

# Quantitative Methods Bootcamp

Rocio Mendez Pineda & Tobias Rüttenauer

2023-09-28

# Before We Start

- Go to: <https://github.com/ruettenauer/Bootcamp>
- Download:
  - The slides for this bootcamp
  - The dataset we will use (“WDI\_Data.dta”)
- Make sure to save the dataset in an easy-to-access folder
  - A place you can also access from elsewhere (e.g., N-drive)

# Objectives of This Bootcamp

## Overarching goal

Be on common starting point + ready to take on the quantitative MSc modules

- Refresh key statistical concepts
  - E.g., sampling distributions, hypothesis testing
- Obtain basic familiarity with R & Stata

# Statistical Inference

Statistical inference is used to learn from incomplete data

We wish to learn some characteristics of a population (e.g., the mean and standard deviation of the heights of all women in the UK), which we must estimate from a sample or subset of that population

## Two types of inference:

1. **Descriptive inference:** What is going on, or what exists?
2. **Causal inference:** Why is something going on, why does it exist?

# Example data

The code below loads the WDI packages and searches for an indicator on CO2 per capita.  
It uses the statistics software R (more later).

```
1 # load package
2 library(WDI)
3
4 # Search GDP per capita (log-transformed)
5 WDIsearch("CO2.*capita")
```

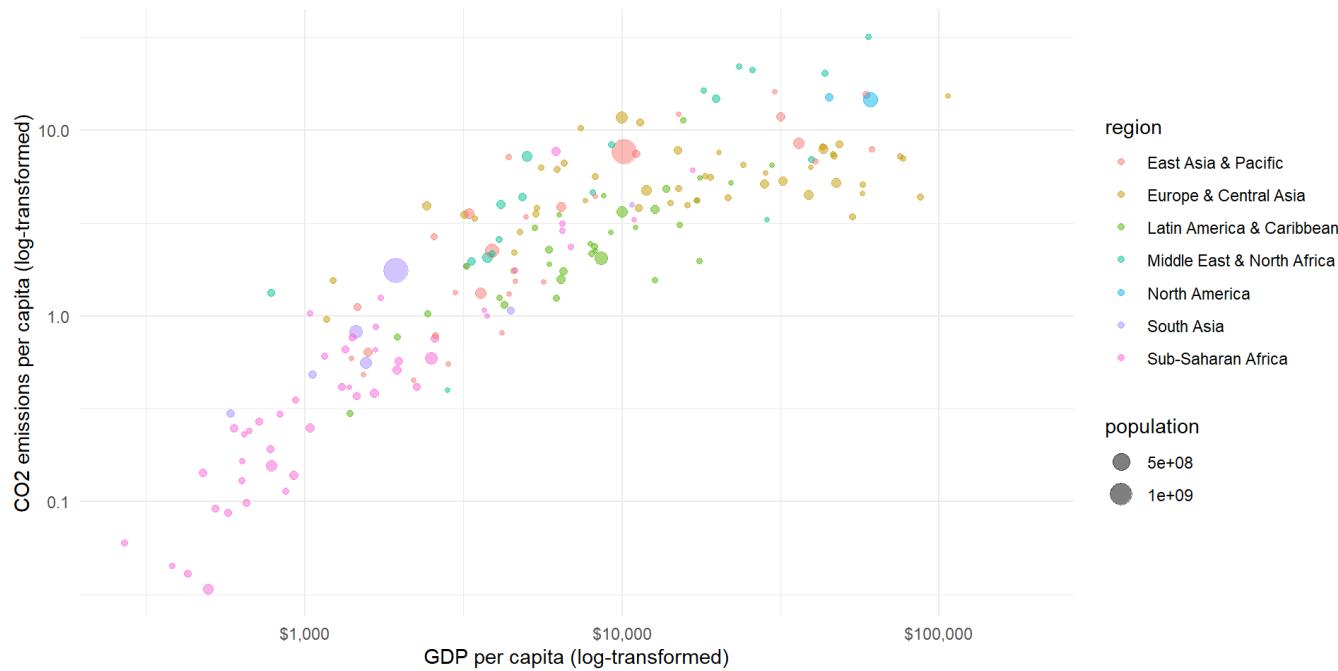
	indicator	name
6032	EN.ATM.CO2E.PC	CO2 emissions (metric tons per capita)
6048	EN.ATM.METH.PC	Methane emissions (kt of CO2 equivalent per capita)
6059	EN.ATM.NOXE.PC	Nitrous oxide emissions (metric tons of CO2 equivalent per capita)

The code below uses the WDI API to retrieve the data and creates a dataframe of three indicators.

```
1 # Define countries, indicators form above, and time period
2 wd.df <- WDI(country = "all",
3                 indicator = c('population' = "SP.POP.TOTL",
4                               'gdp_pc' = "NY.GDP.PCAP.KD",
5                               'co2_pc' = "EN.ATM.CO2E.PC"),
6                 extra = TRUE,
7                 start = 2019, end = 2019)
8
9 # Drop all country aggregates
```

# Descriptive Inference

► expand for full code



**Descriptive:** What are the average CO2 emissions of all European countries?

# Descriptive Inference

What are the average CO2 emissions of all European countries?

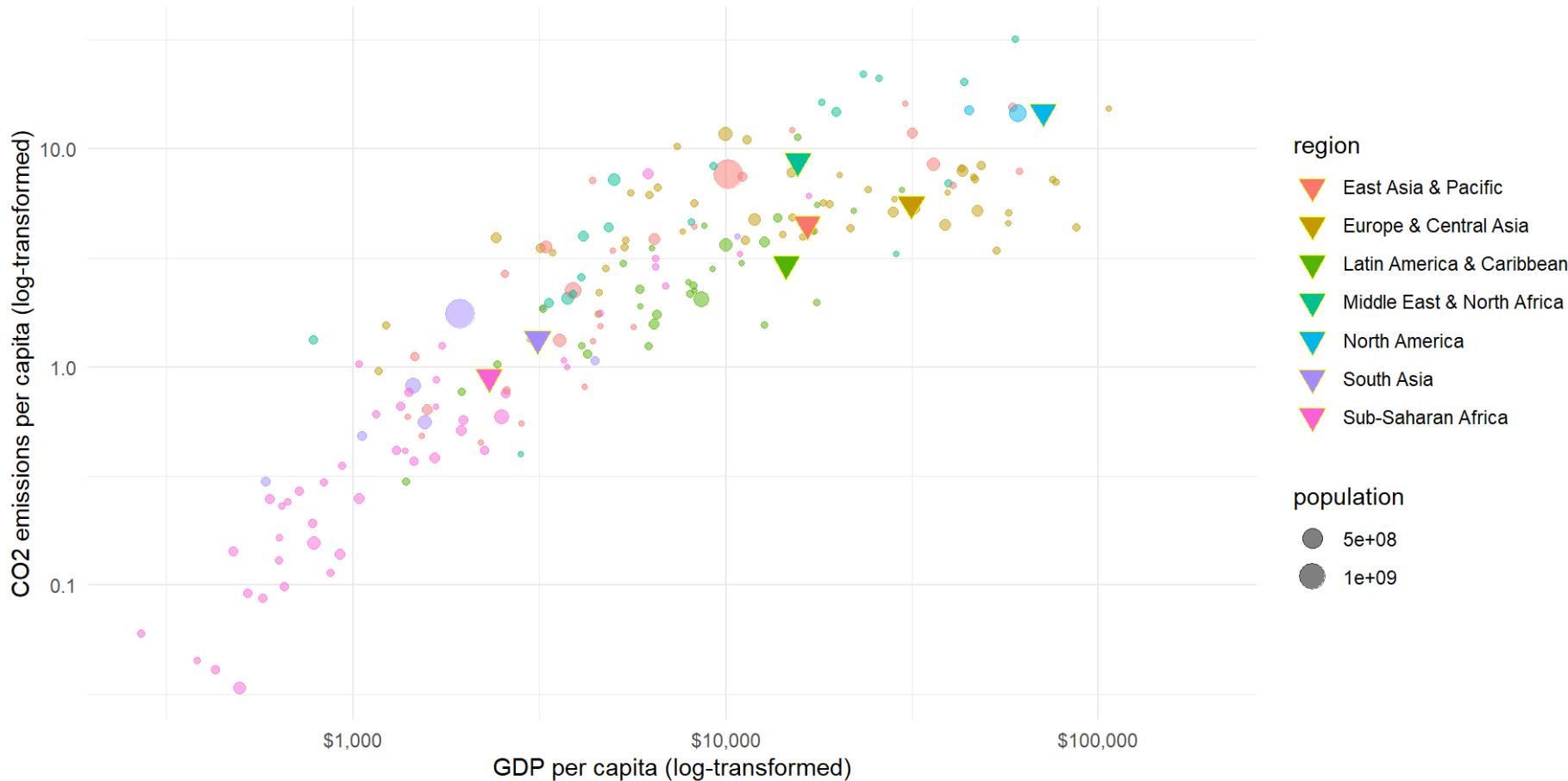
Mean by region

```
1 mean_co2 <- mean(wd.df$co2_pc[which(wd.df$region == "Europe & Central Asia")]
2 mean_co2
[1] 5.608387
```

# Descriptive Inference

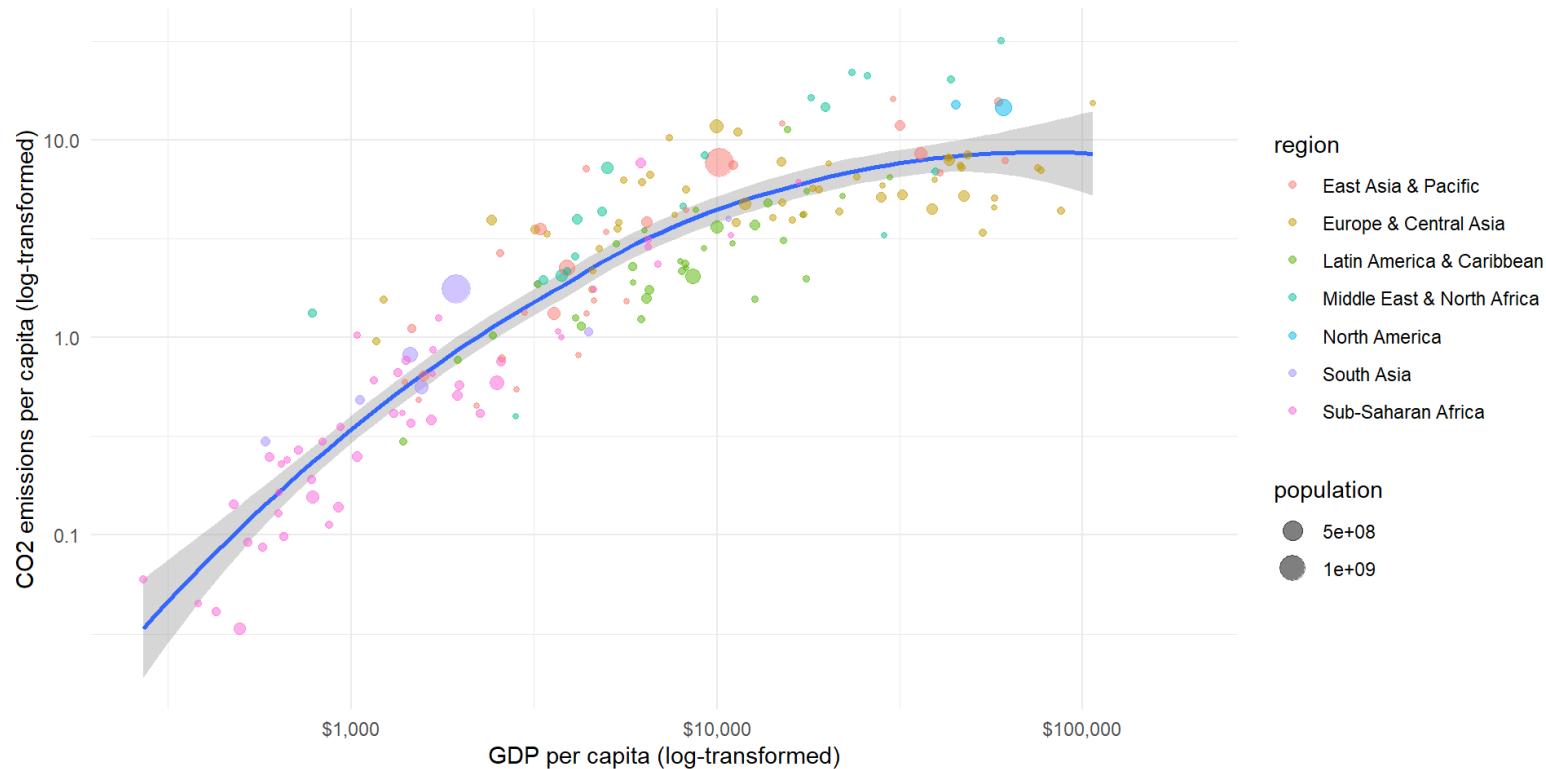
What are the average CO2 emissions of all European countries?

► Code



# Causal Inference

► expand for full code



Causal: How does GDP influence the amount of CO2 emissions?

# Variables and Observations

A **variable** is anything that can vary across units of analysis: it is a characteristic that can have multiple values

- E.g., sex, age, diameter, financial revenue, temperature

A **unit of analysis** is the major entity that you analyse

- E.g., individuals, objects, schools, countries

An **observation** is the value of a particular variable for a particular unit (sometimes a unit is in its entirety referred to as observation)

- E.g., the **individual** King Charles III is **73 years** of **age**

# Different Types of Variables

## Continuous / interval-ratio variables:

They have an ordering, they can take on infinitely many values, and you can do calculations with them

- E.g., income, age, weight, minutes

## Categorical variables:

Each observation belongs to one out of a fixed number of categories

- Ordinal variables: there is a natural ordering of the categories
- Nominal variables: there is no natural ordering of the categories
- E.g., education level, Likert scales, gender, vote choice

# Describing variables

# Describing variables

Describing data is necessary because there is usually too much of it, so it does not make any sense to look at every data point

We thus have to look for ways to summarize central tendencies, variation, and relationships that exist in the data

There are many different ways to do this

1. Visual depictions
  2. Numerical descriptions
- E.g. mean, mode, median, standard deviation

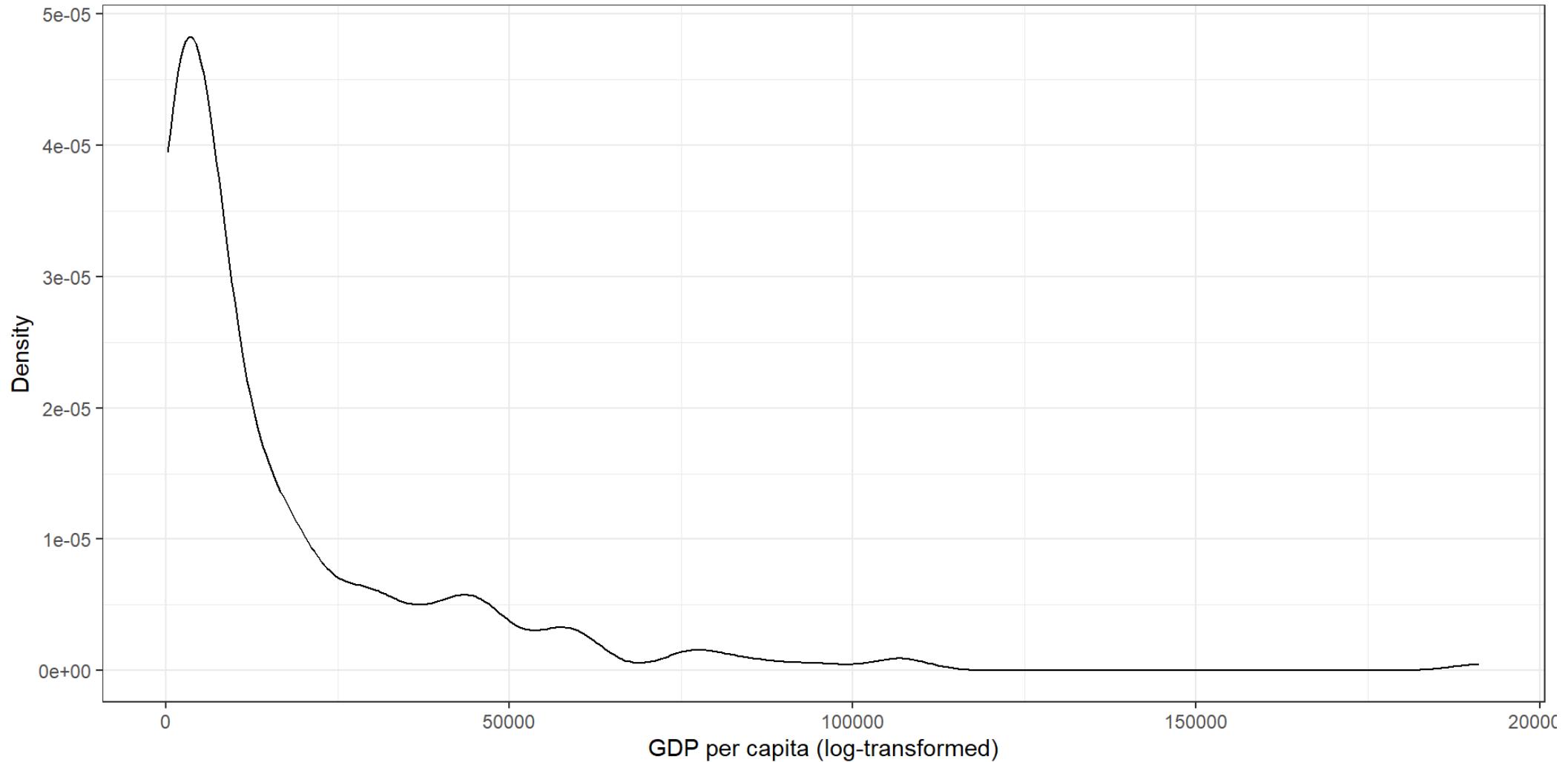
# Distribution

Variables can be characterized by their **frequency distribution**:

The distribution of the (relative) frequencies of their values

- E.g., we can graph the world income distribution:

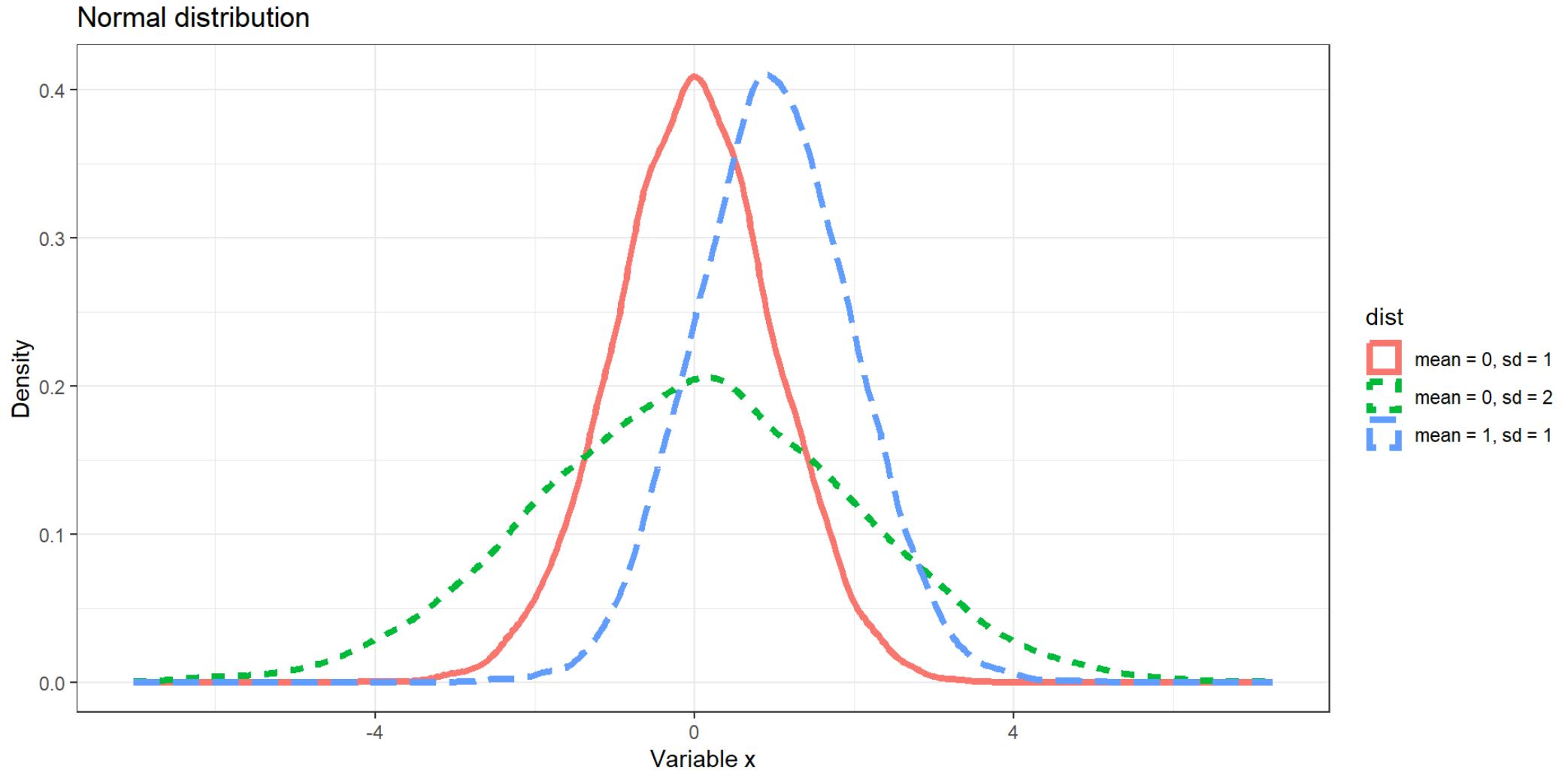
► expand for full code



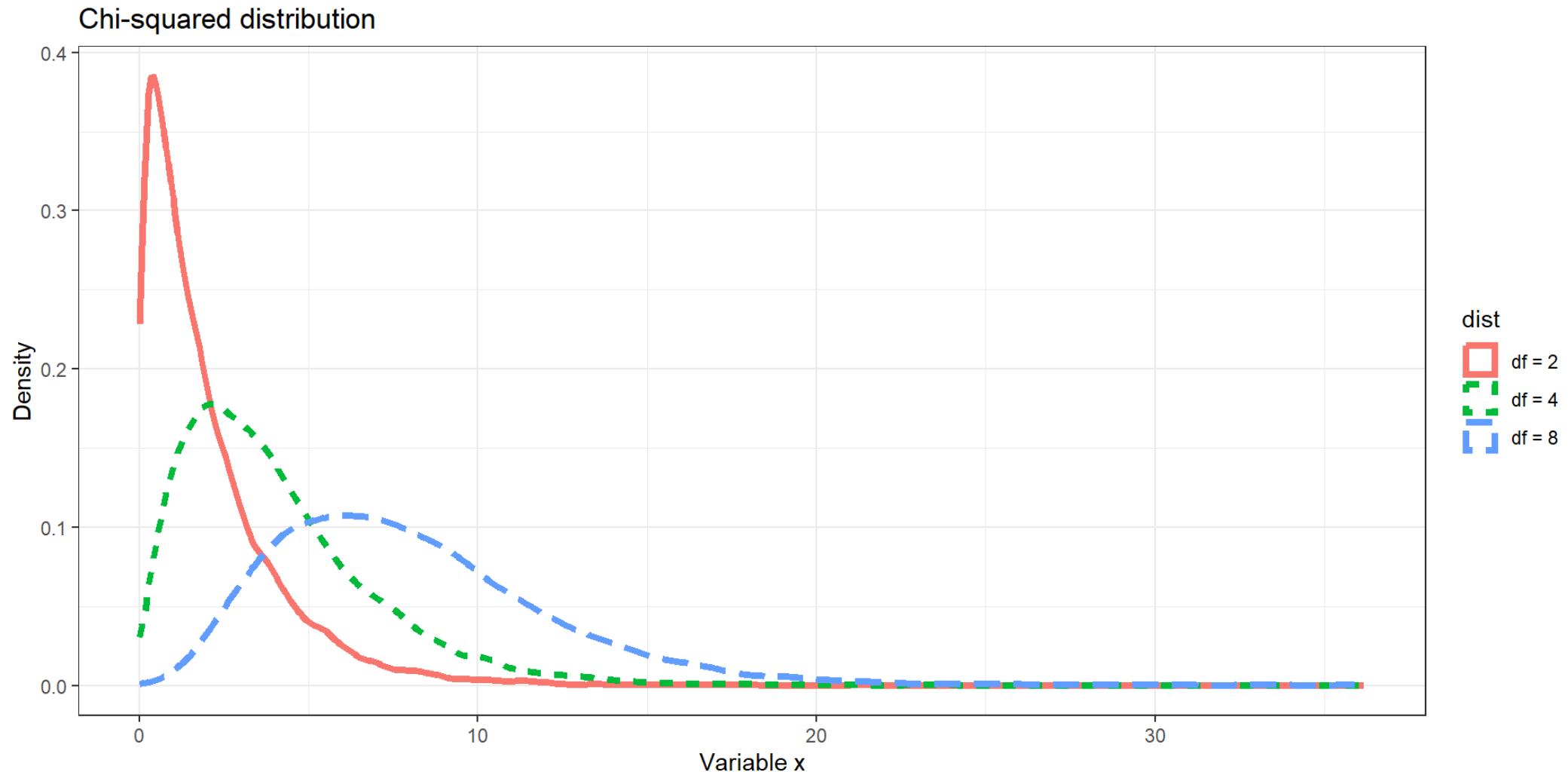
# Distributions: Examples

- Normal distribution
- Chi-squared distribution

► expand for full code

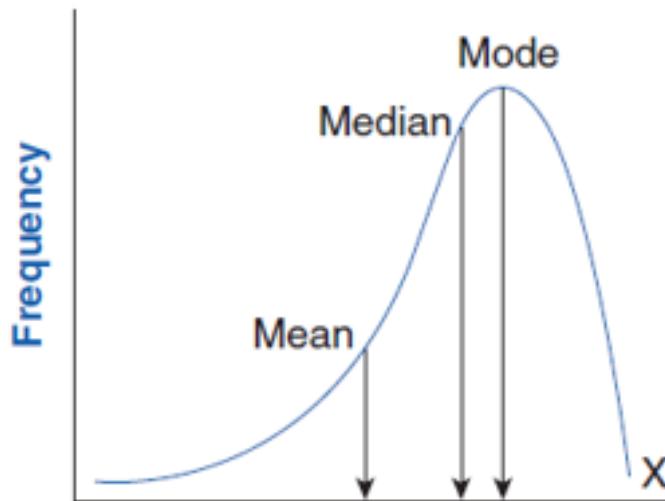


► expand for full code

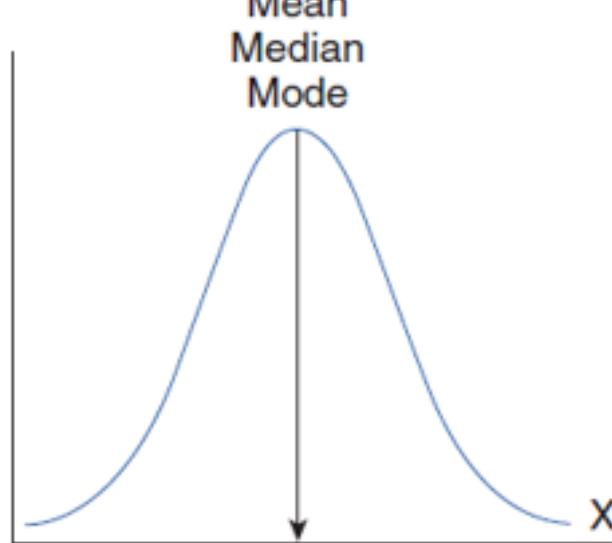


# Distributions can take on many different shapes

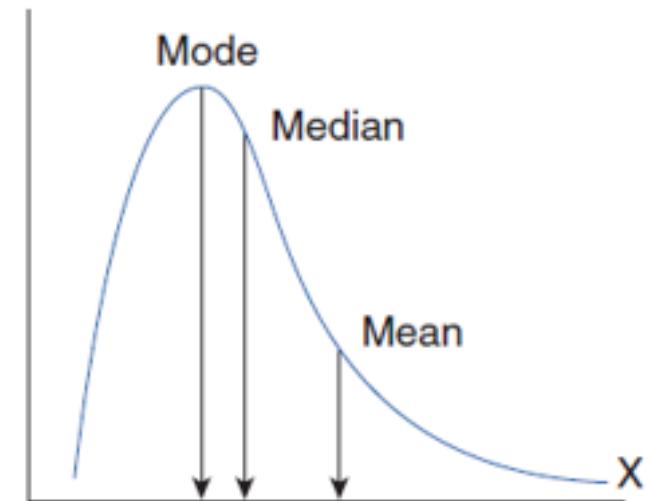
(a) Negatively skewed



(b) Normal (no skew)



(c) Positively skewed



The normal curve  
represents a perfectly  
symmetrical distribution

# Measures of Central Tendency

**Mean:** conventional average calculated by adding all values for all units and dividing by the number of units

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{1}{n}(x_1 + \cdots + x_n), \text{ with units } i = (1, 2, \dots, n)$$

- May give a distorted impression if there are outliers

**Median:** value that falls in the middle if we order all units by their value on the variable

**Mode:** most frequently occurring value across all units

# Measures of Dispersion

**Variance:** average of the squared differences between each observed value on a variable and its mean

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

- Why the square? To treat + and – differences alike

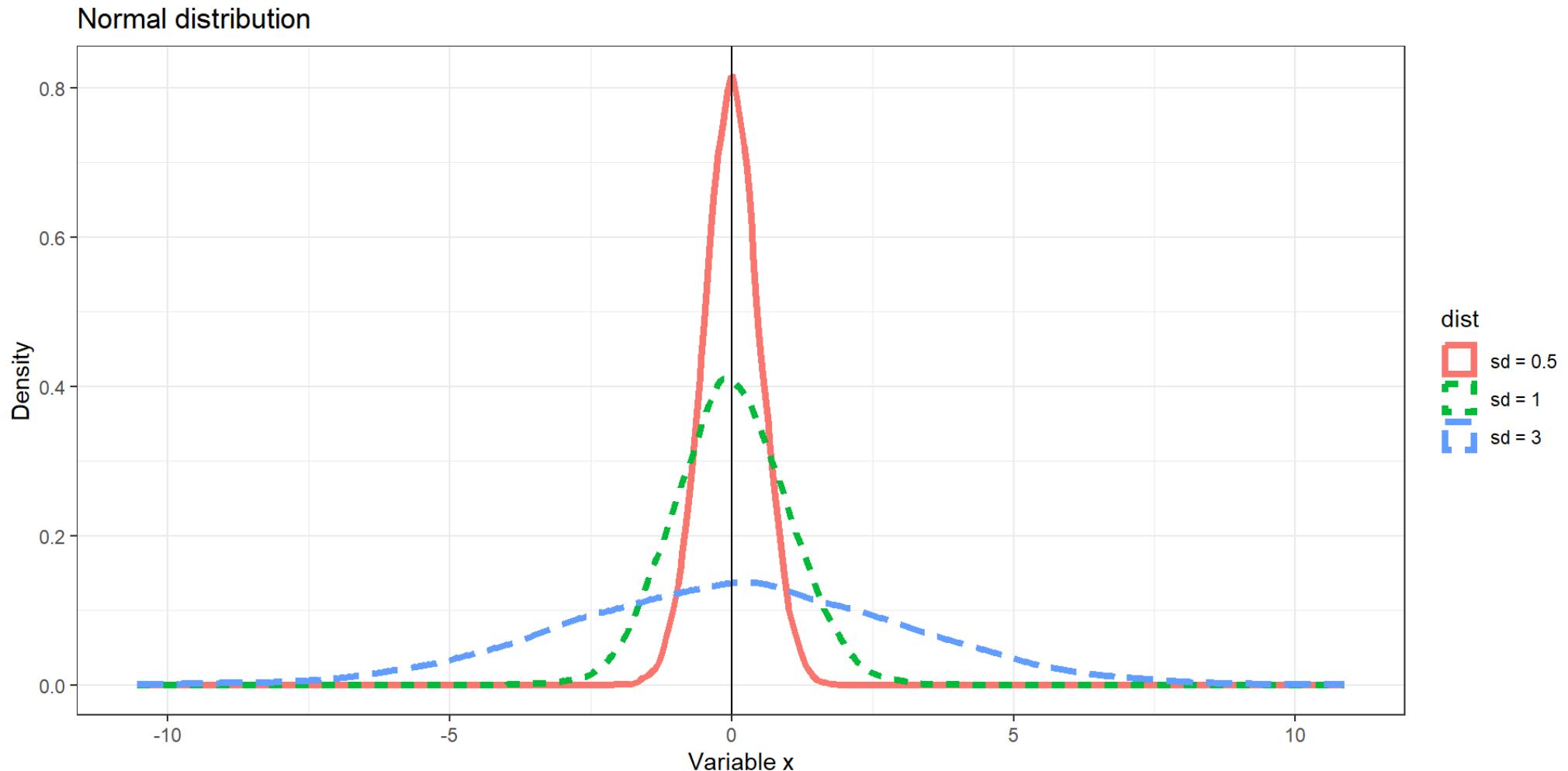
**Standard deviation:** average departure of the observed values on a variable from its mean

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$

- It is the square root of the variance; it “reverts” the square-taking in the variance calculation, to bring the statistic back to the original scale of the variable

# Measures of Dispersion

► expand for full code



# Notation

Typically, we use Roman letters for sample statistics and Greek letters for population statistics:

- Sample mean =  $\bar{x}$  population mean =  $\mu$
- Sample variance =  $s^2$ , population variance =  $\sigma^2$
- Sample standard error =  $s$ , population standard deviation =  $\sigma$

Recall: the sample is what we observe, the population is what we want to make inferences about

# Decribing relationships

# Describing relationships

Often, we are not just interested in the distribution of a single variable.<sup>1</sup>

Instead, we are interested in **relationships** between several variables. If there is a relationship between variable, knowing one variable can tell you something about the other variable.

- Age and height
- GDP and CO<sub>2</sub> emissions
- Education and income

1. Yeah, you could have spent the last 30 minutes on instagram!

# Hypothesis testing

A hypothesis is a theory-based statement about a relationship that we expect to observe

- E.g., girls achieve higher scores than boys on reading tests

For every hypothesis there is a corresponding null hypothesis about what we would expect if our theory is incorrect

- E.g., there is no association between Y and X in the population
- In our example: girls are not better readers than boys

# Covariance

**Covariance** refers to the idea that the pattern of variation for one variable corresponds to the pattern of variation for another variable: the two variables “vary together”

Statistically speaking, covariance is the multiplication of the deviations from the mean for the first variables and the deviations from the mean for the second variable:

$$cov_{x,y} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

- The covariance tells us the direction of an association: + or -
- It does not tell us about the strength of the association (reference to underlying distributions of variables is missing)

# Pearson's Correlation

For continuous data, we can calculate Pearson's correlation ( $\rho$ )

- $\rho$  measures strength & direction of association for linear trends
- $\rho$  rescales the covariance to the underlying distributions of the variables involved:

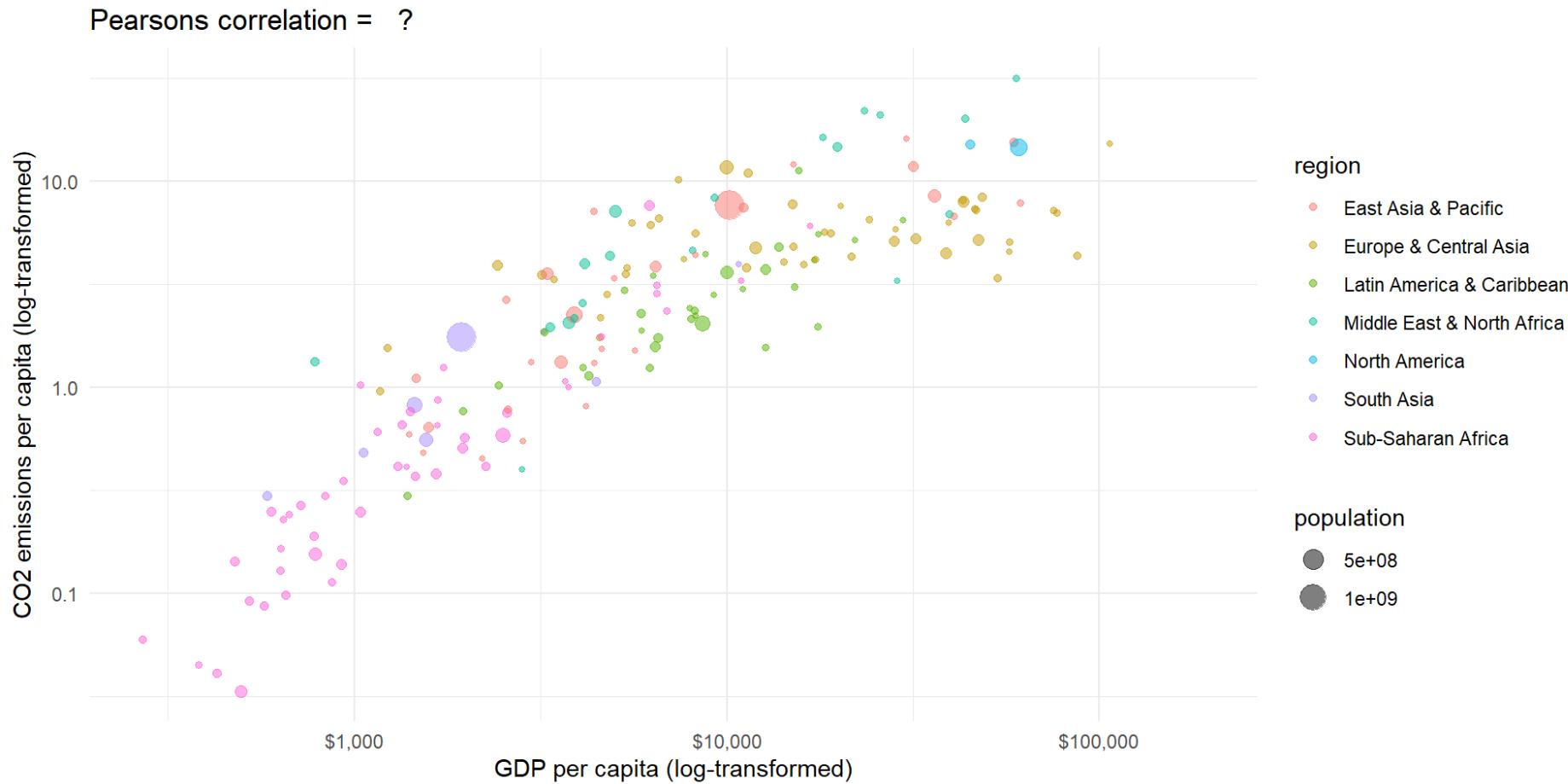
$$\rho = \frac{cov_{x,y}}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Dividing the covariance by the product of the standard deviations normalizes the covariance to a range from -1 to +1

- -1 = perfectly negative correlation (all points on decreasing line)
- +1 = perfectly positive correlation (all points on increasing line)
- 0 = no correlation (random cloud of points)
- Correlation is weaker closer to 0 and stronger closer to +/-1

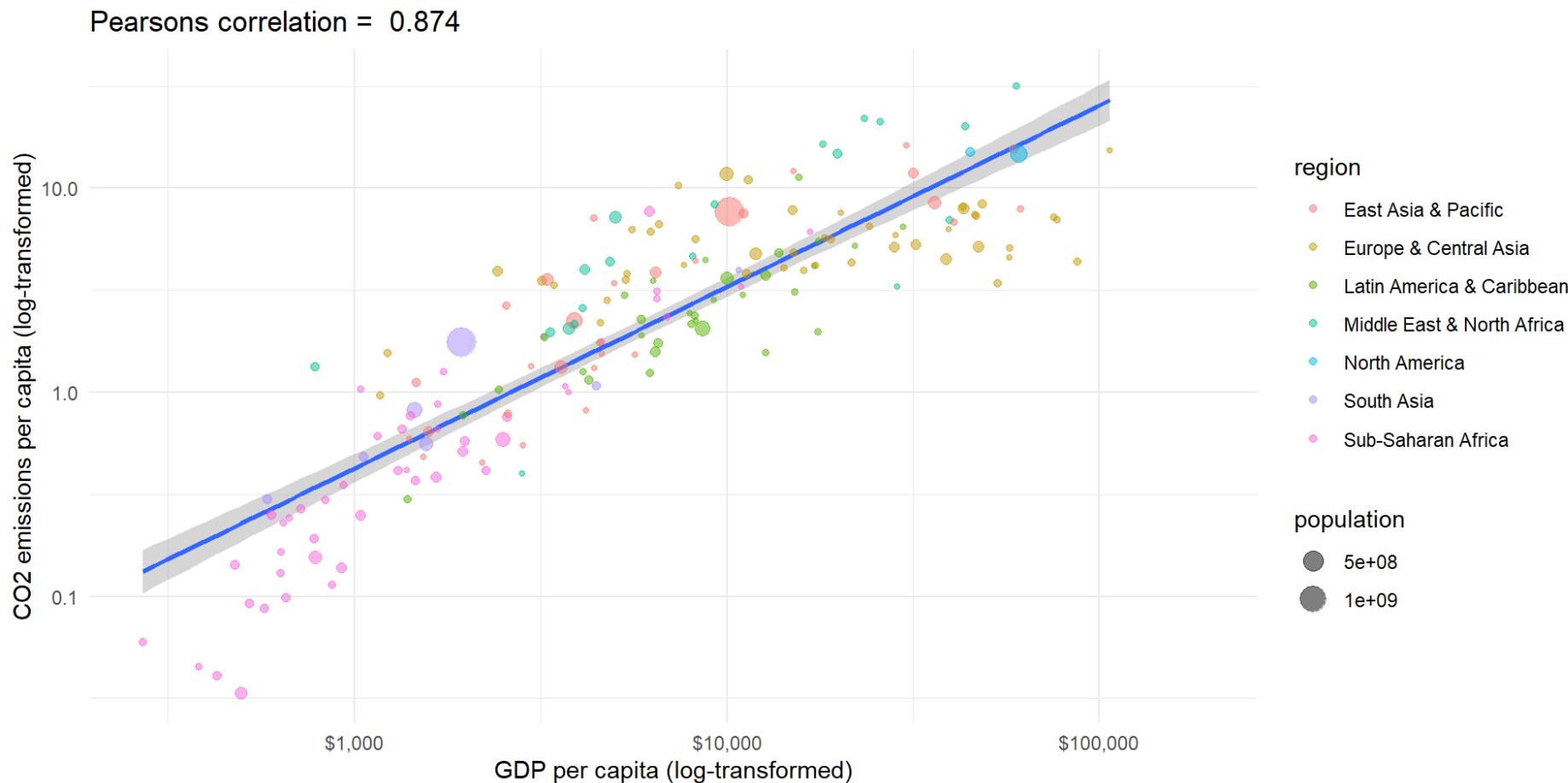
# Example I

## ► Code



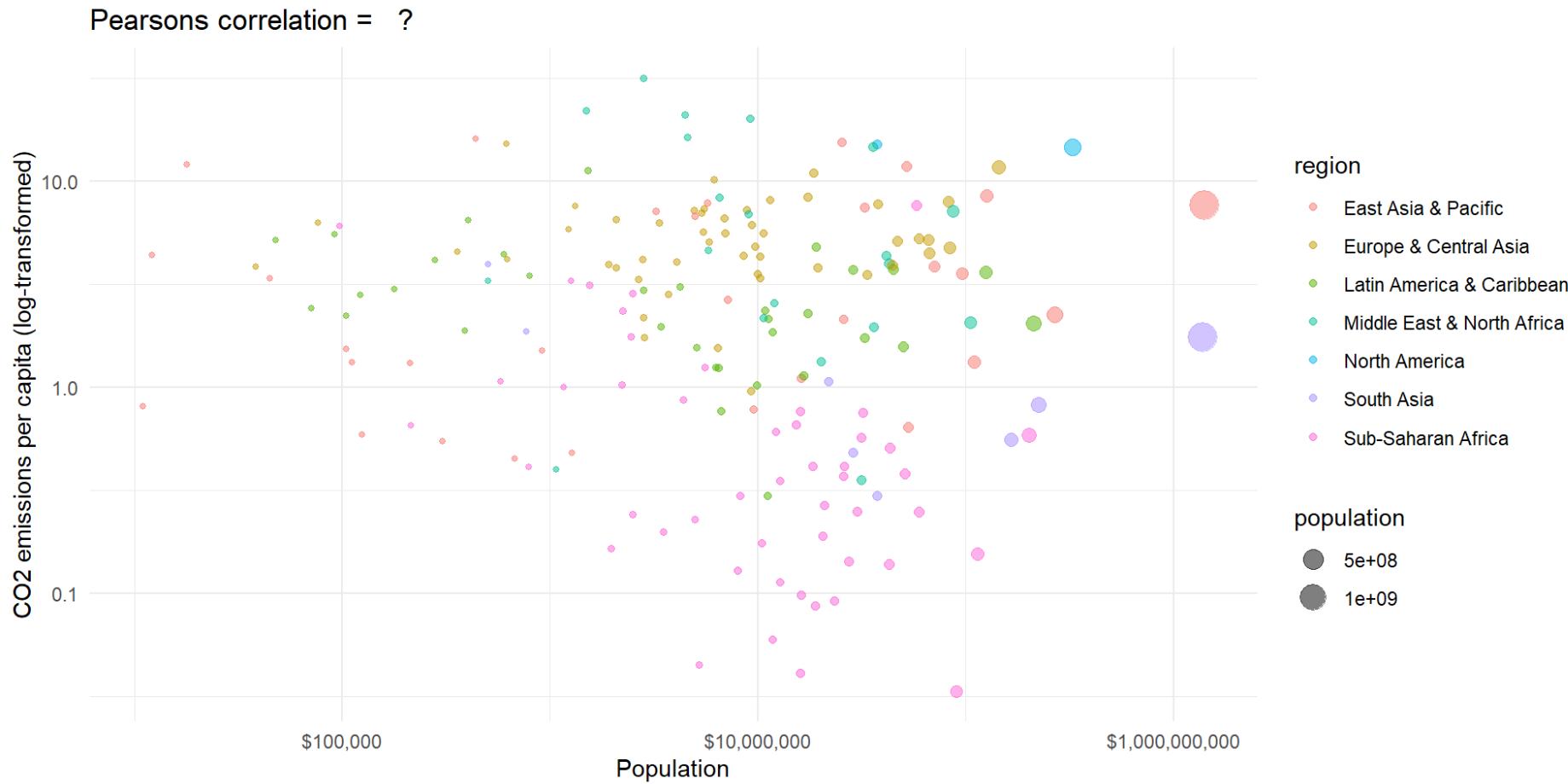
# Example I

## ► Code



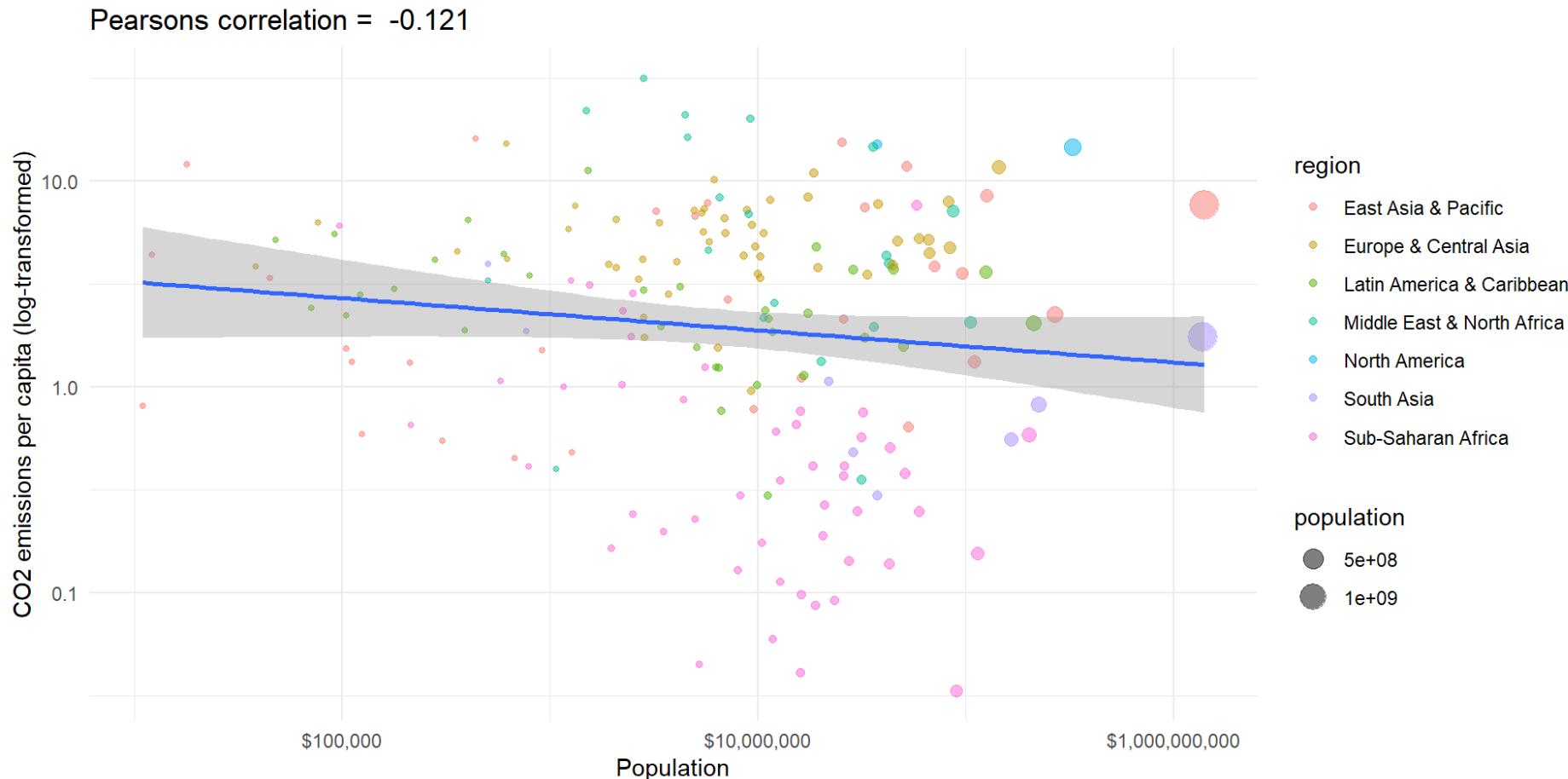
# Example II

## ► Code



# Example II

## ► Code



# The world is more complex

R Bootcamp\_slides - main - RStudio Source Editor

Wages

Filter

	exp	wks	bluecol	ind	south	smsa	married	sex	union	ed	black	lwage	
19	10	52	yes		1	no	no	yes	male	yes	12	no	6.09580
20	11	52	yes		1	no	no	no	male	yes	12	no	6.77878
21	12	46	yes		1	no	no	no	male	yes	12	no	6.86066
22	31	52	yes		0	no	yes	no	female	no	10	yes	6.15698
23	32	46	yes		0	no	yes	no	female	no	10	yes	6.23832
24	33	46	yes		0	no	yes	no	female	no	10	yes	6.30079
25	34	49	yes		0	no	yes	no	female	no	10	yes	6.35957
26	35	44	yes		0	no	yes	no	female	no	10	yes	6.46925
27	36	52	yes		0	no	yes	no	female	no	10	yes	6.56244
28	37	46	yes		0	no	yes	no	female	no	10	yes	6.62141
29	10	50	yes		0	no	no	yes	male	yes	16	no	6.43775
30	11	46	yes		0	no	no	yes	male	yes	16	no	6.62007
31	12	40	yes		0	no	no	yes	male	yes	16	no	6.63332
32	13	50	no		0	no	no	yes	male	no	16	no	6.98286
33	14	47	yes		0	no	yes	yes	male	no	16	no	7.04752
34	15	47	no		0	no	no	yes	male	no	16	no	7.31322
35	16	49	no		0	no	no	yes	male	no	16	no	7.29574
36	26	44	yes		1	no	yes	yes	male	no	12	no	6.90575
37	27	47	yes		1	no	yes	yes	male	no	12	no	6.90575

Showing 19 to 37 of 4,165 entries, 12 total columns

# The Linear Model

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_k x_{ki} + \epsilon_i$$

- $y_i$ :  $i$  observation ( $i = 1, \dots, n$ ) of dependent variable  $y$ ;
- $X_{ji}$ :  $i$  observation of  $j$  explanatory variable  $x$  ( $j = 1, \dots, k$ ).
- $\beta_j$ : parameters to be estimated.
- $\epsilon_i$ : error term. It contains all factors affecting  $y_i$ , but not contained in  $x_{ji}$ .

# The OLS estimator

The Omitted Least Squares (OLS) estimator is one way to estimate the parameter  $\beta$  in our population model

To distinguish between estimates and actual parameters to be estimated, the following notation is generally used:

- The estimate for  $\beta$  is indicated as  $\hat{\beta}$
- The estimated error term, also called residual is indicated as  $\hat{\epsilon}$
- The predicted value of the dependent variable  $y$  is indicated as  $\hat{y}$

# The OLS estimator

The OLS method chooses the estimates of  $\beta$  to minimise the sum of squared residuals (SSR)

$$\min_{\beta_0, \dots, \beta_k} \left[ \sum_{i=1}^N (Y_i - \hat{Y}_i)^2 \right] \quad \text{or} \quad \min_{\beta_0, \dots, \beta_k} [\hat{\epsilon}^\top \hat{\epsilon}]$$

The solution is the OLS estimator:

$$\hat{\beta}_{OLS} = \frac{\sum_{i=1}^N (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^N (X_i - \bar{X})^2}$$

or in matrix notation:

$$\hat{\beta}_{OLS} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

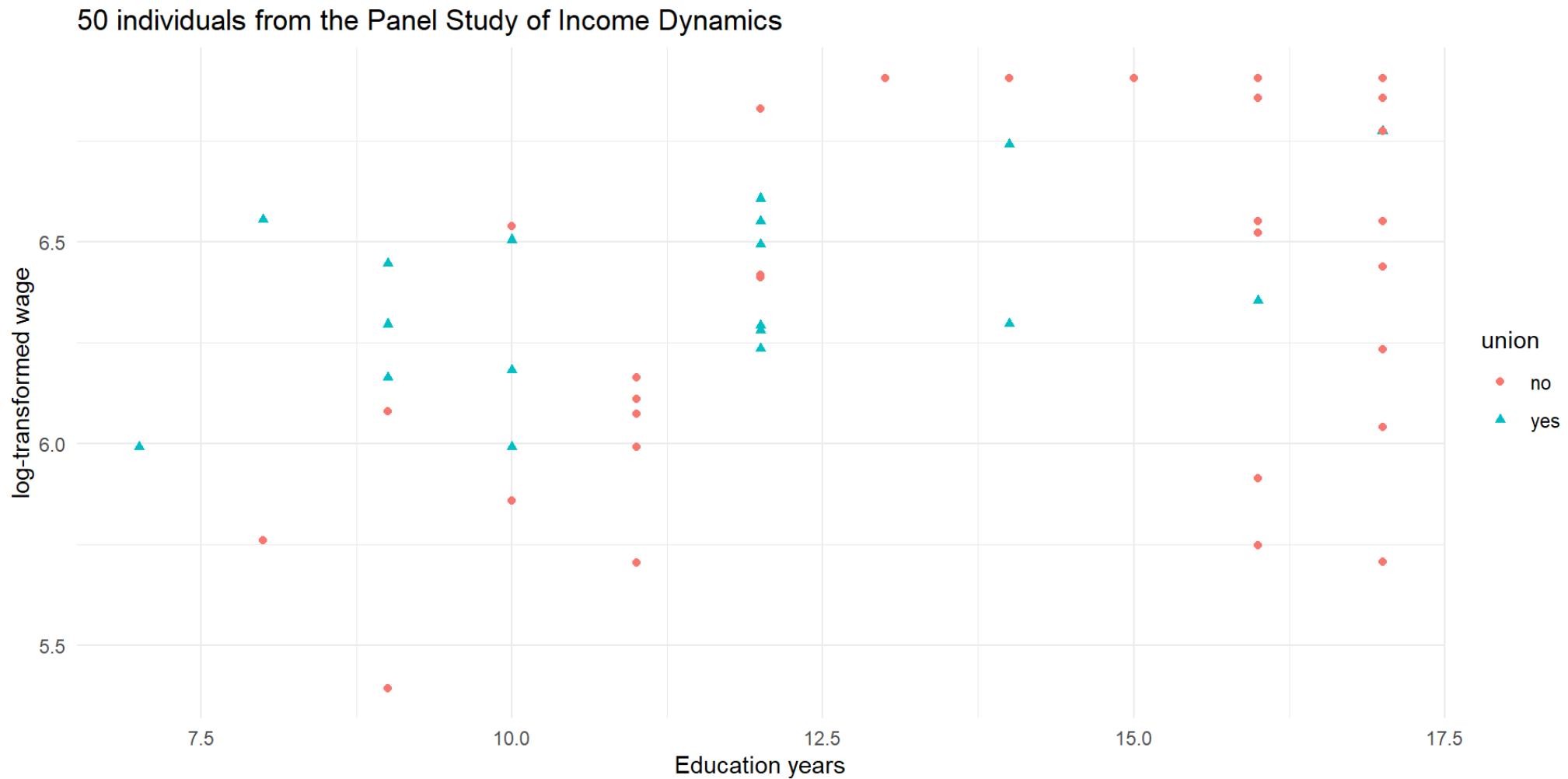
# An example: wage

$$wage_i = \beta_0 + \beta_1 education_i + \beta_2 union_i + \epsilon_i$$

- $y_i$ : log-transformed wage of individual  $i$ ;
- $X_{1i}$ : education years of individual  $i$ .
- $X_{2i}$ : whether or not individual  $i$  is union member.
- $\epsilon_i$ : error term. It contains all factors affecting the wage, but not contained in education years and union membership (such as age, gender, subject, quantitative methods skills, and lots more).

# An example: wage

► Code



# An example: wage

## The OLS estimator

```
1 mod1.lm <- lm(lwage ~ ed + union, data = wages_cs.df)
2 summary(mod1.lm)
```

Call:

```
lm(formula = lwage ~ ed + union, data = wages_cs.df)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.86123	-0.19264	0.01753	0.22068	0.59062

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	5.49218	0.25015	21.956	< 2e-16 ***
ed	0.06330	0.01746	3.626	0.000707 ***
unionyes	0.16195	0.11259	1.438	0.156938

---

Signif. codes: 0 '\*\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3526 on 47 degrees of freedom

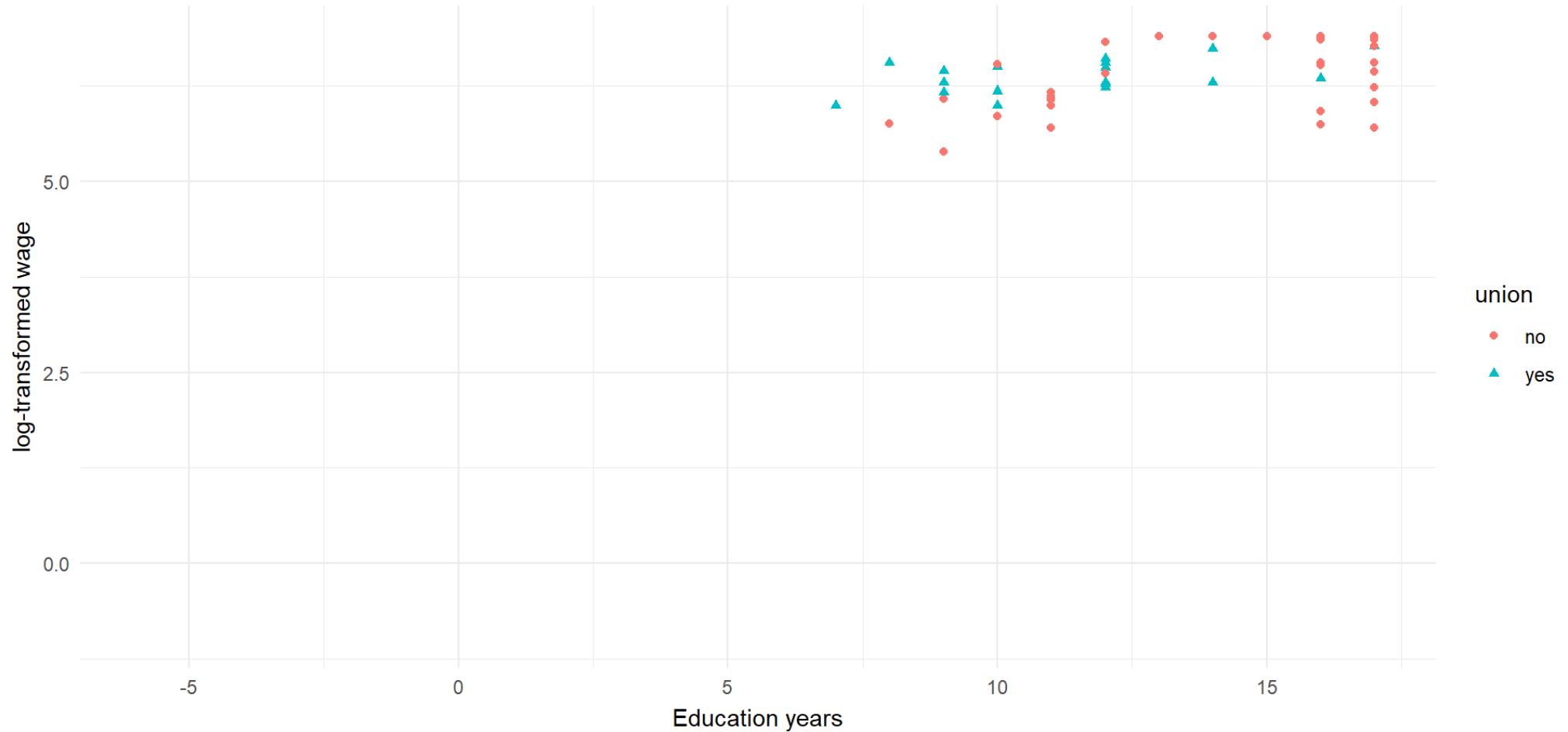
Multiple R-squared: 0.2186, Adjusted R-squared: 0.1853

F-statistic: 6.574 on 2 and 47 DF, p-value: 0.003039

# An example: wage

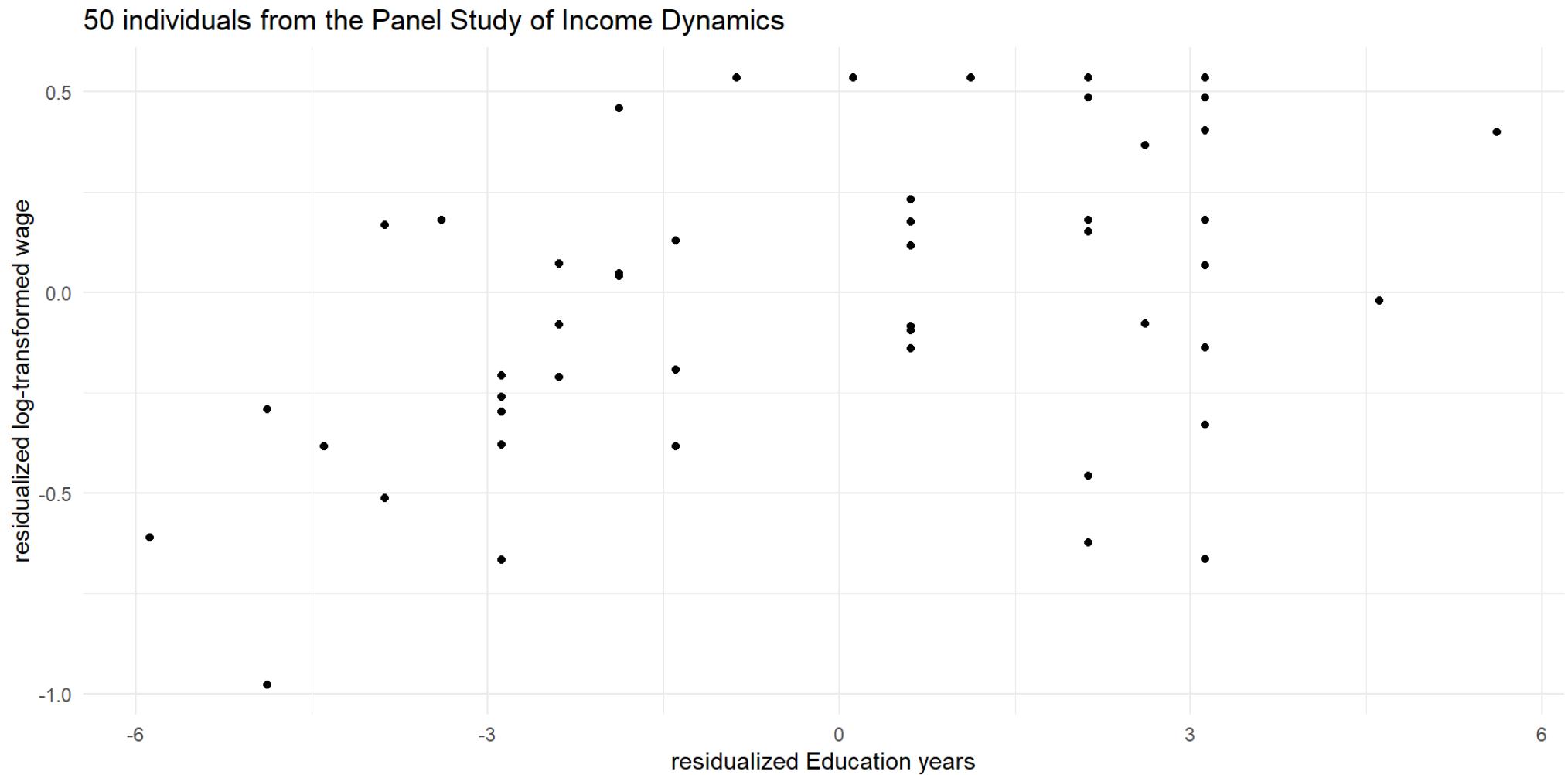
► Code

50 individuals from the Panel Study of Income Dynamics



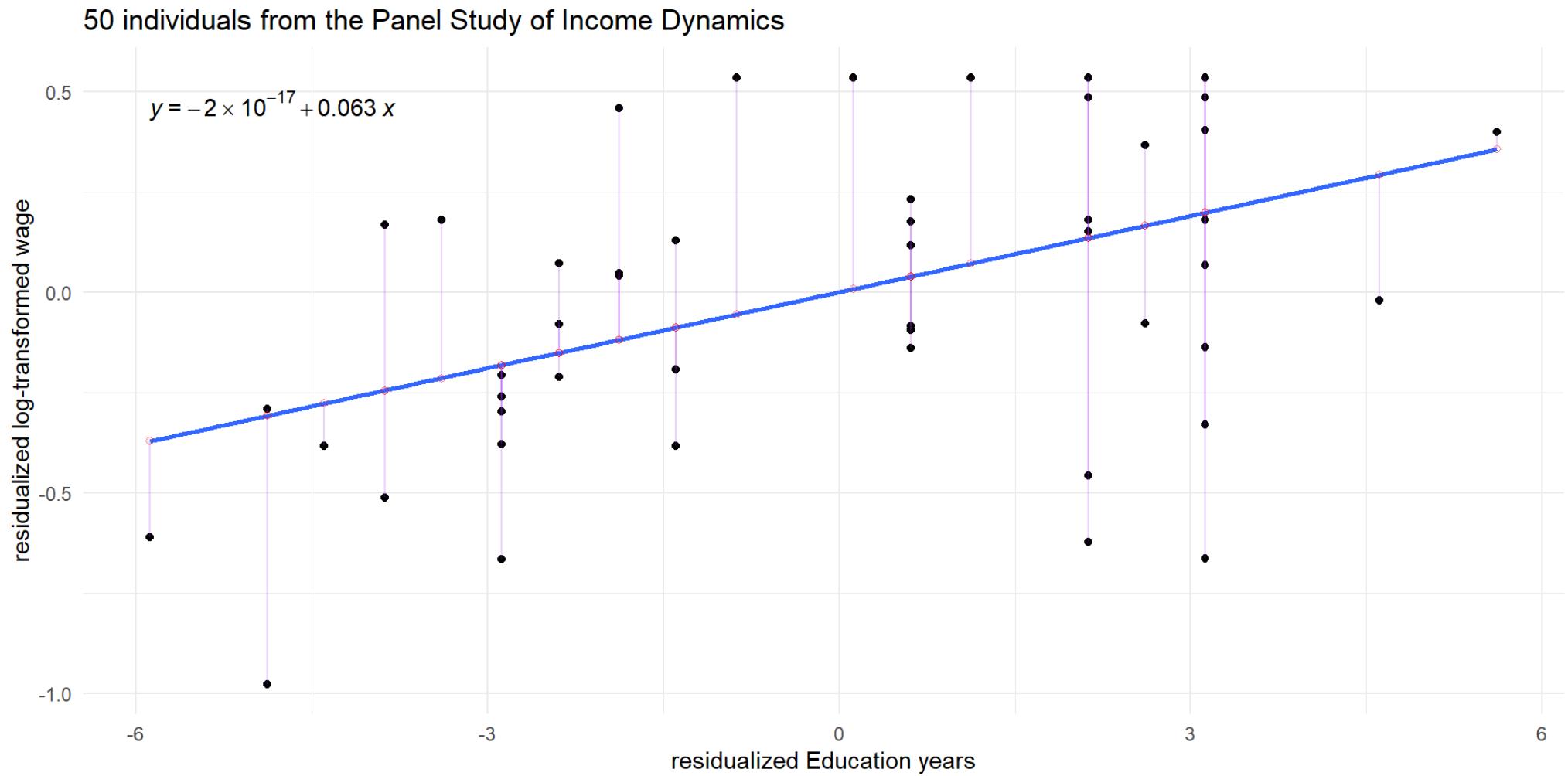
# An example: wage

► Code



# An example: wage

► Code



# Statistical Software

# UCL Download centre

<https://www.ucl.ac.uk/isd/services/software-hardware/software-for-students>

# Statistical software

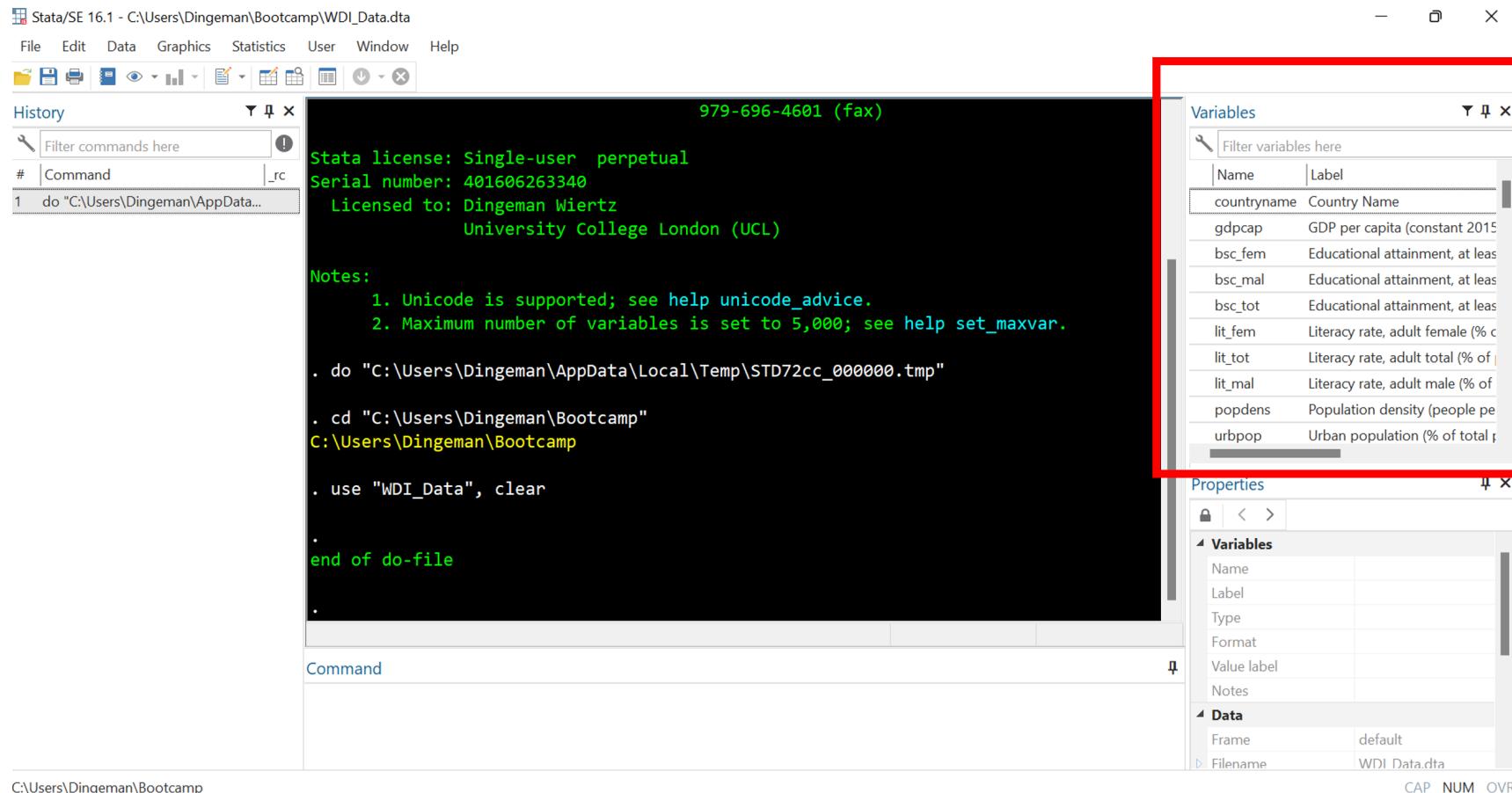
**Stata & R** are powerful software packages that allows you to do:

- Data management and manipulation
- Data visualization
- Statistical analysis
- (Writing papers and presentations)

Writing Stata syntax files and R script facilitates  
**reproducibility**

# Stata's Interface: Variables Window

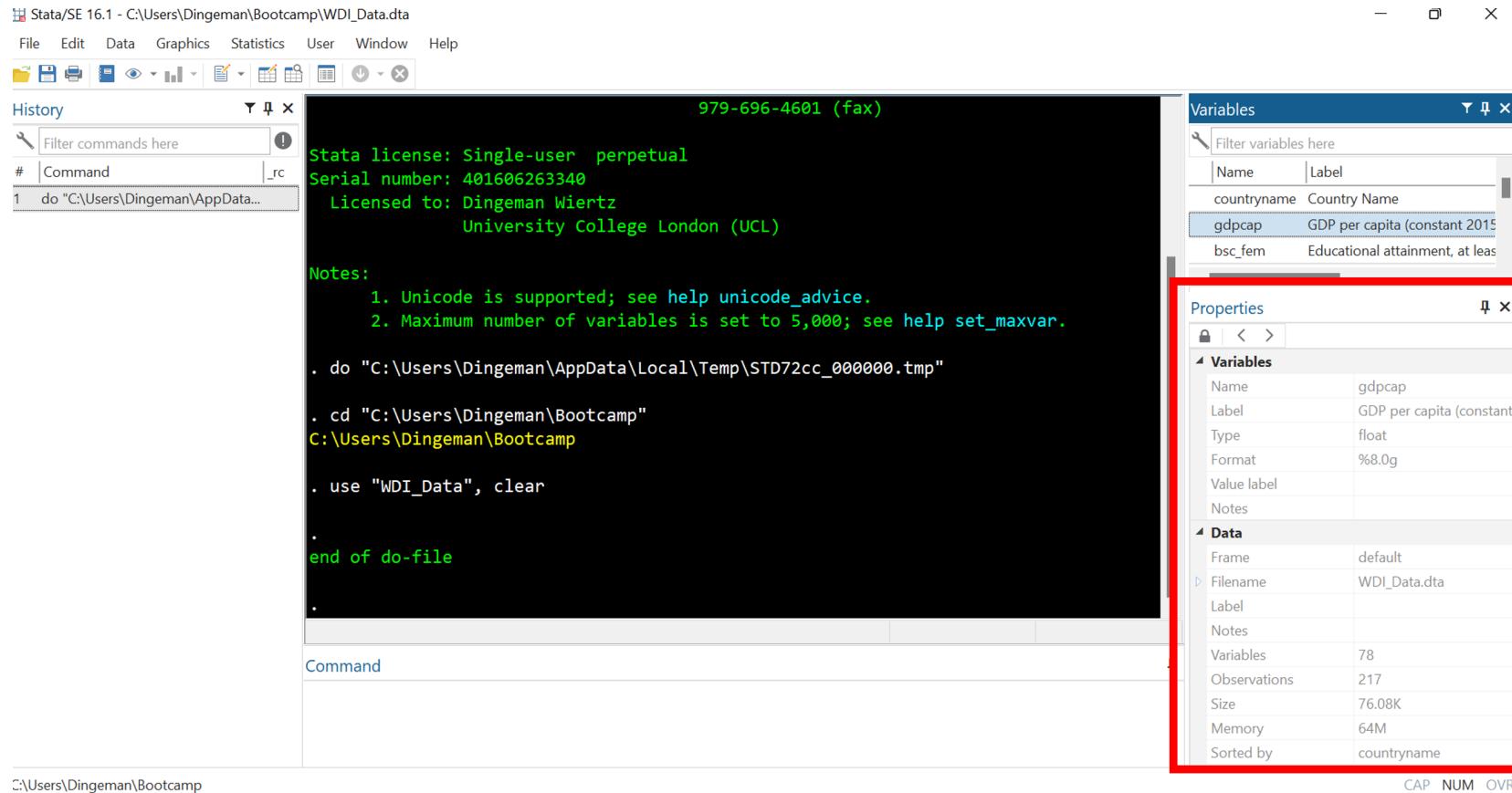
The variables window displays all variables in your dataset



- Single click on variable names to see details in properties window
- Double click to make variables appear in command window

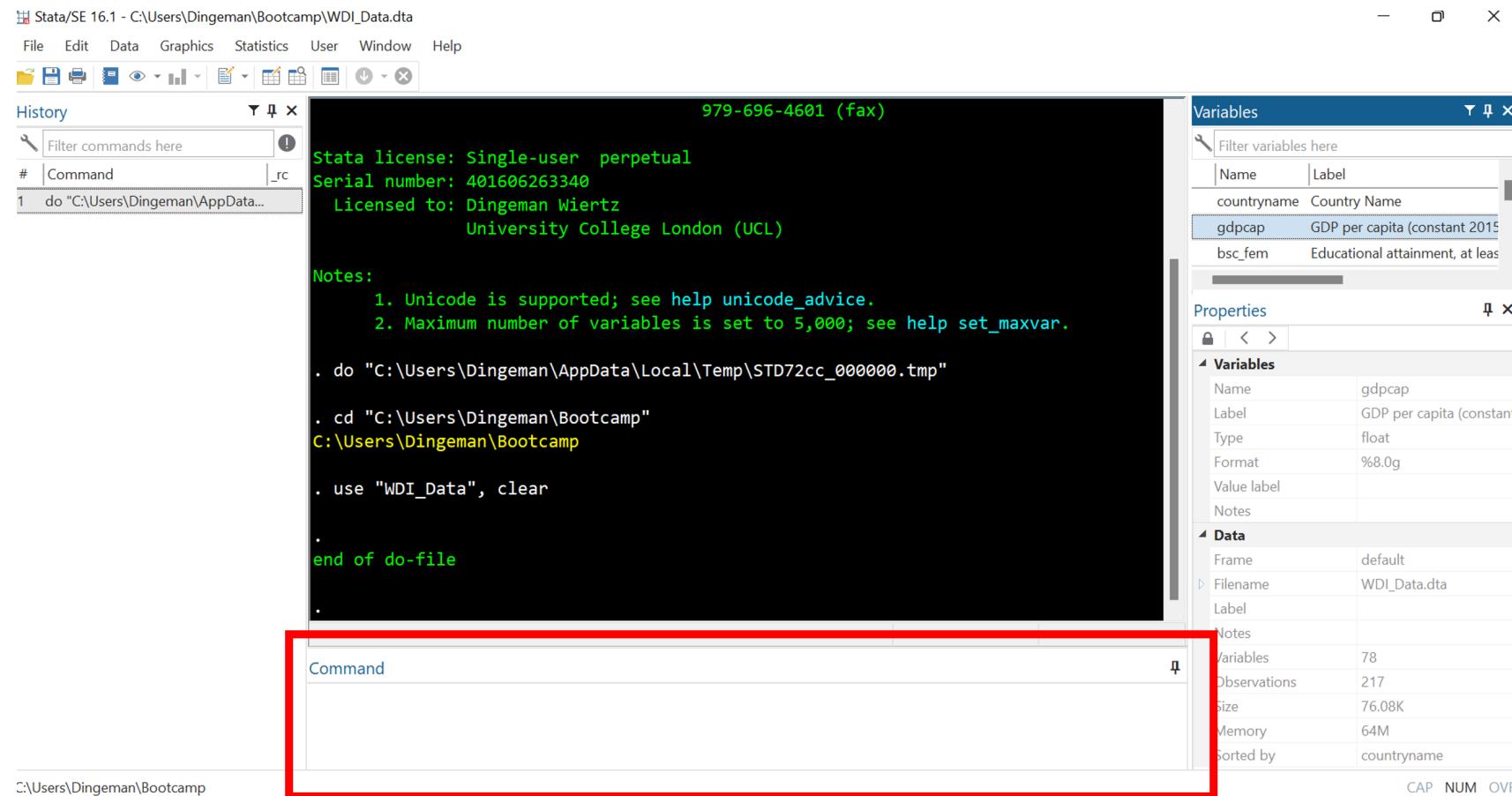
# Stata's Interface: Properties Window

The variables window displays all variables in your dataset



The properties window displays details about selected variables as well as the entire dataset (e.g., number of observations, sort order)

# Stata's Interface: Command Window



The command window is for entering and executing commands

- But it is better to use do-files (same applies to drop-down menus)

# Stata's Interface: Results Window

The screenshot shows the Stata SE 16.1 interface with the following components:

- Title Bar:** "Stata/SE 16.1 - C:\Users\Dingeman\Bootcamp\WDI\_Data.dta".
- Menu Bar:** File, Edit, Data, Graphics, Statistics, User, Window, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Print, and various data analysis tools.
- History:** Shows a history of commands entered: 1. do "C:\Users\Dingeman\AppData...", 2. describe, 3. codebook countryname, 4. sum gdpcap, detail.
- Results Window (highlighted with a red border):**
  - Output title: "Sint Maarten (Dutch part)"
  - Warning message: "warning: variable has embedded blanks"
  - Command: ". sum gdpcap, detail"
  - Summary statistics for "GDP per capita (constant 2015 US\$)":

	Percentiles	Smallest	Obs	Sum of Wgt.
1%	418.7217	278.2026		
5%	649.7603	401.3927		
10%	945.5074	418.7217		
25%	2462.564	446.8542		
50%	6590.069		Mean	16782.43
		Largest	Std. Dev.	24236.55
75%	20305.29	96105.3	Variance	5.87e+08
90%	46135.08	107036.2	Skewness	2.901783
95%	60687.23	108570	Kurtosis	14.89096
99%	107036.2	181709.3		
- Variables Window:** Shows a list of variables with their labels:

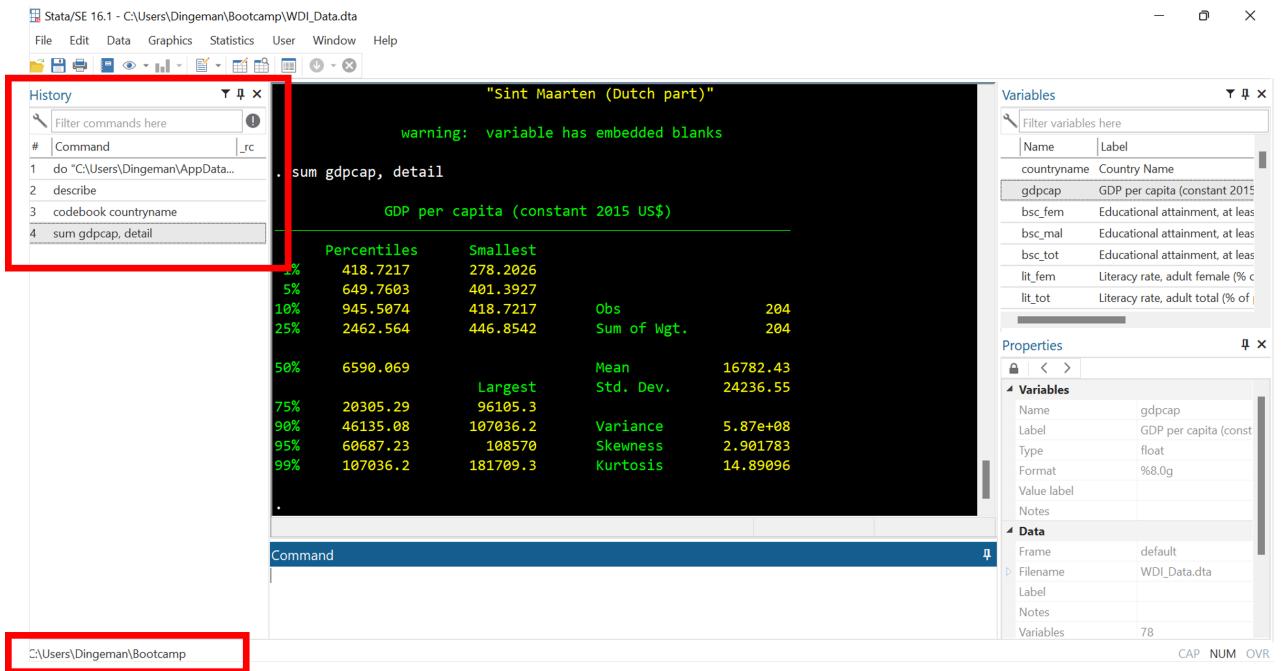
Name	Label
countryname	Country Name
gdpcap	GDP per capita (constant 2015
bsc_fem	Educational attainment, at leas
bsc_mal	Educational attainment, at leas
bsc_tot	Educational attainment, at leas
lit_fem	Literacy rate, adult female (% c
lit_tot	Literacy rate, adult total (% of
- Properties Window:** Shows properties for the selected variable "gdpcap":

Name	Label
Name	gdpcap
Label	GDP per capita (const
Type	float
Format	%8.0g
Value label	
Notes	
- Data Window:** Shows data frame information:

Frame	default
Filename	WDI_Data.dta
Label	
Notes	
Variables	78
- Command Window:** Shows the command ". sum gdpcap, detail".
- Status Bar:** Shows the path "C:\Users\Dingeman\Bootcamp" and status indicators CAP NUM OVR.

The results window displays all output of your commands

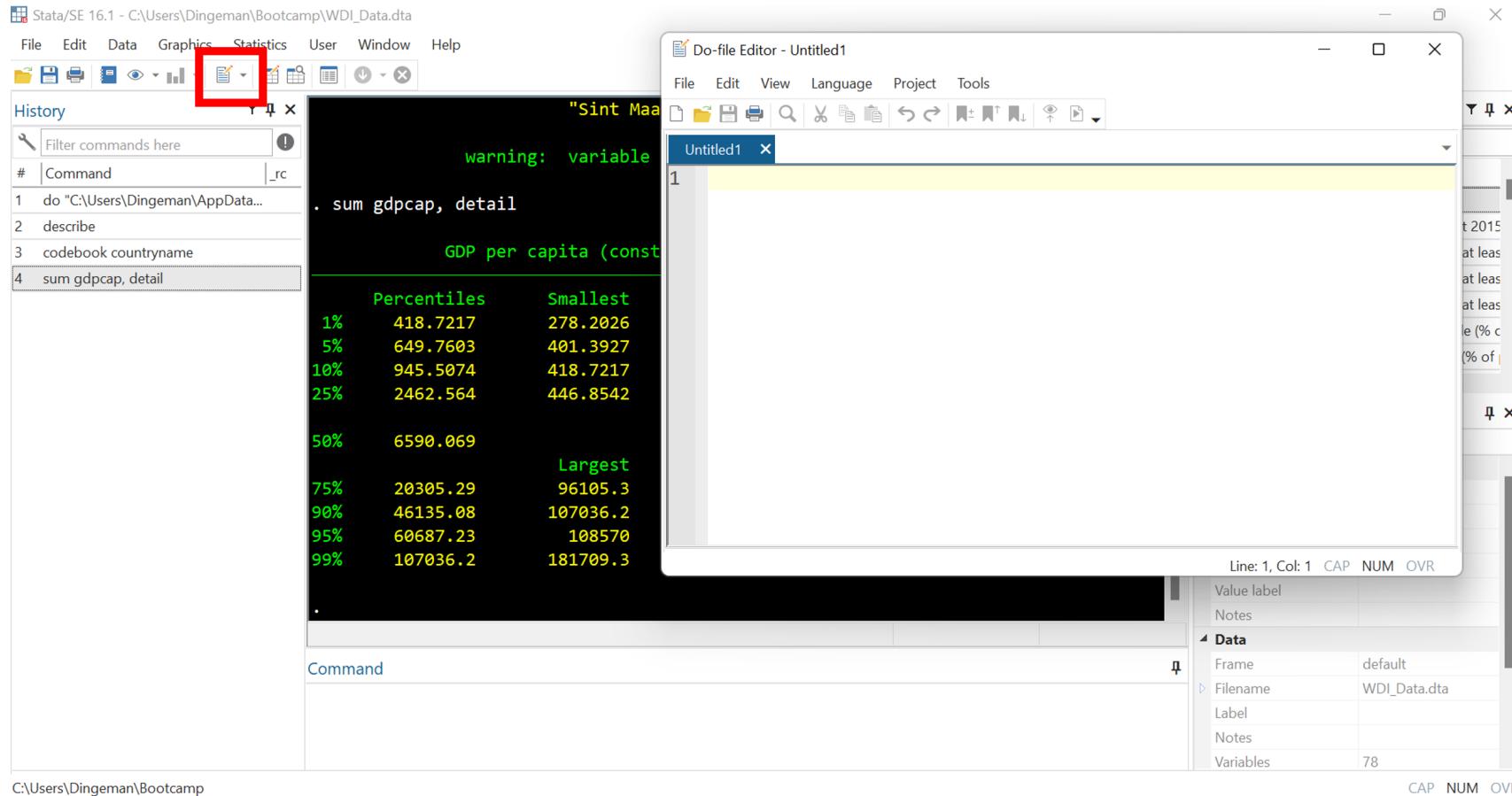
# Stata's Interface: Command History and Current Working Directory



The command history window lists previously run commands

- At the bottom you can see the current working directory: the folder where any files will be loaded from and saved to

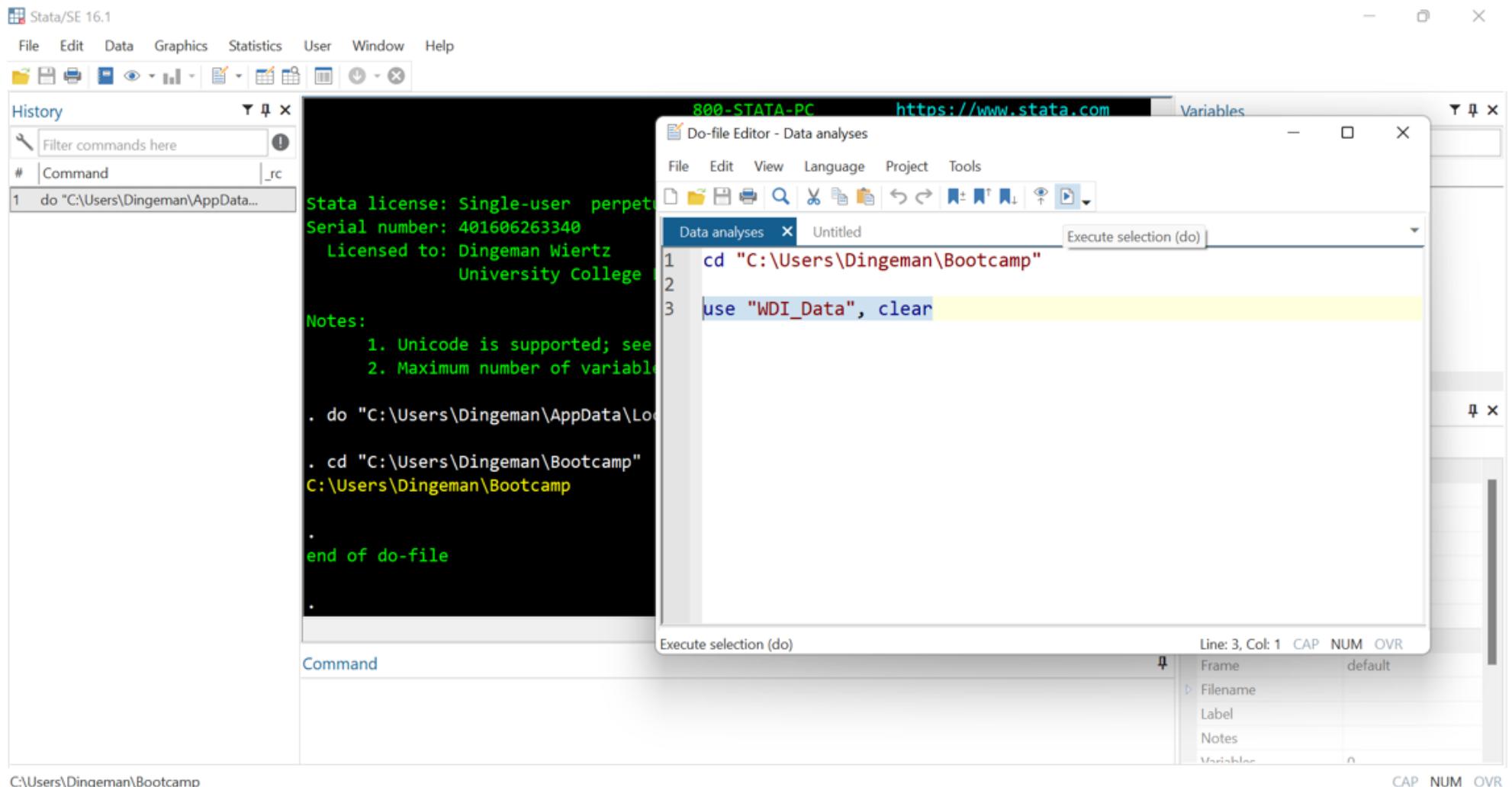
# Stata's Interface: Opening a Do File



Do files are text files where you can store commands for reuse

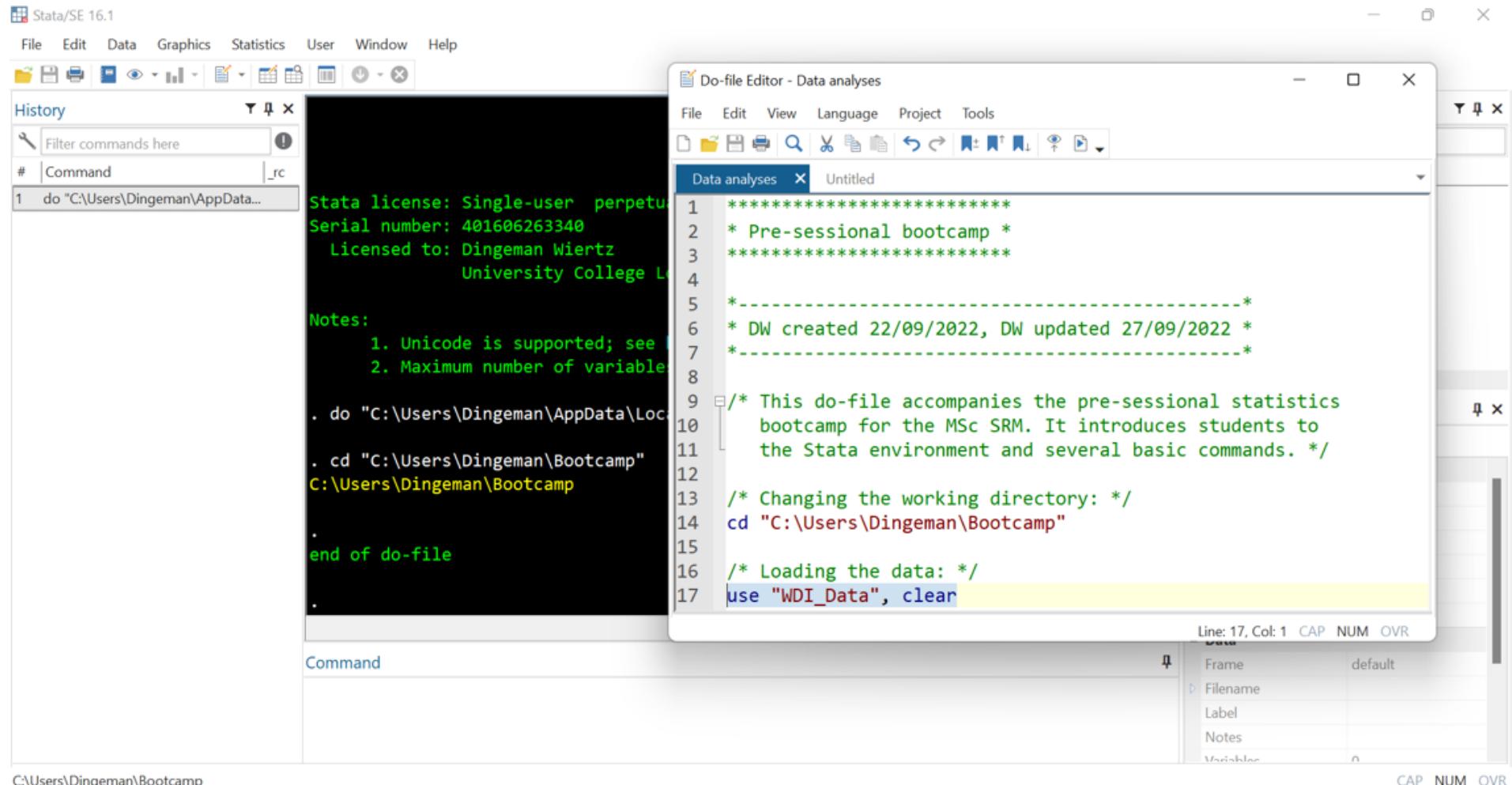
- Huge payoffs for reproducibility, debugging, adapting commands

# Running Commands from a Do File



After entering a command, you select it, and then click the “execute” button or press “Ctrl+D”

# Do File Dos and Don'ts



The screenshot shows the Stata SE 16.1 interface. On the left, the Command window displays the output of a license command:

```
Stata license: Single-user perpetual  
Serial number: 401606263340  
Licensed to: Dingeman Wiertz  
University College L
```

Below this, the Notes section lists:

1. Unicode is supported; see [http://www.stata.com/unicode](#)
2. Maximum number of variables

The History window shows the commands run:

```
1 do "C:\Users\...AppData\Loc...
```

The Do-file Editor window is open, showing a script named "Data analyses". The code includes comments and annotations:\*\*\*\*\*  
\* Pre-sessional bootcamp \*  
\*\*\*\*\*  
\*-----\*  
\* DW created 22/09/2022, DW updated 27/09/2022 \*  
\*-----\*  
/\* This do-file accompanies the pre-sessional statistics  
bootcamp for the MSc SRM. It introduces students to  
the Stata environment and several basic commands. \*/  
/\* Changing the working directory: \*/  
cd "C:\Users\...Bootcamp"  
  
/\* Loading the data: \*/  
use "WDI\_Data", clear

The status bar at the bottom indicates the current line and column (Line: 17, Col: 1), and the keyboard mode (CAP NUM OVR).

1. Use annotations to facilitate replicability (incl. for future self!):
  - Use \* for single-line comments and /\* \*/ for multiple lines

Stata/SE 16.1 - C:\Users\Di... Bootcamp\WDI\_Data.dta

File Edit Data Graphics Statistics User Window Help

History Filter commands here

```
# | Command _rc
1 do "C:\Users\Di... AppData\Roaming\Stata\16.1\dofiles\WDI_D...
```

stats	totpop	popdens	ur...
mean	3.54e+07	443.7099	61...
p50	6661478	92.7822	62...
p10	84589	15.94115	2...
p90	6.68e+07	511.8337	9...

stats	povg~215	povg~685	co2...
mean	1.386667	9.015	.43...
p50	.2	2.7	.33...
p10	0	.15	.14...
p90	2.1	24.25	.93...

```
end of do-file
```

Command

Do-file Editor - Data analyses

Data analyses Untitled

```
12
13 /* Changing the working directory: */
14 cd "C:\Users\Di... Bootcamp"
15
16 /* Loading the data: */
17 use "WDI_Data", clear

18 ****
19 * 1. Summary statistics *
20 ****

21
22 /* 1a. Using the summarize command: */
23 summarize totpop popdens urbpop ///
24 gdpcap gini top10p ///
25 povrat povgap215 povgap685 ///
26 co2intens ghgchange natresource

27
28 /* 1b. Using the tabstat command: */
29 tabstat totpop popdens urbpop ///
30 gdpcap gini top10p ///
31 povrat povgap215 povgap685 ///
32 co2intens ghgchange natresource, ///
33 statistics(mean median p10 p90)

34
```

Line: 34, Col: 52 CAP NUM OVR

Variables: 70 CAP NUM OVR

2. Break down code into clearly labelled sections / subsections
3. Use tab indentations to making things easy to read

The screenshot shows two windows side-by-side. On the left is the Stata SE 16.1 interface. The top menu bar includes File, Edit, Data, Graphics, Statistics, User, Window, and Help. Below the menu is a toolbar with various icons. A 'History' window is open, showing a command history with one entry: 'do "C:\Users\...AppData..."'. The main workspace displays two tables of statistical results from the 'stats' command. The first table has columns 'stats', 'totpop', 'popdens', and 'urban'. The second table has columns 'stats', 'povg~215', 'povg~685', and 'co2intens'. Both tables include rows for 'mean', 'p50', 'p10', and 'p90'. Below these tables, the text 'end of do-file' is visible. At the bottom of the Stata window, there's a 'Command' field.

The right window is titled 'Do-file Editor - Data analyses'. It has a menu bar with File, Edit, View, Language, Project, and Tools. The main area contains a script for a 'do' file. The script starts by changing the working directory to 'C:\Users\...Bootcamp' and loading a dataset named 'WDI\_Data'. It then performs two sections of analysis: 1. Summary statistics, which uses the 'summarize' command to produce statistics for variables like 'totpop', 'popdens', 'urbpop', 'gdpcap', 'gini', 'top10p', 'povrat', 'povgap215', 'povgap685', 'co2intens', 'ghgchange', and 'natresource'. 2. Statistics, which uses the 'tabstat' command to produce statistics for the same variables. The 'statistics' command is highlighted with a yellow background.

#### 4. Don't put too much information on a single line

- Use /// to continue your command on the next line and
- write “top-to-bottom” instead of “left-to-right”

# R & R Studio Interface

The screenshot displays the RStudio interface with the following components:

- Code Editor:** Shows an R script named `Example-script.R` containing code to load packages (WDI, ggplot2), get WDI data for all countries from 2019, drop aggregates, save the data, and create a ggplot of GDP and CO<sub>2</sub> per capita.
- Environment Browser:** Displays the global environment with objects like `d1`, `d2`, `d3`, `d4`, `p`, `p1`, `p12`, `p13`, `wd.df`, and various values such as `a`, `b`, `cobs`, `cor`, `cov`, `sd_co2`, and `sd_adp`.
- File Browser:** Shows the project directory structure under `C:/work/Lehre/Bootcamp_slides`, including files like `_quarto.yml`, `.gitignore`, `.Rhistory`, `Bootcamp_slides.qmd`, `Bootcamp_slides.Rproj`, `docs`, `figs`, `README.md`, `WDI_short.RData`, and `Example-script.R`.
- Console:** Displays R session history with commands like `[1] 8`, `> 7 + 2`, `[1] 9`, `> 12 * 8`, `[1] 96`, `> paste("hallo", "world")`, `[1] "hallo world"`, `> a = 7; b = 2`, `> a + b`, `[1] 9`, and `>`.

# **Structure of Commands in Stata**

# General Stata Command Syntax

**summarize** [*varlist*] [*if*] [*in*] [*weight*] [, *options*]

Stata commands mirror everyday commands in their structure:

- They often start with a verb: “Bring me...”
- They then list an object: “... a pint of milk...”
- They may add a condition: “... if it is still before noon...”
- They may specify further details after the comma: “, quickly please” or “, I want semi-skimmed”

**summarize** [*varlist*] [*if*] [*in*] [*weight*] [, *options*]

In nearly all cases, Stata syntax consists of four parts:

- **Command:** What action do you want to see performed?
- **Names of variables, files, objects:** On what objects is the command to be performed (“varlist”)
- **Qualifier(s) on observations:** Which observations are to be taken into account (and how)? (“if”, “in”, “weight”)
- **Options:** What special things should be done in the execution?

# “help [command]” is your friend

The screenshot shows a "Viewer - help summarize" window with the following content:

File Edit History Help

← → C Print

help summarize

Dialog

[R] **summarize** — Summary statistics  
(View complete PDF manual entry)

Syntax

`summarize [varlist] [if] [in] [weight] [, options]`

<i>options</i>	Description
<u>Main</u>	
<u>detail</u>	display additional statistics
<u>meanonly</u>	suppress the display; calculate only the mean; programmer's option
<u>format</u>	use variable's display format
<u>separator(#)</u>	draw separator line after every # variables; default is <code>separator(5)</code>
<u>display_options</u>	control spacing, line width, and base and empty cells

CAP NUM INS

```
1 help summarize
```

# **Structure of Commands in R**

# General R Workflow

```
1 object_name <- value
2
3 # Example
4 a <- 3
5 b <- 4
6 c <- a + b
7 c
```

- Assign a value to an object

# General R Workflow

```
1 object_name <- value
2
3 # Example
4 a <- 3
5 b <- 4
6 c <- a + b
7 c
```

- Define the objects a and b

# General R Workflow

```
1 object_name <- value  
2  
3 # Example  
4 a <- 3  
5 b <- 4  
6 c <- a + b  
7 c
```

- Perform an operations with them

# General R Workflow

```
1 object_name <- value  
2  
3 # Example  
4 a <- 3  
5 b <- 4  
6 c <- a + b  
7 c
```

[1] 7

- Return the results

# General R Syntax

```
1 function_name(arg1 = val1[which()], arg2 = val2[which()], option1 = val3, .
```

- **function\_name:** What action do you want to see performed?
- **args, files, objects:** On what objects is the command to be performed
- **Qualifier(s) on observations:** Which observations are to be taken into account (and how)? (“which”)
- **options:** What special things should be done in the execution?

# General R Syntax

Produce a sequence between 0 and 10 with 20 values

```
1 y <- seq(0, 10, length.out = 20)
2 y
```

```
[1] 0.0000000 0.5263158 1.0526316 1.5789474 2.1052632 2.6315789
[7] 3.1578947 3.6842105 4.2105263 4.7368421 5.2631579 5.7894737
[13] 6.3157895 6.8421053 7.3684211 7.8947368 8.4210526 8.9473684
[19] 9.4736842 10.0000000
```

Calculate the mean

```
1 mean(y)
```

```
[1] 5
```

Calculate the mean of all values above 5

```
1 mean(y[which(y >= 5)])
```

```
[1] 7.631579
```

# “?[command]” is your friend

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Source Editor:** Displays code from `Bootcamp_slides.qmd` and `Example-script.R`. The code includes sections for General R Syntax and a specific example involving `seq`.
- Environment Tab:** Shows the Global Environment with four data frames: `d1`, `d2`, `d3`, and `d4`, each containing 10000 observations of 2 variables.
- Help Tab:** The `seq` function documentation is open. It includes:
  - Description:** Generates regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.
  - Usage:** Shows the syntax for `seq`, `seq.int`, `seq_along`, and `seq_len`.
  - Arguments:** Describes the arguments: `from` and `to` (starting and maximal end values), `by` (increment), `length.out` (desired length), and `along.with` (length from another argument).
- Console:** Shows R session history with commands like `> 7 + 2`, `[1] 9`, `> 12 * 8`, `[1] 96`, `> paste("hallo", "world")`, `[1] "hallo world"`, `> a = 7; b = 2`, `> a + b`, `[1] 9`, and `> ?seq`.

1 ?seq

# There are a lot of helpful resources

- [Stata Cheatsheet](#)
- [Base R Cheatsheet](#)
- [Tidy R Cheatsheet](#)

