MANCHESTER 1824
The University of Manchester

SeM-in-Rs

A brief introduction to R

February 25th 2020

Dr Patricio Troncoso
patricio.troncoso@manchester.ac.uk
@ptroncosoruiz

With thanks to Dr Ana Morales-Gomez (UKDS)
This presentation uses materials published by the UKDS here

MIE
S E M

Plan for today

10:00-10:45	R basics
10:45-11:15	Practical 1: Getting started with R
11:15-11:30	Break
11:30- 12:00	Descriptive statistics, data manipulation and graphics
12:00-12:30	Practical 2: GBBO demo
12:30-13:00	Practical 3: Data manipulation and graphics

R Basics

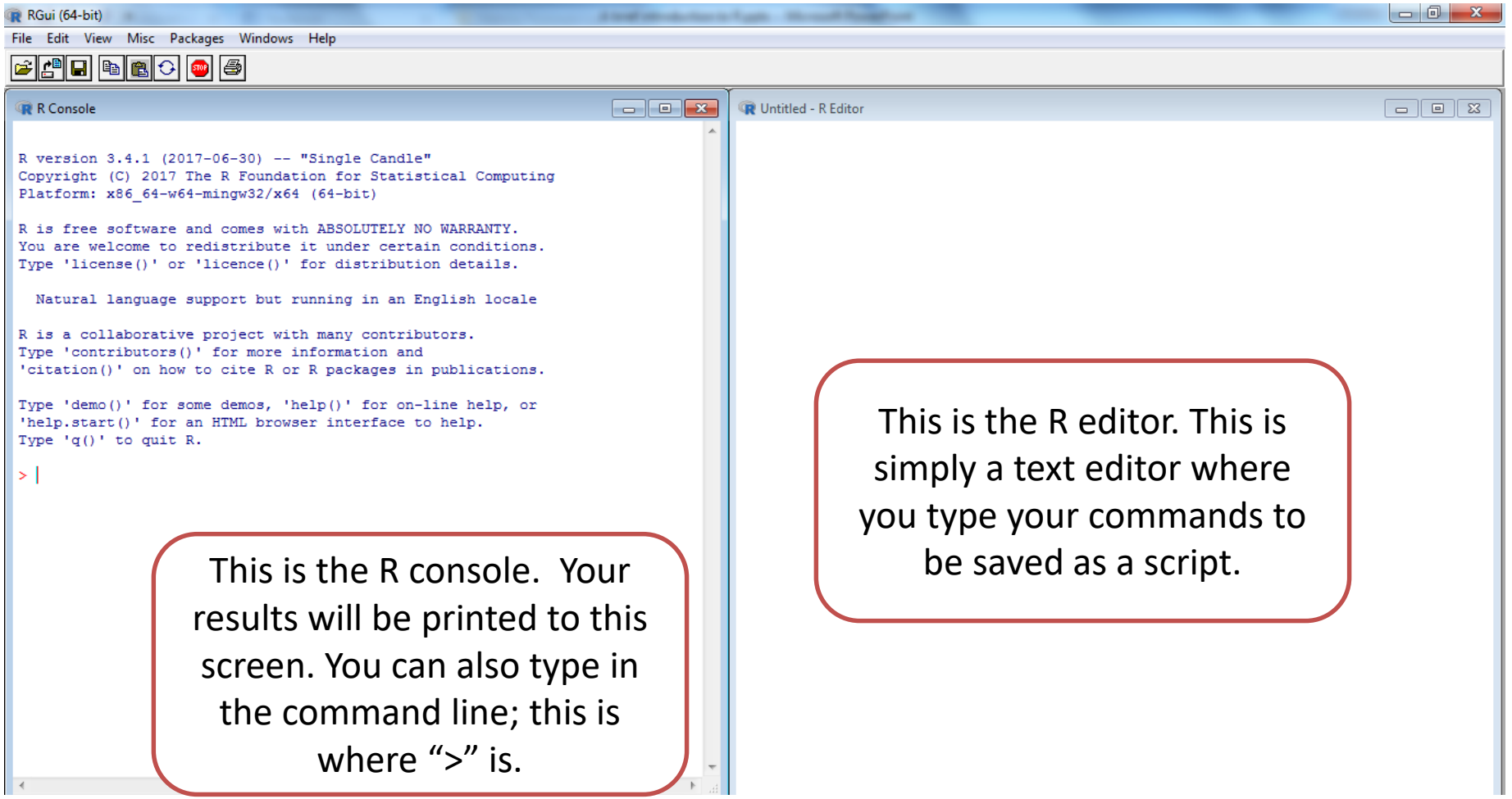
What is R?

- A high-level programming language
- free-ware
- in addition to basic functions (*base* package) researchers have contributed thousands of packages http://cran.r-project.org/web/packages/
- runs on most platforms
- syntax is easy to share and repeat

The leaRning pRocess

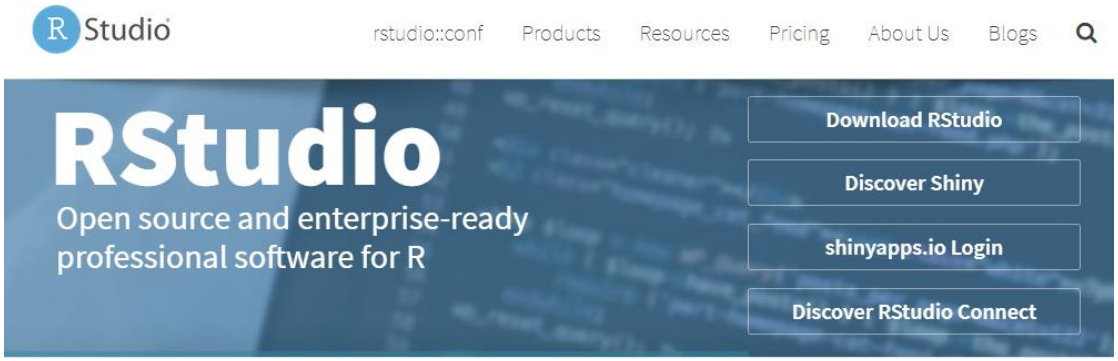
- Trial and error (actually errors... and lots of them!)
- Search code online:
  - Quick R: http://www.statmethods.net/
  - http://www.ats.ucla.edu/stat/r/
  - http://stackoverflow.com/
  - https://stats.stackexchange.com/
  - https://github.com/trending/r
  - http://www.cookbook-r.com/
  - See also the swirl R tutorial on the web http://swirlstats.com
  - Or… simply google your questions
- Copy code, modify it if necessary and run it
- Repeat

The R interface

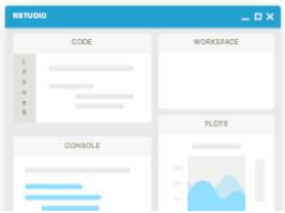


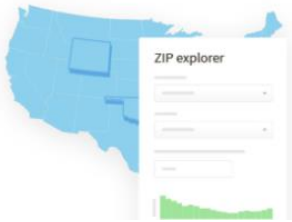
RStudio

- RStudio is a free and open-source integrated development environment (IDE).



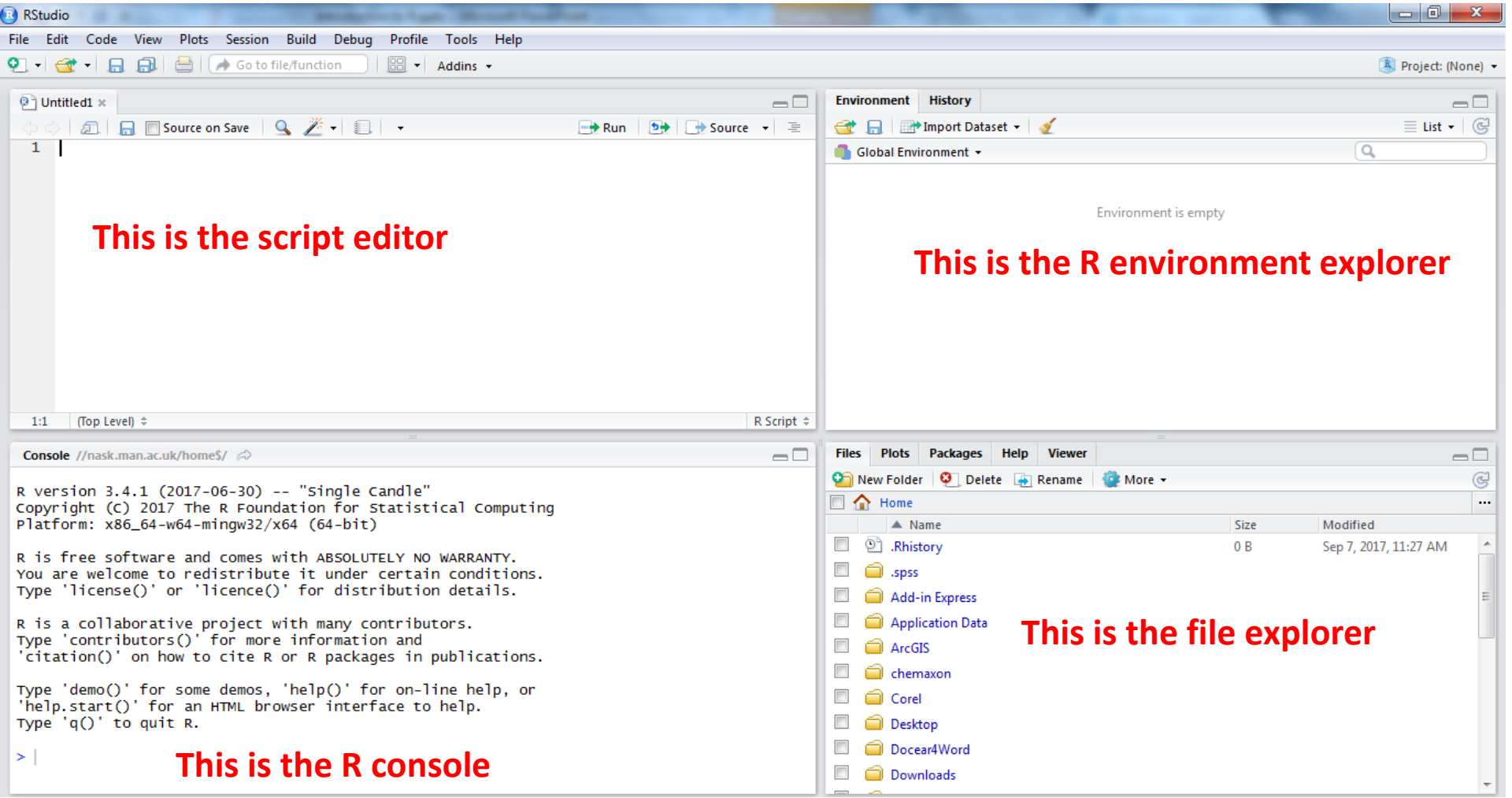
Everything you can do in R, you can do in RStudio





For some, it's simply a matter of taste

RStudio - Windows



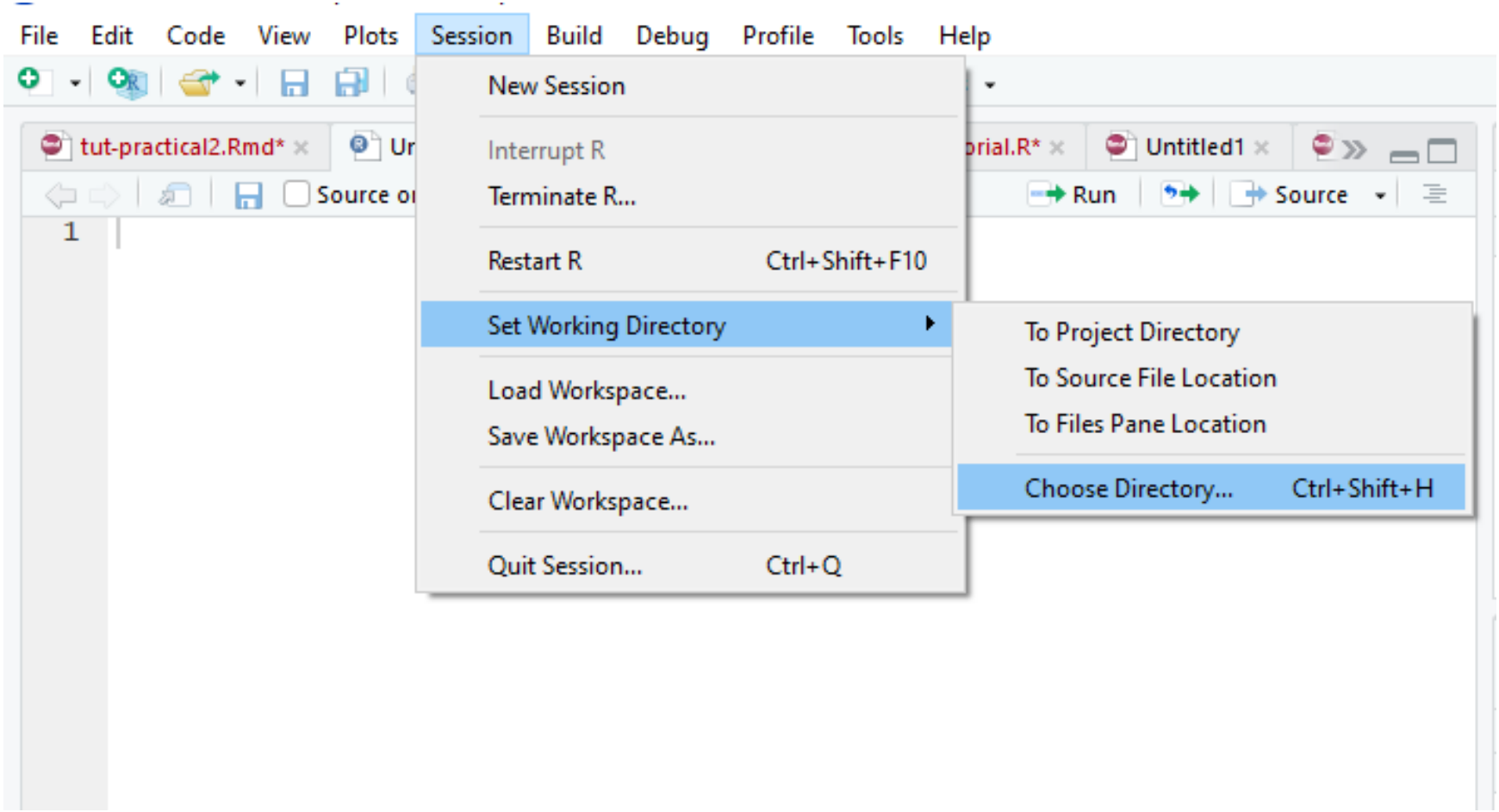
First things first…
The working directory (wd)

- Tells R where your files R (are)
- Tells R where to save our new analyses and figures
- Code to set the working directory:

  setwd(“a link to the folder in your machine”)
- To check where the working directory is

  getwd()
- OR…

Set the working directory (in Rstudio)



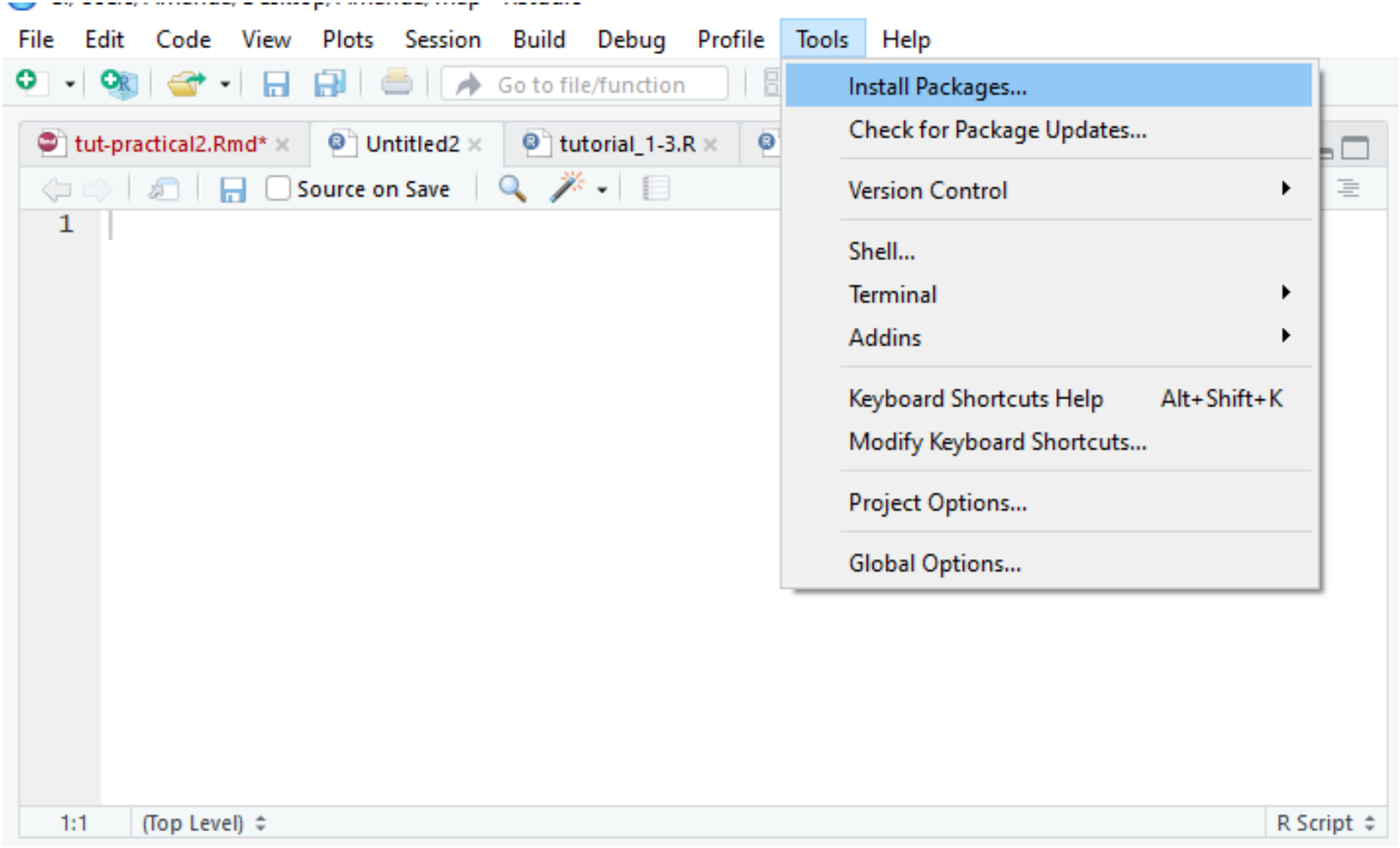
Installing packages

- Set of basic operations pre-installed: base package.
- R needs packages to do certain tasks
  - Haven: For importing datasets in other formats (SPSS, Stata, csv, etc).
  - ggplot2: For graphs
  - Tmap: For maps
- Code

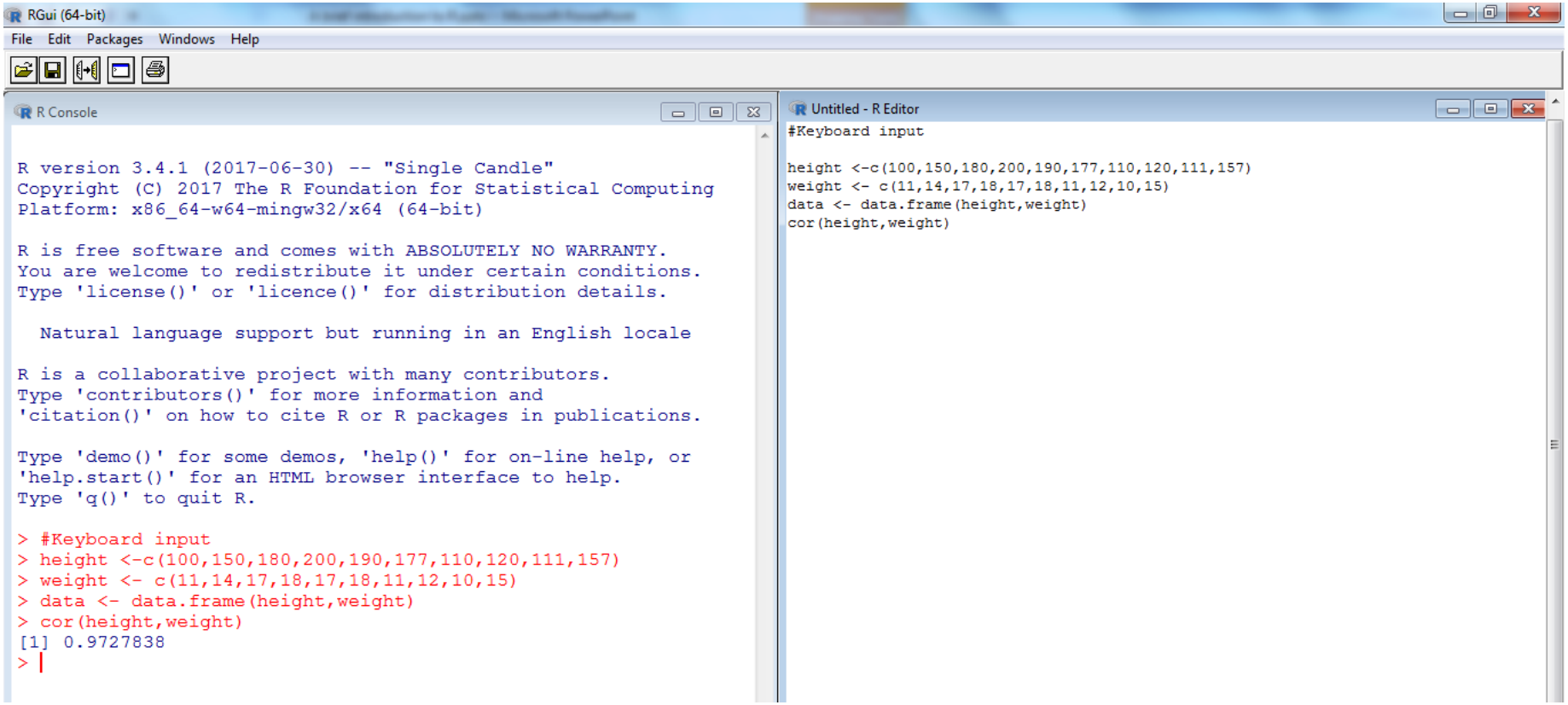
  install.packages(“haven")

  install.packages(“haven“, “ggplot2”)
- OR…

Installing packages (in Rstudio)



Keyboard input



Copy and paste on the R editor:

height <- c(100, 150, 180, 200, 190, 177, 110, 120, 111, 157)

weight <- c(11, 14, 17, 18, 17, 18, 11, 12, 10, 15)

data <- data.frame(height, weight)

cor(height, weight)

Working with datasets

- First, tell R where your data is; i.e. define your working directory
- Second, install/load the required package(s)

  install.package(ggplot2)

  library(ggplot2)
- Third, Import the data:

  Stata files: read.dta(“mydata.dta”)

  CSV files: read.csv(“mydata.csv”)

  SPSS files: read.spss(“mydata.sav”)
  - Give your data a name!: **name<- read_dta(“mydata.dta”)**
  - Remember:
    - R is case sensitive, be careful with spaces and capitals/lower case
    - Choose an informative and easy to type name for your data. You will need to write it a lot while you analyse!

Important to remember!

- Scripts are used to save our work and analyses
  - Can be stored as R script or Notepad
  - Can be opened again in later sessions
  - Can be copied and modified (and easily shared with others)
- Console shows the operations
  - This is not saved!
- Environment stores variables created, datasets, values, etc.
  - These are only available for the duration of the session

R objects (1)

- We will be usually working with data frames
  - A set of rows and columns, e.g. rows are individuals and columns are variables.
  - This is R jargon for what we usually understand as “dataset”.
- A vector or column (regardless of type) is a one-dimensional set of elements.
  - A vector can contain numbers, text, dates, etc.
  - The class of a vector affects how it’s treated by certain commands.
- To check what kind of object you’re dealing with, type:

  class(name_of_object) # you can check the class of any object in the R environment tab.

  class(data$variable) # to find out about a variable within a data frame.
- Estimation commands will also return “model fit” objects containing information about the model you ran.
  - E.g. “lm” objects are returned when an OLS regression is run.

It’s time for a computer practical!

Data manipulation

Loading data

- Data can be read into R in various ways:

  read.csv()

  read.delim()

  # or using the “foreign” package

  read.spss()

  read.dta()
- You can also use the package “haven” to read data in various formats (similar to “foreign”)
- If the file to be read is in the wd, then:

  read.csv(“mydata.csv”, header=T)
- Alternatively, specify full path:

  read.csv(“C:/mydirectory/mydata.csv”, header=T)
- You can also read data from online sources:

  read.delim("https://us.sagepub.com/sites/default/files/upm-binaries/26934_exercise.dat")

Working with data.frames

- An R data frame is simply a series of rows and columns with headers indicating variable names and row numbers indicating case number.
- To perform operations on variables, we need to specify the data frame and the variable:

  table(mydata$wcond) # this will run a frequency table

mydata

Filter

	wcond	class	w1sn02	w2sn02	w1int	w2int	w1att	w2att	z1pbc	z2pbc	sqw1	sqw2
1	1	1	7.00	7	7.0	7.0	7.0	7.0	7.00	7.00	2.5495098	2.5495098
2	1	1	5.00	3	5.5	3.5	6.0	4.0	4.75	4.25	2.3452079	2.3452079
3	1	1	2.00	2	2.5	1.5	4.5	3.5	1.75	2.50	0.7071068	0.7071068
4	1	1	4.00	2	3.5	1.5	4.0	2.5	4.25	2.75	0.7071068	0.7071068
5	1	1	2.00	2	3.0	1.5	4.5	3.0	5.25	3.50	1.2247449	0.7071068
6	1	1	2.00	4	6.0	3.0	5.5	4.5	3.75	2.75	1.8708287	2.1213203
7	1	1	2.00	4	3.5	4.0	5.5	4.5	5.50	4.00	2.1213203	2.1213203
8	1	1	2.00	4	4.0	4.0	5.0	4.0	3.50	4.00	1.5811388	1.5811388
9	1	1	3.00	2	4.5	3.0	6.0	4.5	3.50	2.50	0.7071068	0.7071068

Showing 1 to 10 of 503 entries

Editing data (with dplyr)

- The base package comes fully loaded with many routines for data manipulation.
- A complete course of data manipulation could be done. We'll focus on some frequently used functions.
- The "dplyr" package is a powerful tool that complements the base package nicely.

Package “dplyr”, selected functions

- select() select columns (variables)
- filter() select row (cases)
- rename() to rename variables
- join() to merge two dataset
- summarise() to summarise data (mean, SD)
- group_by to summarise data by groups or variables

An example in R

select()

select(dataset, variable1, variable2, variable3)

select(dataset, variable1:variable3)

** The minus sign before a variable tells R to drop the variable.

select (dataset, variable1, -variable2, -variable3)

select (dataset, variable1, -c(variable2, variable3))

filter()

filter(dataset, condition)

filter(dataset, month==”August”)

*** Multiple conditions can be used

filter(dataset, month==”August” & birth_year==1990)

recode()

- To recode a categorical variable, we use recode_factor()
  - "football" into "sports"

dataset$new_variable<- recode_factor(dataset$old_variable, "football" = "sports")

  - "football" and "tennis" into "sports"

dataset$new_variable<- recode_factor(dataset$old_variable, "football" = "sports", "tennis"= "sports")

** The new value will always be a category

join() and rename()

left_join(dataset1,dataset2)

Return: all variables and cases for dataset1 plus new variables from dataset2. If there are cases in dataset1 that are not in dataset2, then a NA (missing) value is given

rename()

rename(dataset_name,
new_variable_name=old_variable_name)

summarise() and group_by()

summarise(name_of _dataset_in_R, mean(variable))

summarise(name_of _dataset_in_R, sd(variable))

group_by()

*** This function works better when using in combination with other functions

group_by(gbbo, hometown=="London")

Descriptive statistics and graphics

Descriptive Statistics (1)

- You can search the R reference card for frequently used functions.
- To obtain the mean of a variable, type:

  mean(data$height)
- Standard deviation:

  sd(data$height)
- Or a quicker way, would be to get a summary of statistics:

  summary(data$height)

Descriptive Statistics (2)

- For categorical variables, you can run frequency tables by typing:

  table(x) # that gives the absolute frequency

  prop.table(table(x)) # cell percentages

- For a crosstab, you only need to add another variable:

  table(x, y) # prop.table() works in the same way.

  prop.table(table(x,y), 1) # row percentages

  prop.table(table(x,y), 2) # column percentages

  chisq.test(table(x, y)) # for a chi-square test

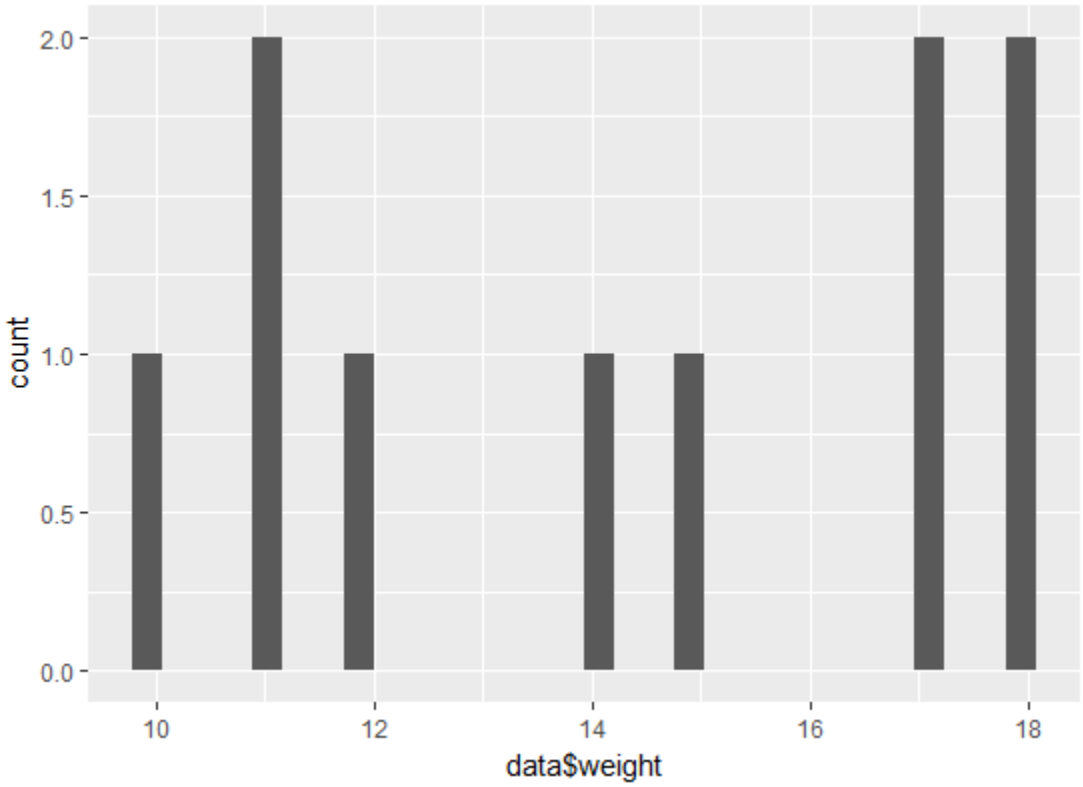
Dealing with missing values

- Missing values in R are coded “NA” (Not assigned).
- For an example, type the following in R:
  x <- c(7,9,2,NA,1)
- Then, get the mean:
  mean(x) # This gives “NA”
- To obtain the mean, you need to specify:
  mean(x, na.rm=TRUE) # This gives 4.75
- Now type:
  y <- c(1,0,0,0,1,NA,0,0,0,1,NA) # There are 11 observations
- Then get a frequency table:
  table(y) # The table shows 9 observations
- Perhaps typing “na.rm=TRUE” will work. Does it?
  table(y, na.rm=TRUE)
- Check the explanation on this website. Hint: type Ctrl+F and then “missing” to get there quicker.

Plotting (1)

- ggplot2 package:

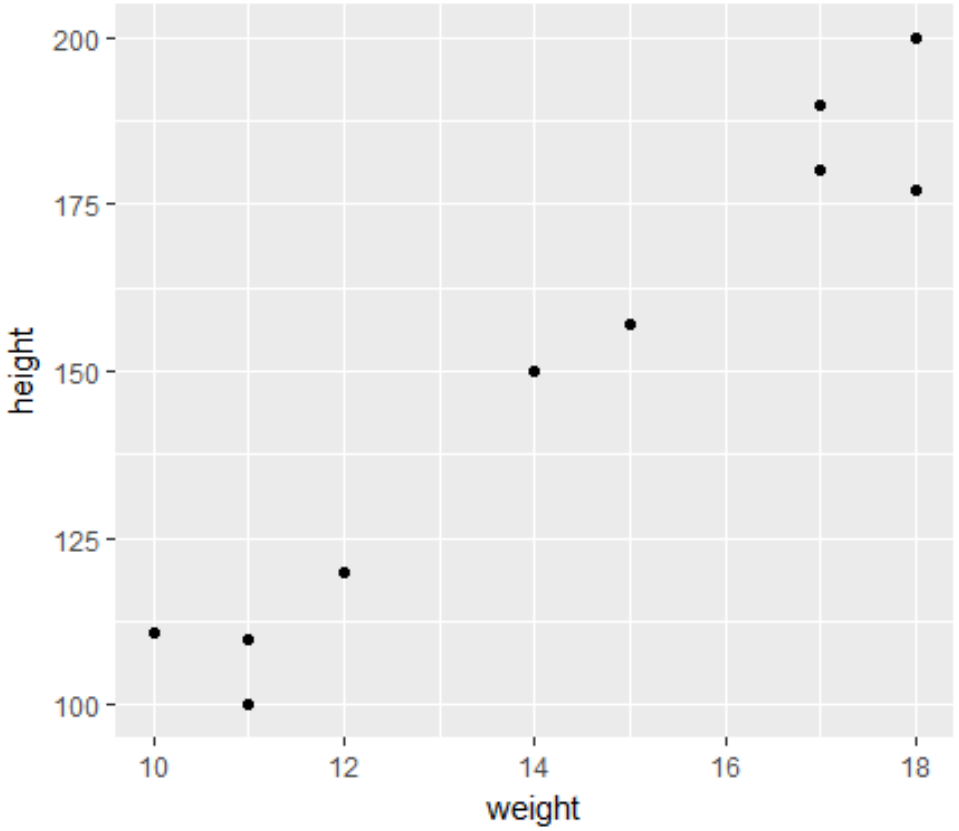
  ggplot(data=data, aes(weight)) + geom_histogram()



Plotting (2)

- ggplot2 package:

  ggplot(data=data, aes(y=height, x=weight)) + geom_point()



Saving plots

- Plots need to be saved to the wd. Copying to clipboard may distort images.

  ```
  png(“plot.png", width = 11.69, height = 16.53, units = "in", res = 600)
  plot(weight, height) # Any plot command (including ggplot)
  dev.off()
  ```

- Or you can also save as pdf ,as such:

  ```
  pdf()
  plot(…)
  dev.off()
  ```

  NB: Specified values of width and height are the measures in inches of size A4.

It’s time for a computer practical!

Next SeM-in-R session

Regression in R and Mplus

(March 3rd, Chemistry G.08, 10:00 – 13:00)

Rescheduled to March 6th, 10:00 – 13:00)

- Basics of regression
- Model specification
- Interpretation of output
- Plotting predictions and diagnostics