2`

Stata Tricks

System variables (_variables) or underscore variables

These are built-in system variables that are created and updated by Stata while running and contain information about the data such as the the value (to machine precision) of regression coefficients.

_n	contains the number of the current observation. Example: `gen id=_n`
_N	contains the total number of observations in the dataset These variables are useful for indexing observations or generating sequences of numbers. _n can act as a running counter within a by-group and _N acts as the total number within each by-group. Example: `bysort foreign: gen count=_N`

Stata Tricks

Missing values

Sometimes, a data set may have "holes" in it, that is, **missing values**. Some statistical procedures such as regression analysis will not take into account the missing observations, called "listwise deletion" or "complete cases only", see `help missing`.

- if the variable is **numeric**, the missing value is represented by a dot . Example: `gen class_age=.`
- if the variable is a **string**, the missing value is represented by "" (blank)
- be careful! Stata also interprets dots as extremely large numbers. Example: `browse if class_age>10` would show you data for all with class_age strictly above 10 and also those with missing data

Some useful egen functions

Type help egen for more

`count(exp)`	creates a constant (within varlist) containing the number of non-missing observations of exp
`diff(varlist)`	may not be combined with by. It creates an indicator variable equal to 1 if the variables in varlist are not equal and 0 otherwise
`max(exp)`	creates a constant (within varlist) containing the maximum value of exp
`mdev(exp)`	returns the mean absolute deviation from the mean (within varlist) of exp
`mean(exp)`	creates a constant (within varlist) containing the mean of exp
`median(exp)`	creates a constant (within varlist) containing the median of exp
`min(exp)`	creates a constant (within varlist) containing the minimum value of exp
`total(exp)`	creates a constant (within varlist) containing the sum of exp treating missing as 0

Descriptive Statistics and Graphical Analysis

Outline

Features of good descriptive statistics

A good descriptive statistics must:

1. allow the reader to understand the **structure of the dataset**
2. highlight **relevant elements** for the econometric analysis using tables and figures
3. not provide irrelevant or redundant information

Outline

Structure of the datasets

Datasets are classified according to the number of units N and the number of times T each unit is observed

`Cross-section`	N different units observed only once (i.e. 1980 Census data by state, where states are observed just once). Example:
	`sysuse census, replace`
	`tab state`

`Time-series`	one unit observed for T periods (i.e. values of one financial share observed during a time span). Example: `sysuse sp500, replace`

`Panel`	N different units observed for T different periods (i.e. several cohorts of individuals observed from 1976 to 1984 in a labour force survey). Example:
	`webuse abdata.dta`
	`tab id`

Outline

How to tell Stata about the structure of the datasets

Stata automatically reads cross-sections, thus you have to declare when data are time-series or panel

`tsset`	declares the data in memory to be a time series. `tssetting` the data is what makes Stata's time-series operators such as L. and F. (lag and lead) work. There specific time-series command that work only if you have `tsset` the data first. If you save the data after `tsset`, the data will be remembered as time series. Example:
	`sysuse sp500, replace`
	`tsset date, daily`

`xtset`	declares data to be panel data. In the declare syntax, you have to specify the a `panelvar` that identifies the individuals and a optional `timevar` that identifies the times within the panel. If you save the data after `xtset`, the data will be remembered as panel. Example:
	`webuse abdata.dta`
	`xtset id year, yearly`
	Panel are **balanced** when all individuals are observed along the whole time span.

Summary statistics and test

Summary statistics and tests

Summary statistics and correlations

`summarize`	summarize calculates and displays a variety of univariate summary statistics. If no varlist is specified, summary statistics are calculated for all the variables in the dataset. Example:
	`sum testscr str`
	`sum testscr str, detail`

`ci`	ci computes standard errors and confidence intervals for each of the variables in varlist. Example to obtain normal-approximation 90% confidence intervals for means of normally distributed variables: `ci testscr str, level(90)`
`correlate`	displays the covariance matrix for a group of variables. Example: `corr testscr str`

Summary statistics and tests

Summary statistics available after specifying tsset or xtset

`tsreport`	reports time gaps in a sample of observations. Example:
	`webuse tsrptxmpl`
	`list edlevel month income`
	`tsreport, report panel`

`tsfill`	is used after `tsset` to fill in gaps in time-series data and gaps in panel data with new observations, which contain missing values. Example:
	`webuse tsfillxmpl`
	`list mdate income`
	`tsfill`
	`list mdate income`

`xtsum`	summarize xt data. Example: `xtsum`
	`webuse nlswork`
	`xtset id year, yearly`
	`( http://www.stata.com/manuals13/xtxtsum.pdf)`

Summary statistics and tests

Distribution of random variables

`mvtest normality`	performs tests for univariate, bivariate, and multivariate normality.
	
	Example: we use data on three species of the flower iris. We have a sample of 50 observations on the length and width of the sepal and petal for each specie. We hypothesize that these features might be normally distributed within species, though they are likely not normally distributed across species. We will examine the *Iris setosa* data (`iris==1`).
	
	`webuse iris`
	`mvtest norm pet* sep* if iris==1, ///`
	`bivariate univariate stats(all)`

`cumul`	creates a new variable, defined as the empirical cumulative distribution function of the variable we are interested in. Example:
	
	`webuse hsng`
	`cumul faminc, gen(cumul)`
	`line cumul faminc, sort`

Summary statistics and tests

Mean, median and centiles

`mean`	produces estimates of means, along with standard errors. Example: we want to estimate the average mileage of the cars without the fuel treatment (`mpg1`) and those with the fuel treatment (`mpg2`).
	`webuse fuel`
	`mean mpg1 mpg2`

`centile`	estimates specified centiles and calculates confidence intervals. If no `varlist` is specified, centile calculates centiles for all the variables in the dataset, otherwise medians (centile(50)) are reported.
	Example:
	`sysuse auto, clear`
	`centile price` – 50th percentile, median –
	`centile price, centile(5 50 95)`
	`centile price, level(99)`

Summary statistics and tests

ttest on the equality of means

`ttest`	performs t tests on the equality of means. In the first form, `ttest` tests that `varname` has a mean of #. In the second, ttest tests that varname1 and varname2 have the same mean. In the third form, `ttest` tests that varname has the same mean within the two groups defined by `groupvar`.
	Example:
	`sysuse auto`
	`ttest mpg==20`
	`webuse fuel`
	`ttest mpg1==mpg2`
	`webuse fuel3`
	`ttest mpg, by(treated)`

Tables

Tables

Tables of summary statistics, table

Stata has several commands to build tables of summary statistics

`table`	table calculates and displays tables of statistics. In the first form it shows a one-way table with frequencies. In the second, it shows a one-way table with count of nonmissing observations for mpg. In the third, it adds multiple statistics on mpg. In the fourth, it shows a two-way table wth frequencies.
	Example:
	`sysuse auto`
	`table rep78`
	`table rep78, contents(n mpg)`
	`table rep78, c(n mpg mean mpg sd mpg median mpg)`
	`table rep78 foreign`

Tables

Tables of summary statistics, tabstat

`tabstat`	displays summary statistics for a series of numeric variables in one table, possibly broken down on (conditioned by) another variable. In the first form it shows the mean of the `varlist`. In the second, it shows the mean by categories of foreign. In the third, it adds also standard deviation, minimum, and maximum.
	Example:
	`sysuse auto`
	`tabstat price weight mpg rep78`
	`tabstat price weight mpg rep78, by(foreign)`
	`tabstat price weight mpg rep78, by(foreign) stat(mean sd min max)`

Tabulate

One- and two-way tables of frequencies

`tabulate` is a simple but very important command that provides one- and two-way tables of frequencies

`tabulate`	produces two-way tables of frequency counts, along with various measures of association, including the common Pearson's chi-squared, the likelihood-ratio chi-squared, Fisher's exact test, etc. It includes a list of options such `col` and `row` to display the relative frequency of each cell within its column/row in a two-way table.
	Example:
	`webuse citytemp2`
	`tabulate region agecat`
	`tabulate region agecat, row`
	`tabulate region agecat, column`
	`tabulate region agecat, cell`

Graphical Analysis

Graphical Analysis

Introduction

- Stata has excellent graphic facilities, accessible through the command `graph`
- There are several families of plots, the most common are:

  1. **Twoway graphs** are X-Y plots showing points (`scatter`) or lines (`line`)
  2. **Graph bar** are used for descriptive statistics
  3. **Histograms** are used for showing the density of a variable

- From Stata 10 a graphics editor that can be used to modify a graph interactively is introduced.

Graphical Analysis

Twoway graphs syntax

```
twoway (line mpg weight, [line options]) ///
(scatter mpg weight, [scatter options]), [twoway options]
```

- Twoway graphs show the relationship between numeric data
- What distinguishes a twoway graph is that it fits onto numeric y and x axes
- **Twoway** is called a **graph**, what appeared in the graphs are called **plots**, i.e. a scatter or a line
- You can specify **options** for each plot within () and for the twoway graph, type `twoway_options`

Graphical Analysis

Twoway plot types

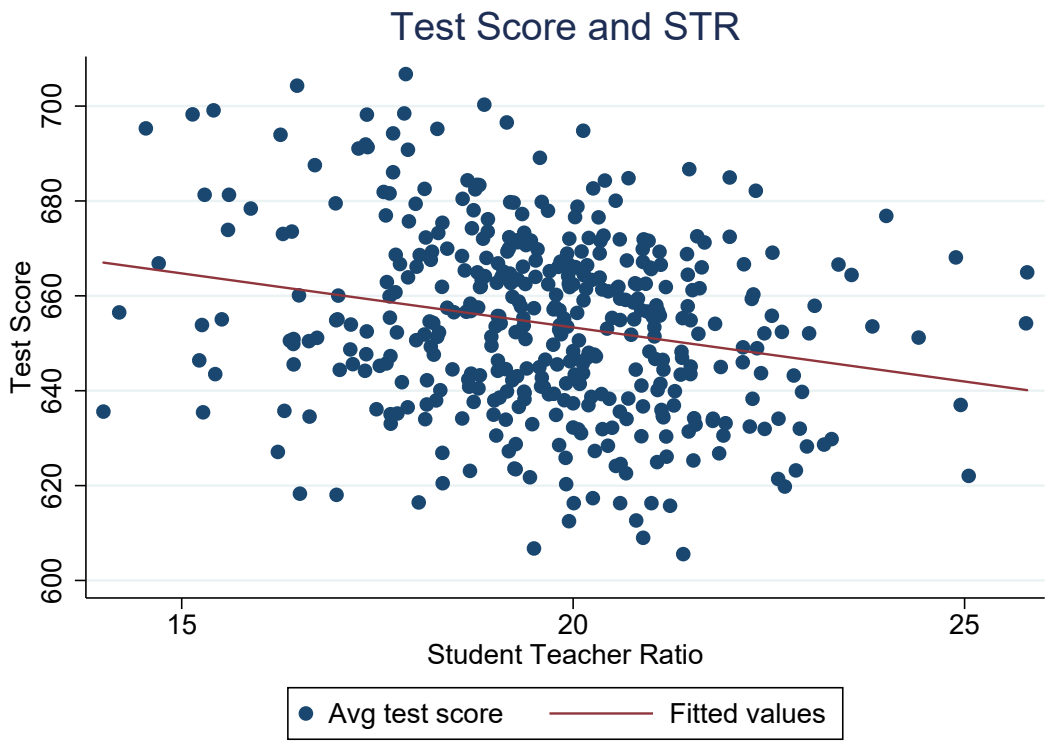
There are many plot types for `twoway`, type `help twoway`

`scatter`	scatterplot
`line`	line plot
`connected`	connected-line plot
`scatteri`	scatter with immediate arguments
`area`	line plot with shading
`bar`	bar plot
`spike`	spike plot
`dropline`	dropline plot
`dot`	dot plot

Graphical Analysis

EXAMPLE: California School

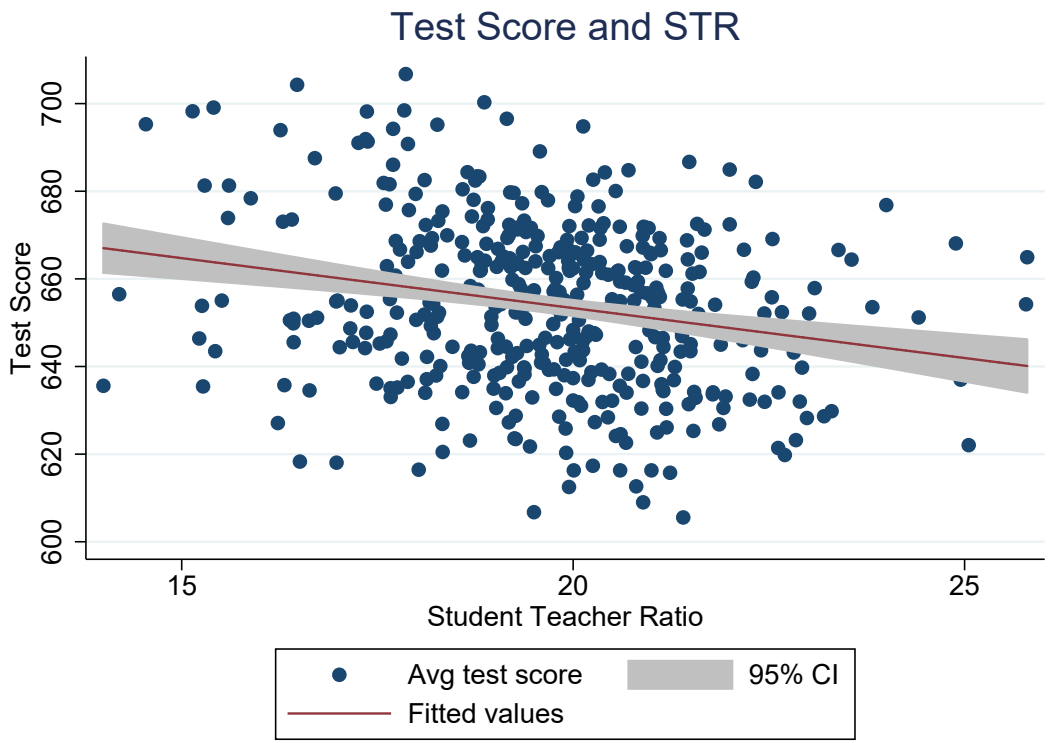
```
graph twoway (scatter testscr str) (lfit testscr str) , ///
title("Test Score and STR") ytitle("Test Score") xtitle("Student Teacher Ratio") ///
graphregion(color(white))
gr save testscore , replace
gr export testscore.pdf , replace
```



Graphical Analysis

EXAMPLE: California School

```
graph twoway (scatter testscr str) (lfitci testscr str) , ///
title("Test Score and STR") ytitle("Test Score") xtitle("Student Teacher Ratio") ///
graphregion(color(white))
gr save testscore_ci , replace
gr export testscore_ci.pdf , replace
```



Graphical Analysis

Graph bar syntax

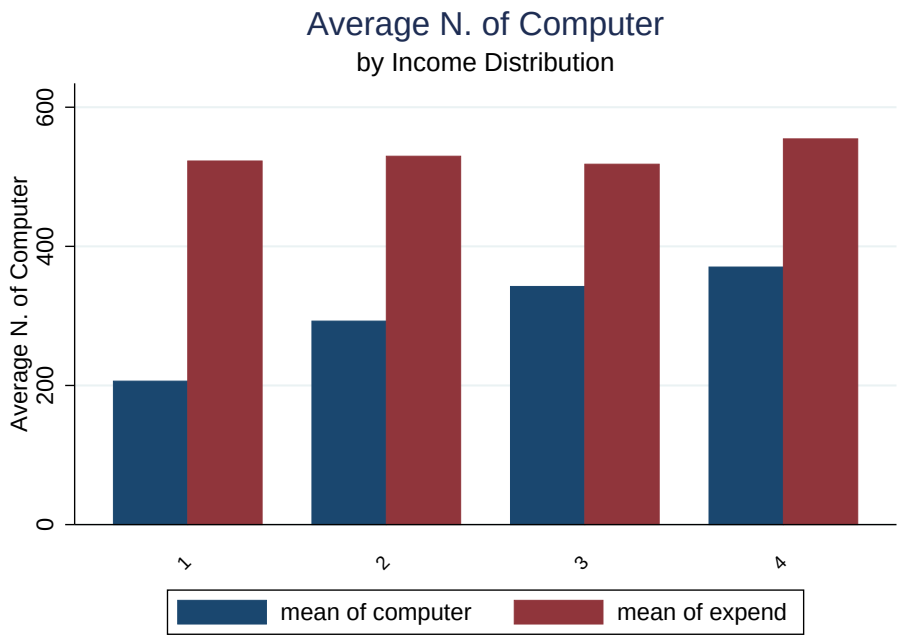
```
graph bar (mean) numeric_var, over(cat_var)
```

- graph bar draws vertical bar charts of **summary statistics**. In a vertical bar chart, the y axis is numerical, and the x axis is categorical
- You can also draw **horizontal bar** charts. In a horizontal bar chart, the numerical axis is still called the y axis, and the categorical axis is still called the x axis, but y is presented horizontally, and x vertically
- The syntax for vertical and horizontal bar charts is **the same**; all that is required is changing `graph bar` to `hbar`

Graphical Analysis

EXAMPLE: California School

```
sum avginc
gen income=.
replace income=1 if avginc<10.63
replace income=2 if avginc>=10.63 & avginc<13.72
replace income=3 if avginc>=13.72 & avginc<17.63
replace income=4 if avginc>=17.63
tab income
gr bar (mean) computer expend , over(income , label(angle(45) labsize(small) labgap(5))) ///
blabel(bar, position(inside) format(%9.1f) color(white)) ytitle("Average N. of Computer") ///
title("Average N. of Computer") subtitle("by Income Distribution")
```



Graphical Analysis

Histograms syntax

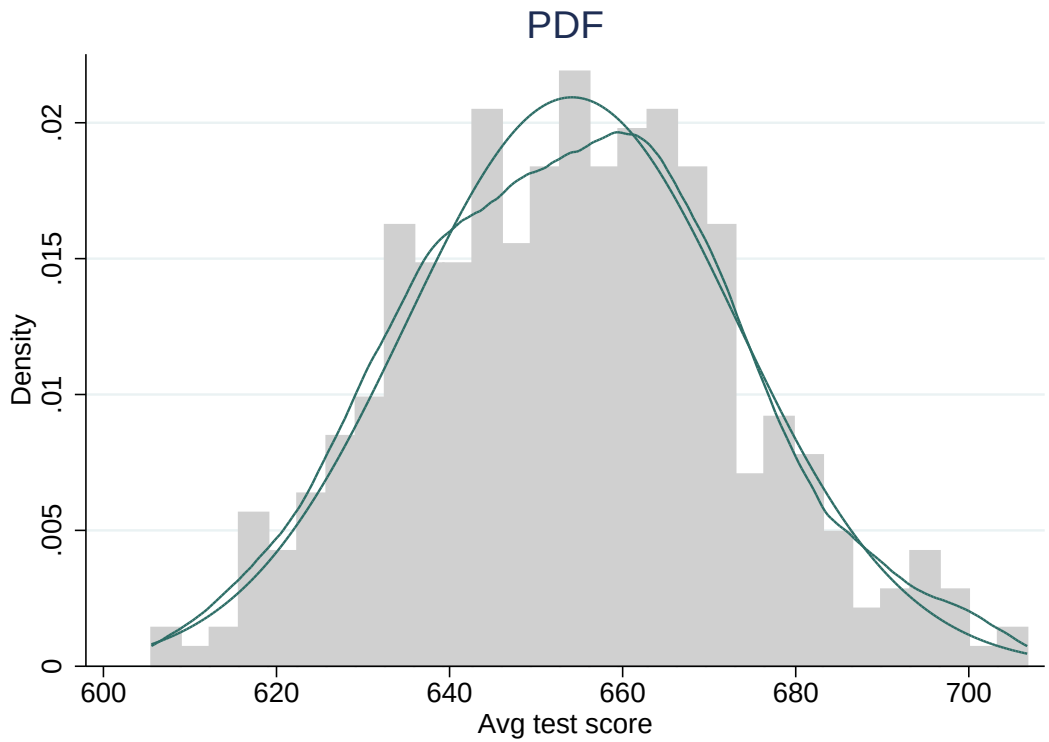
```
hist varname [if] [in] [weight] [, options]
```

- draws histograms of a variable, which is assumed to be the name of a continuous variable unless the discrete option is specified
- You can specify how the data are to be aggregated into bins: use the **option bin()** by specifying the number of bins and the option **width()** by specifying the bin width
- The **option normal** specifies that the histogram be overlaid with an appropriately scaled normal density. The normal will have the same mean and standard deviation as the data

Graphical Analysis

EXAMPLE: California School

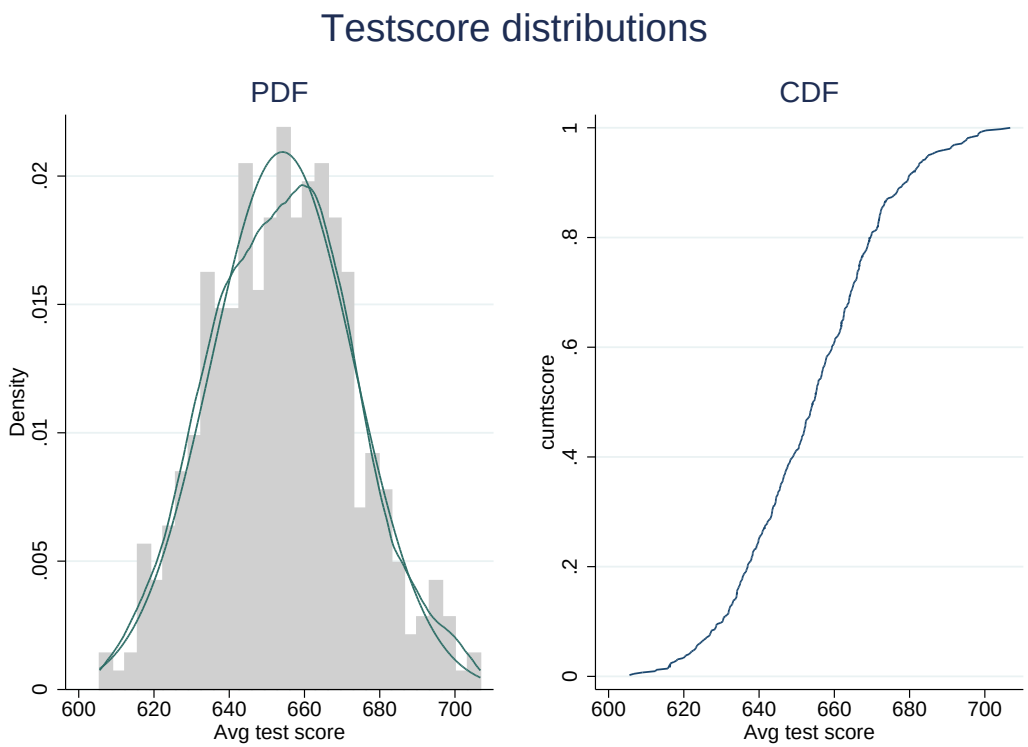
```
histogram testscr , bin(30) kden normal legend(off) color(gs13) title(PDF) ///
graphregion(color(white))
gr save histscore , replace
gr export histscore.pdf , replace
```



Graphical Analysis

EXAMPLE: California School

```
cumul testscr, gen(cumtscore)
line cumtscore testscr, sort title(CDF) graphregion(color(white))
gr save cumtestscr , replace
graph combine "histscore" "cumtestscr" , title("Testscore distributions") graphregion(color(whi
gr save pdfcdf , replace
gr export pdfcdf.pdf , replace
```



Graphical Analysis

Graph utilities

Here is a list of useful tools you can use with `graph`

`graph save`	save graph to disk.
	Example: `graph save mygraph`
`graph use`	redisplay graph stored on disk.
	Example: `graph use mygraph`
`graph display`	redisplay graph stored in memory using a different scheme or style. To see a list of available schemes type `graph query, schemes`
	Example: `graph display, scheme(economist)`
`graph combine`	combine multiple graphs
	Example: `graph combine mygraph1 mygraph2`
`graph export`	export .gph file to an image
	Example: `graph export mygraph.png, as(png)`

Stata Tricks

Stata Tricks

Abbreviations, * and sets of variables

`abbreviations`	most commands and variable names can be abbreviated, check the help for the underlined part of each command.
	
	Example:
	`help tabulate`
	`ta` instead of `tabulate`
	`webuse iris`
	`des ir`

`[prefix]*`	indicates all the variables starting with a prefix. Example `webuse iris` `des sep*` – describes all variables starting with sep
`var1-var15`	includes all the variable from `var1` to `var15`. Example: `des iris-petwid`
`CTRL+H`	Open Stata replace facility (useful when editing do files)

Stata Tricks

Capture

`capture` is a useful command to start your do file

`capture`	capture executes command, suppressing all its output (including error messages, if any). It can be useful when used with opening and closing **log files**. Unlike `quietly`, `capture` ensures that Stata will continue to run if Stata returns an error message.
	
	Example, when you start your do file, follow this code:
	
	`capture: log close`
	`log using filename, text replace`
	`[your lines of code]`
	`log close filename`

Preliminary: Multiple linear regression

Multiple linear regression

Regress command

If you want to estimate the following model:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \epsilon_i$$

You can do it by using the command `regress`:

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

- `regress` **fits a model** of dependent variable on independent variables using ordinary least squares regression method
- Stata **drops** any observation where a regressor contain a missing value
- Stata **stores** all the results generated such as regression coefficients, number of observations, tests etc. so that they can be easily recalled and used, type `help regress`
- Hint: type `return list` or `ereturn list` after `regress` for a list of the stored results

Multiple linear regression

Regress command, Options

`regress` has the following useful options:

`nocons`	suppresses constant term
`vce()`	specifies the type of standard error reported. For example, `robust` specifies heteroscedastic robust standard errors; `cluster` *clustvar* allows for intragroup correlation; by default classical ols standard errors are reported.
`beta`	specifies to report the standardized beta coefficients. The beta coefficients are the regression coefficients obtained by first standardizing all variables to have a mean of 0 and a standard deviation of 1.
`level()`	set confidence interval level, the default is level(95)

Multiple linear regression

EXAMPLE: Child Birth Weight

This data set contains information on 1,388 births.

```
bcuse bwght , replace
lab var bwght "birth weight"
lab var cigs "number of cigarettes mother smoked"
lab var parity "birth order"
lab var faminc "annual family income"
lab var motheduc "years of schooling for mother"
lab var fatheduc "years of schooling for father"
```

We want to estimate the following model:

$$bwght = \beta_0 + \beta_1 cigs + \beta_2 parity + \beta_3 faminc + \beta_4 motheduc + \beta_5 fatheduc + u$$

Multiple linear regression

EXAMPLE: Child Birth Weight

```
reg bwght cigs parity faminc motheduc fatheduc
```

```
. reg bwght cigs parity faminc motheduc fatheduc

      Source |       SS           df       MS      Number of obs   =     1,191
-------------+----------------------------------   F(5, 1185)      =      9.55
       Model |  18705.5567         5  3741.11135   Prob > F        =    0.0000
    Residual |  464041.135     1,185  391.595895   R-squared       =    0.0387
-------------+----------------------------------   Adj R-squared   =    0.0347
       Total |  482746.692     1,190  405.669489   Root MSE        =    19.789

------------------------------------------------------------------------------
       bwght |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        cigs |  -.5959362   .1103479    -5.40   0.000    -.8124352   -.3794373
      parity |   1.787603   .6594055     2.71   0.007     .4938709    3.081336
      faminc |   .0560414   .0365616     1.53   0.126    -.0156913    .1277742
    motheduc |  -.3704503   .3198551    -1.16   0.247    -.9979957    .2570951
    fatheduc |   .4723944   .2826433     1.67   0.095    -.0821426    1.026931
       _cons |   114.5243   3.728453    30.72   0.000     107.2092    121.8394
------------------------------------------------------------------------------
```

Multiple linear regression

Regress, Output 1

From the regression output we obtain the following information:

`Source`	is the source of variance (due to independent variables or residuals)
`SS(T)`	is the total sum of squares, i.e. the total variability around the mean $SST = SSE + SSR$
`dF`	are the degrees of freedom associated with the sources of variance. Total: N-1; Model: K-1; Residual: N-K-1
`MS`	mean squares, i.e. the sum of squares divided by the degrees of freedom
`F(K-1, N-K-1)`	it is a measure of the joint significance of the coefficients and it is given by (Mean Squares Model)/(Mean Squares Residual)

Multiple linear regression

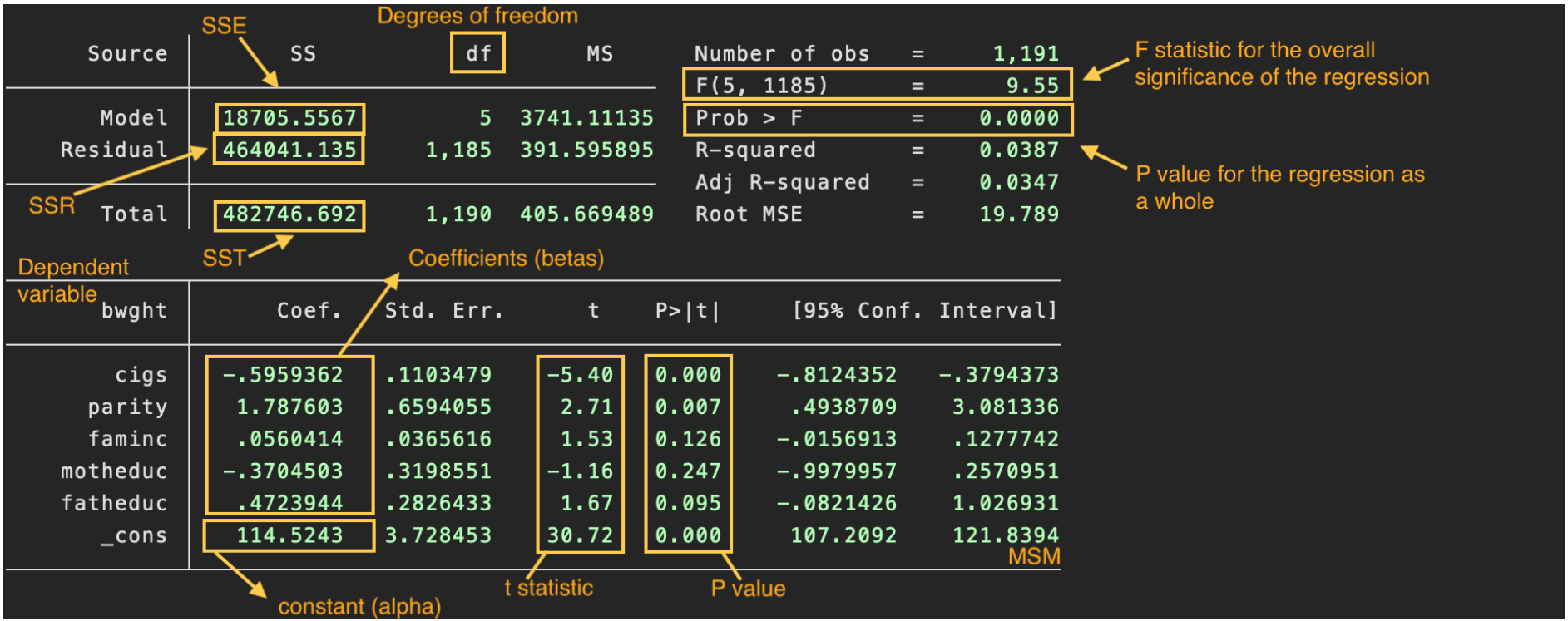
Regress, Output 2

and also these:

`Prob>F`	it is the p-value associated with F, i.e. the highest probability at which the test fails to reject the null hypothesis
`R-squared`	it is the proportion of variance of the dependent variable that can be explained by the variance of the independent variables
`Root MSE`	it is the standard deviation of the error term
`t`	it is the t statistics value of a two-sided test on the coefficient
`P>t`	is the p-value associated to t. If $p - value < 0.05$ you can reject the null hypothesis at the 5% significance level

Multiple linear regression

EXAMPLE: Child Birth Weight (Annotated output)



Source	SS	df	MS
Model	18705.5567	5	3741.11135
Residual	464041.135	1,185	391.595895
Total	482746.692	1,190	405.669489

Number of obs	1,191
F(5, 1185)	9.55
Prob > F	0.0000
R-squared	0.0387
Adj R-squared	0.0347
Root MSE	19.789

bwght	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
cigs	-.5959362	.1103479	-5.40	0.000	-.8124352	-.3794373
parity	1.787603	.6594055	2.71	0.007	.4938709	3.081336
faminc	.0560414	.0365616	1.53	0.126	-.0156913	.1277742
motheduc	-.3704503	.3198551	-1.16	0.247	-.9979957	.2570951
fatheduc	.4723944	.2826433	1.67	0.095	-.0821426	1.026931
_cons	114.5243	3.728453	30.72	0.000	107.2092	121.8394

Partialling Out

$$y = \beta_1 X_1 + \beta_2 X_2 + u \qquad (3.36)$$

Theorem 3.15.1 ***Frisch-Waugh-Lovell (FWL)***
In the model (3.36), the OLS estimator of $\boldsymbol{\beta_2}$ *and the OLS residuals* $\hat{\boldsymbol{e}}$ *may be equivalently computed by either the OLS regression (3.37) or via the following algorithm:*

1. *Regress* $\boldsymbol{y}$ *on* $\boldsymbol{X_1}$, *obtain residuals* $\widetilde{\boldsymbol{e}}_1$;
2. *Regress* $\boldsymbol{X_2}$ *on* $\boldsymbol{X_1}$, *obtain residuals* $\widetilde{\boldsymbol{X}}_2$;
3. *Regress* $\widetilde{\boldsymbol{e}}_1$ *on* $\widetilde{\boldsymbol{X}}_2$, *obtain OLS estimates* $\widehat{\boldsymbol{\beta}}_2$ *and residuals* $\widehat{\boldsymbol{e}}$.

Hansen (2015), Econometrics

filippo.pavanello2@unibo.it