R and RStudio Cloud
A HeRtie Basic Review

Stats I TAs
2020

Introductory note

- This is a review on R, NOT a review on Stats I.
  - RStudio Cloud and how it works.
- Nevertheless, we will be using different examples from the lecture and show you how to apply them in R.
- Use these slides/script as a cheat sheet for your future self!
- What do we cover? What we have seen so far, and what we speculate that you *might* need for the FDA!
- In this review, R code marked in blue font

Agenda

- RStudioCloud intro
  - Scripts, functions and objects
  - Clean environment and set working directory
- Datasets:
  - How to open, clean, and get familiar with them (summaries, structure)
- Subsetting information
- Tables and CrossTabs in R
- Linear regression
- Graphs

R, RStudio & RStudio Cloud

R, Rstudio, and Rstudio Cloud

R

The language & the program: the motor

RStudio

The interface or “Integraded Development Environment” (IDE)

Rstudio Cloud

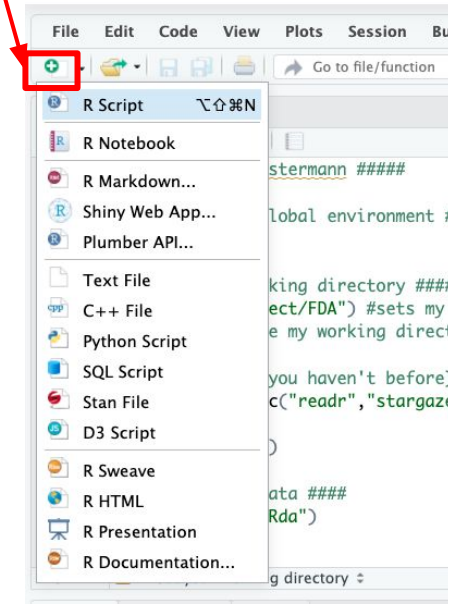
The interface on the cloud

Scripts

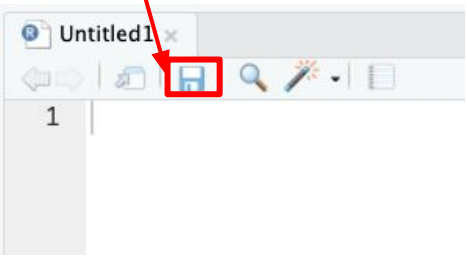
- Use scripts!!!
- Scripts allow you to:
  - Write and correct your functions, objects and libraries
  - Document your work (for yourself and others!!)
- In the FDA, you will be required to attach your script.

- Create, name & save a new Script:

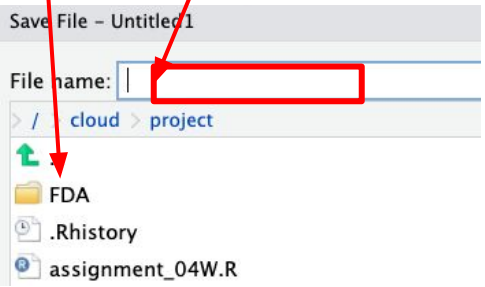
1. Click on the plus symbol and then on R Script.



2. Click on the save symbol



3. Choose a name and a folder



Scripts - good coding practices

```
##############################
# HeRtie R workshop
# Stats I
##############################
##############################

# 1. Clean your global environment ####
rm(list=ls())
cat("\014")  #Delete everything in the console

# 2. Set your working directory ####
setwd("/cloud/project/workshop") #sets my working directory
getwd() # shows me my working directory

# 3. Install (if you haven't before) and load packages/libraries  ####
install.packages(c("readr","stargazer","dplyr","readxl", "haven"))
library(readr)
library(stargazer)
library(dplyr)
library(foreign)
library(readxl)
library(writexl)
library(haven)

# 4.  Load your data ####
```

1. Start by cleaning your environment.
2. Set working directory
3. Load libraries (if any)
4. Load your data
5. Use # (hashtag) to write comments (everything after the # will be treated as something for human eyes)

Now go to the Script

R: Data Types and Structures

Objects

- Most general concept/term
- What is an object?
  - Almost everything that we use on R
  - A “vessel”: something that contains something else (vectors, data-frames, functions, etc.)
- How to create an object?
  - Decide on a name and use the assignment operator (“=”, “<-” or “->”)
  - tom = 3
  - “random text” -> victor
- Why use objects?
  - They are handy, practical and you can use them over and over again!
  - Example: reg <- lm(prestige ~ education + income + women, data= Prestige)

R data types

R has 4 basic data types:

- **character**: "a", "swc"
- **numeric**: 2, 15.5
- **integer**: 2L (the L tells R to store this as an integer)
- **logical**: TRUE, FALSE

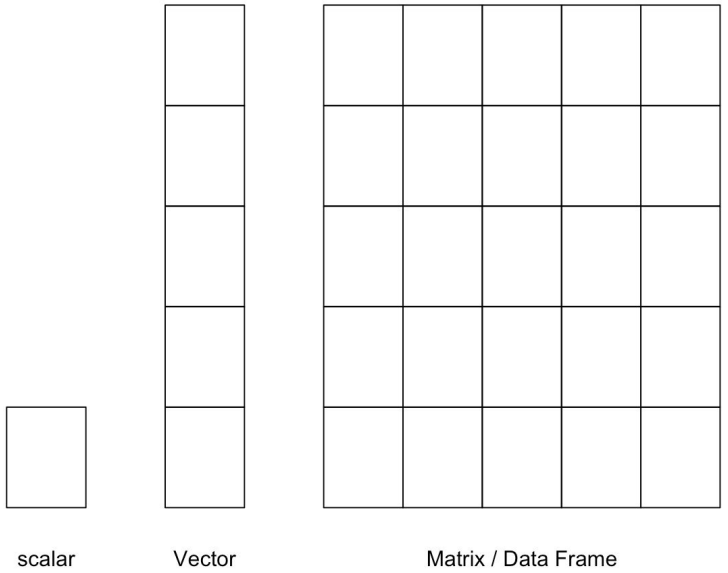
R has many data structures. These include:

- **vectors** (character, logical, integer or numeric)
- **matrix**
- **data frame**
- **factors**

	Linnear	**Rectangular**
All same type	vector	matrix
Mixed	list	data frame

Vectors

- Decide on a name and
- use the assignment operator (“=”, “<-” or “->”)
    - tom = 3
    - “random text” -> victor
    - x <- c(10.4, 5.6, 3.1, 6.4, 21.7)



Source: Phillip (2018)

Vectors

- Different ways of creating vectors

Command in R	Gives you
a <- c(2,5,8)	2 5 8
b <- 2:8	2 3 4 5 6 7 8
c <- seq(2,4, by=0.5)	2.0 2.5 3.0 3.5 4.0
d <- rep(1:2, times=4)	1 2 1 2 1 2 1 2
e = rep(1:2, each=3)	1 1 1 2 2 2

Now go to the Script

Logical vectors

- Logical vectors are generated by conditions.
- Take the values: TRUE, FALSE, NA.

```
student_ages <- c(20,22,28,31,27,24,23,21,25,26)
student_ages > 25
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE
```

- Logical vectors to clean our dataset from NAs:

`is.na(world$corrupt)` #returns a logical vector of the same size as world$corrupt with value TRUE if the corresponding element is NA

`!is.na(world$corrupt)` #returns TRUE for those values that are Not NA

`world.clean<-world[!is.na(world$corrupt), ]` #would clean dataset of NAs (rows with NAs for variable HDI)

Now go to the Script

Factors

- Factors are used to store categorical data.
- Can be unordered (when the data is categorical/nominal) or ordered (when the data is ordinal)
- Factors are stored as integers (1,2,3 …) , and have labels associated with these unique integers.
    - Let’s see an example in R

Now go to the Script

Matrices and Dataframes

A dataframe is a “rectangular” type of object. Typically observations are in the rows and variables in the columns.

The dataframe contains values that several observations take for different variables. All columns have the same length.

Create a dataframe of boat sales:

```
bsale<- data.frame(name = c("a", "b", "c", "d", "e", "f", "g", "h", "i", "j"),
        color = c("black", "green", "pink", "blue", "blue",
                  "green", "green", "yellow", "black", "black"),
                age = c(143, 53, 356, 23, 647, 24, 532, 43, 66, 86),
                price = c(53, 87, 54, 66, 264, 32, 532, 58, 99, 132),
                cost = c(52, 80, 20, 100, 189, 12, 520, 68, 80, 100),
                stringsAsFactors = FALSE)   # Don't convert strings to factors!
```

Note that the code above is somehow artificial, in practice we don’t use this too much. A common task is to download/use data from official sources.

No, not this one!

Sí, this one!

Filter

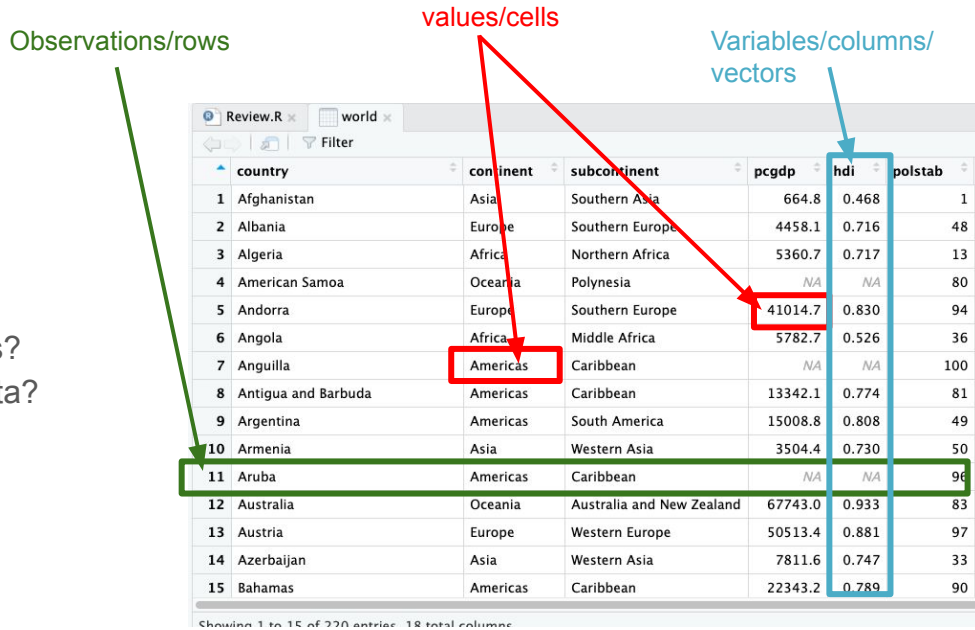
	name	color	age	price	cost
1	a	black	143	53	52
2	b	green	53	87	80
3	c	pink	356	54	20
4	d	blue	23	66	100
5	e	blue	647	264	189
6	f	green	24	32	12
7	g	green	532	532	520
8	h	yellow	43	58	68
9	i	black	66	99	80
10	j	black	86	132	100

Matrices and Dataframes

It is more common for us to open a dataset/dataframe that we download from official websites, etc. (We also sometimes merge different datasets given to us).

Let's load and take a look our dataframe:

```
load("indicators.Rda") # Load an Rda dataset in R
View(world)        # Open the data in a new window
head(bsale)        # Show me the first few rows
str(bsale)         # Show me the structure of the data
names(bsale)       # What are the names of the columns?
nrow(bsale)        # How many rows are there in the data?
```



	country	continent	subcontinent	pcgdp	hdi	polstab
1	Afghanistan	Asia	Southern Asia	664.8	0.468	1
2	Albania	Europe	Southern Europe	4458.1	0.716	48
3	Algeria	Africa	Northern Africa	5360.7	0.717	13
4	American Samoa	Oceania	Polynesia	NA	NA	80
5	Andorra	Europe	Southern Europe	41014.7	0.830	94
6	Angola	Africa	Middle Africa	5782.7	0.526	36
7	Anguilla	Americas	Caribbean	NA	NA	100
8	Antigua and Barbuda	Americas	Caribbean	13342.1	0.774	81
9	Argentina	Americas	South America	15008.8	0.808	49
10	Armenia	Asia	Western Asia	3504.4	0.730	50
11	Aruba	Americas	Caribbean	NA	NA	96
12	Australia	Oceania	Australia and New Zealand	67743.0	0.933	83
13	Austria	Europe	Western Europe	50513.4	0.881	97
14	Azerbaijan	Asia	Western Asia	7811.6	0.747	33
15	Bahamas	Americas	Caribbean	22343.2	0.789	90

Showing 1 to 15 of 220 entries, 18 total columns

Now go to the Script

Opening different datasets

vs.

Datasets: open and save (I)

Type	From	Command in R	Package
.csv	Excel, Stata,..	df <- read_csv(“c:/mydata.csv”) write_csv(df, “mydata.csv”) df <- read.csv("c:/mydata.csv") df <- read.table("c:/mydata.csv", header=TRUE, sep=",", row.names="id")	library(tidyverse) / library(readr) library(foreign) library(utils)
.xlsx	Excel	df<- read_xlsx(“c:/mydata.xlsx”) write_xlsx(df, “c:/mydata.xlsx”) mydata <- read.xlsx("c:/myexcel.xlsx", 1) mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet") # read in the worksheet named mysheet	library(readxl) library(writexl) library(xlsx)

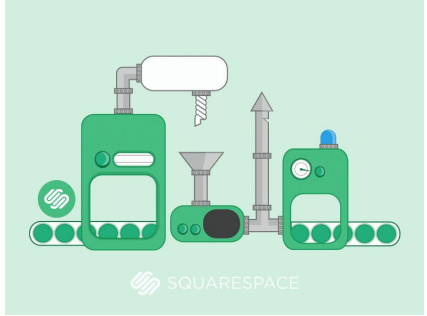
Datasets: open and save (II)

Type	From	Command in R	Package
.dta	Stata	df <- read_dta("c:/mydata.dta") write_dta(df, "c:/mydata.dta") df <- read.dta("c:/mydata.dta") df <-read.dta13(“c:/mydata.dta”)	library(haven) library(foreign)#if file generated w/ before Stata v13 library(readstata13)#if file generated w/ Stata v.13 or newer
.Rda	R :-)	load("dataset_asia.Rda") save(asia_full, file = "dataset_asia.Rda")	No package needed, baseR Note: here you don’t need to assign a name to your dataframe
.por	SPSS	mydata <- spss.get("c:/mydata.por", use.value.labels=TRUE) # last option converts value labels to R factors	library(Hmisc)

Now go to the Script

Functions

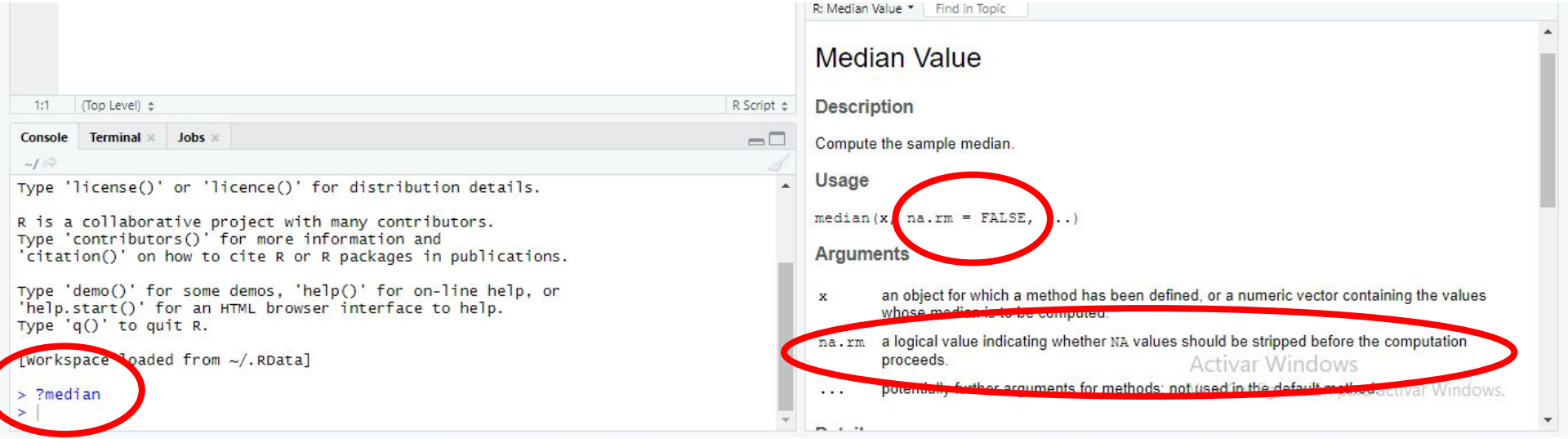
Functions (I)



- What is a function?
  - A set of predefined instructions that allows us to do something in R.
  - You give an input, you get an output (see upper right corner)
- What “something”?
  - Almost anything! +2,7000,000 indexed functions to date (https://www.rdocumentation.org/)
- How to recognise a function?
  - They are ALWAYS a name, followed by brackets, such as: “some.function()” or “anyfunction()”
- How to use a function?
  - Essentially, put what you want to apply the function to in the brackets
- Tip: Not sure how a specific function works?
  - You can always type, in the console, the question mark before the function name to find out more! E.g.: “?some.function” or “?anyfunction”.

Functions (II)

- Let's see an example!
- median()
- Inside the brackets we define the "arguments" (e.g. default value of na.rm is "FALSE")



Functions (III)

- Some basic functions
  - mean(x)
  - var(x)
  - sd(x)
  - length(x), unique(x)
  - load(x), library(x) - opening datasets
  - cbind(x), rbind(x) - connecting columns or rows
  - table(x), prop.table(x)
  - summary(x)
  - is.na(x)
  - plot(x), hist(x), density(x), boxplot()
  - cut(x) - recoding data from interval to categorical
  - pnorm(x), qnorm(x)
  - t.test(x)
  - lm(x) - your (soon-to-be) favourite function

Now go to the Script

Data Transformation: Subsetting

Subsetting parts of your dataset

- Always start by writing the name of the object you want to subset (data set or variable), and then squared brackets:
  - world.clean[*rows* , *columns*]

- Ask yourself: Are you subsetting rows or columns?
  - Rows: removing missing observations, or looking at certain groups of observations in your data, e.g. democratic countries:
    world.clean[world$demo == “Democratic” ,]
  - Columns: including only the variables that you are interested in:
    world.clean[, c(“continent”, “hdi”, “demo”)]

- If you leave either *rows* or *columns* blank, you include everything

Now go to the Script

Optional: Subsetting with the “dplyr” package

- Functions from the “dplyr” package can be a good alternative to subsetting in base R:
  - filter() - subsets rows/observations by matching conditions
  - select() - subsets columns/variables by name

- In these functions, the first argument is the name of the dataset, and the following arguments are the variables you want manipulate:

- First, I want to subset rows from the dataset, so I only have European countries that experience economic growth:

  filter(world.clean, continent == “Europe”, growth_cat == “Growth”)

- Second, I want to subset columns from the dataset, so I only have information about country name, per capita GDP, and economic growth category:

  select(world.clean, country, pcgdp, growth_cat)

Now go to the Script

T-Test

Difference of Means test (t-Test)

- Do democratic and non-democratic countries on average differ in their number of homicides?
  1. Create a two subsets - one for democratic countries and one for non-democratic countries

```
demo <- world.clean[ world.clean$demo == “Democratic” ,  ]
nondemo <- world.clean[ world.clean$demo == “Non-Democratic” ,  ]
```

  2. Use the t.test function to compare the means of the two subsets

```
t.test(demo$homicide, nondemo$homicide)
```

```
        Welch Two Sample t-test

data:  demo$homicide and nondemo$homicide
t = -1.259, df = 117.61, p-value = 0.2105
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -7.117840  1.584973
sample estimates:
mean of x mean of y
 8.218182 10.984615
```

Now go to the Script

Crosstabs

Crosstabs with table() & prop.table()

- What proportion of countries on a specific continent (e.g. Asia) are democratic?
- table()
  - You can generate frequency tables using the **table( )** function:

    ```
    mytable <- table(world.clean$continent, world.clean$demo) # world$continent will be rows, world$demo will be columns
    mytable # print table
    ```

- prop.table()
  - You can then generate tables of proportions using the **prop.table( )** function:

    ```
    prop.table(mytable) # cell percentages
    prop.table(mytable, 1) # row percentages (to answer the initial question, we need this command)
    prop.table(mytable, 2) # column percentages
    ```

Now go to the Script

Linear Regression

Linear regressions (I)

- Create an object to contain a linear regression.
- For example, with the dataset “Indicators”, we can make a bivariate regression with “corrupt” (control of corruption) as the dependent variable and “polstab” (political stability) as the independent variable...
    - bi_reg <- lm(corrupt ~ polstab, data = world.clean)
- ...or make a multiple regression (don’t forget the “+”)...
    - mul_reg <- lm(corrupt ~ polstab + homicide, data = world.clean)

Linear regressions (II)

- ...use summary() to get your regression output in the console...
  - summary(bi_reg)

```
> summary(bi_reg)

Call:
lm(formula = corrupt ~ polstab, data = world.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-48.439 -12.867   0.714  12.243  57.657

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.12157    2.86114   2.489   0.0139 *
polstab      0.82634    0.05086  16.247   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 17.62 on 150 degrees of freedom
Multiple R-squared:  0.6377,    Adjusted R-squared:  0.6352
F-statistic:   264 on 1 and 150 DF,  p-value: < 2.2e-16
```

Linear regressions (III)

- ...and stargazer() to create a nice table that you can use in publications...

  - library(stargazer)
  - stargazer(mul_reg, title = "multiple regression", out = "reg.txt")

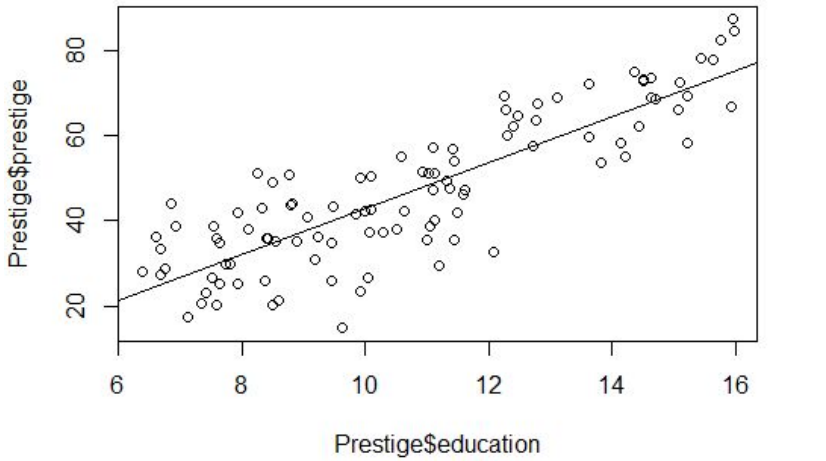
Now go to the Script

Data Visualization

Graphs (I)

- You can use the plot() function to create scatterplots, and abline() to draw lines (such as the regression line) on bivariate regressions

    - plot(Prestige$education, Prestige$prestige) # first the “x”, then the “y”
    - abline(bi_reg) # the regression that we made before

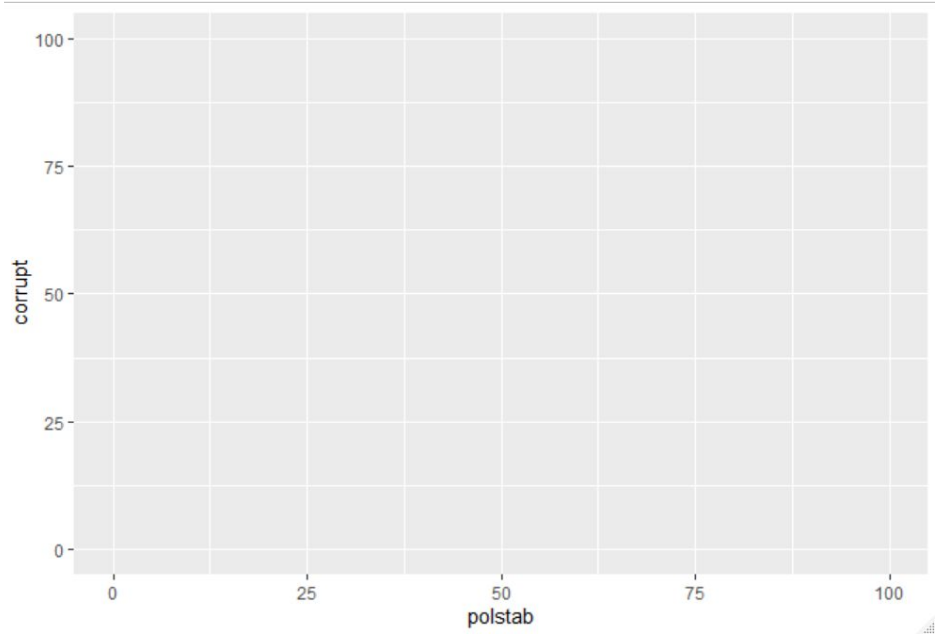


Graphs (II)

- Or use the ggplot2 library!!
- Basic idea: you put layers upon layers of what you want to plot:

```
library(ggplot2)

ggplot(world.clean, aes(x = polstab, y = corrupt)) # specify dataset & x and y axes
```

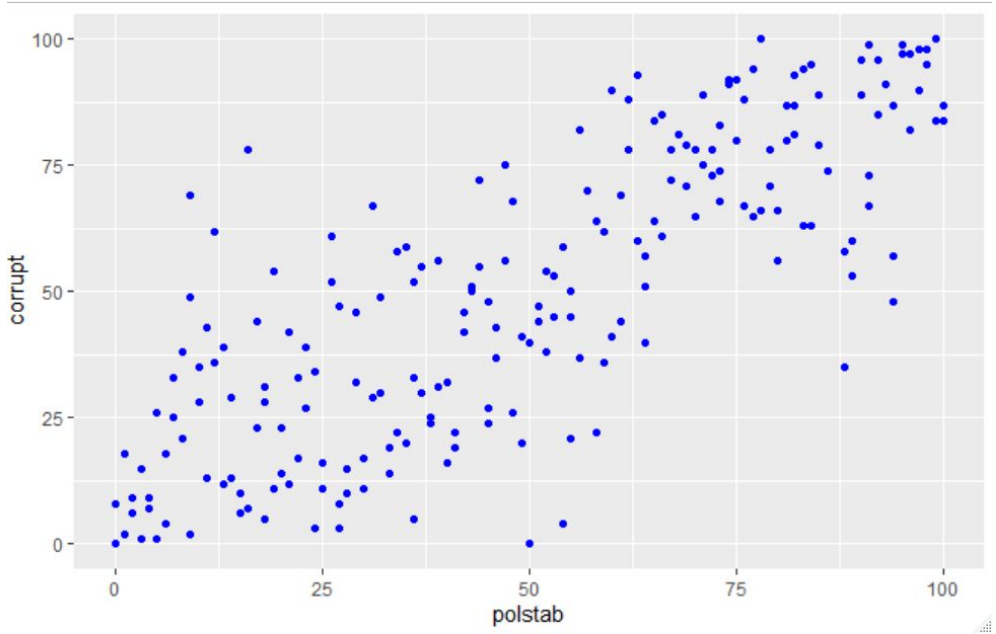


Graphs (III)

- Basic idea: you put layers upon layers of what you want to plot:

```
library(ggplot2)

ggplot(world, aes(x = polstab, y = corrupt)) +
geom_point(color= “blue”) # specify that we want data points. The “+” is very important!
```

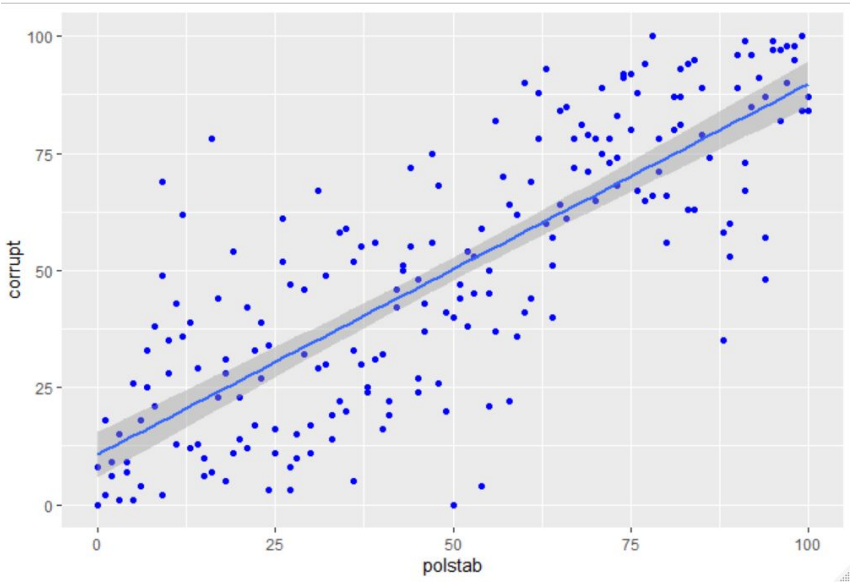


Graphs (IV)

- Basic idea: you put layers upon layers of what you want to plot:

```
library(ggplot2)

ggplot(world.clean, aes(x = polstab, y = corrupt)) +
geom_point(color= “blue”) +
geom_smooth(method = “lm”)
```

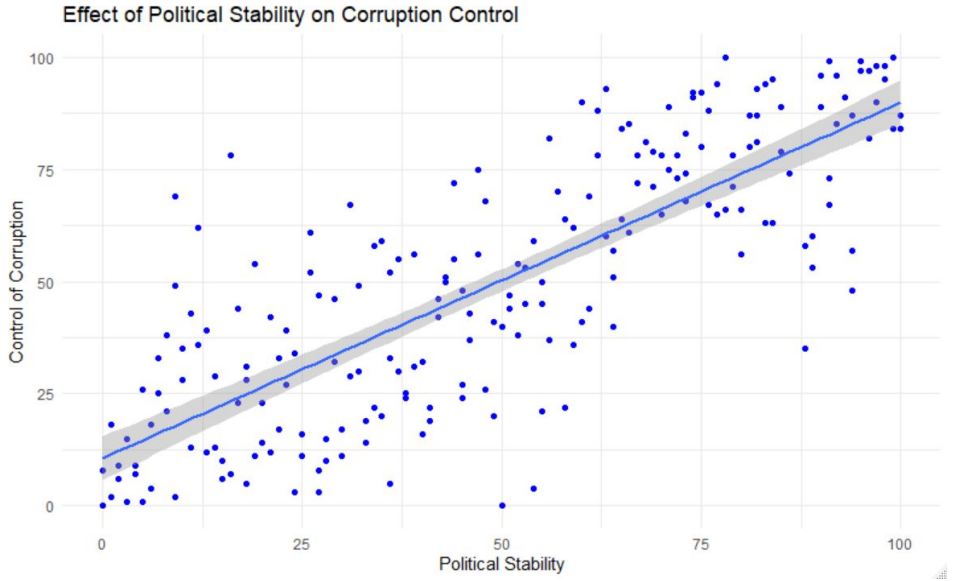


Graphs (V)

- Now, let's make our plot even prettier:

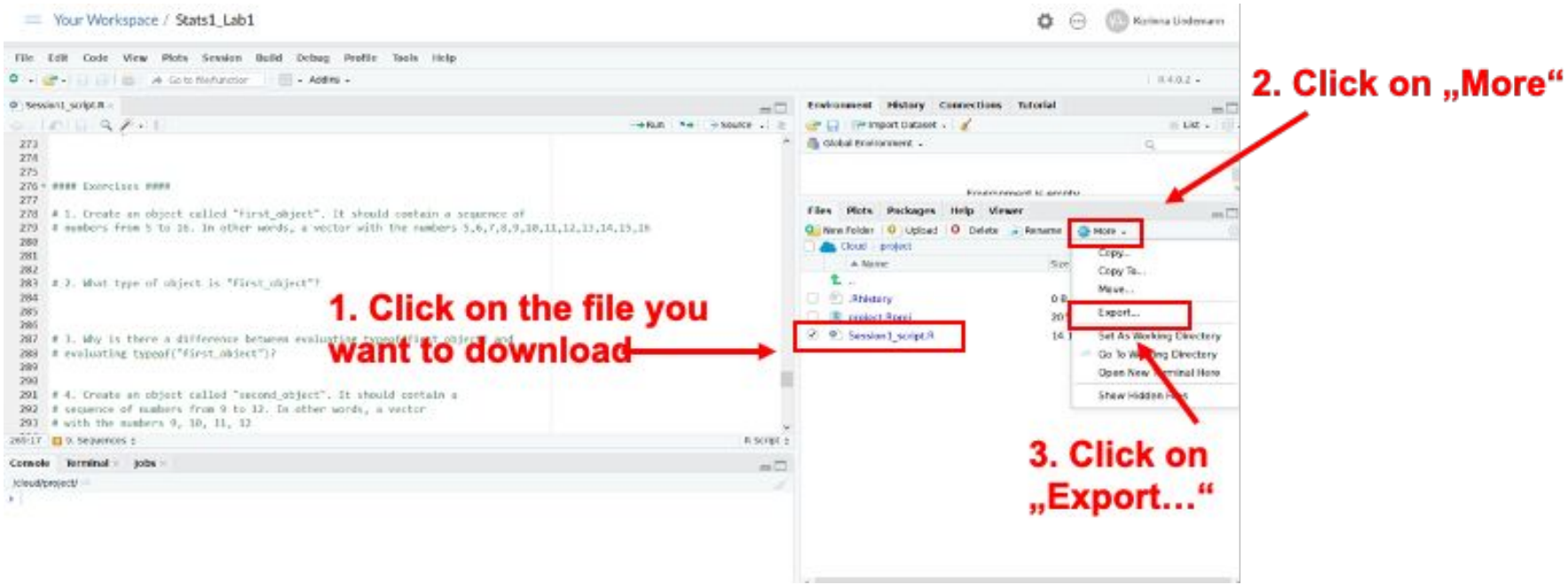
```
library(ggplot2)

ggplot(world.clean, aes(x = polstab, y =
corrupt))+
geom_point(color="blue") +
geom_smooth(method = "lm") +
theme_minimal() +
labs(x="Political Stability",
      y="Control of Corruption",
      title = "Effect of Political Stability
            on Corruption Control")
```



Now go to the Script

Once you are done. Download your Script.



BACK-UP SLIDES

Setting up R and RStudio for the first time

Download both R and RStudio:

- R (the language that we will use):
  - https://cran.r-project.org/
- RStudio (the interface that we will use):
  - https://www.rstudio.com/products/rstudio/download/

- Once you have downloaded them both, you only need to open RStudio to be able to use R

How RStudio works