

Data Analytics for Management

Overview & Introduction to R

Igor Vyshnevskyi

Endicott College of International Studies

Friday, Oct 28 / Nov 4, 2022

Agenda

1. Overview
2. Getting Readied
3. Introduction to R
4. Assignment

1. Overview

About This part of the course

Description

- The second half of the course is designed to provide students with some **basic** exposure with R syntax and programming environment
- We will start from scratch, assuming **no prior knowledge** at all: installing the software, using R as a calculator, and understanding R objects

Prerequisites

- Some exposure to quantitative methods (e.g. taking Statistics) is preferable but not required
- Open mind to **(try and fail)*Inf**

Learning Objectives

Learn how to:

- ~~become an R expert~~
- do basic arithmetic operations, creating objects, and loading and saving files
- subset data by conditions and gain basic insights into visualization with R

Prepare you for:

- your future job
- other data related courses
- etc.

Learning Objectives (cont'd)

Help you start a journey to become:

- a **smart consumer** of data science
- an **informed producer** of elementary data science product

Accept sucking and persevere

Sucking

"Whenever you're learning a new tool, for a long time you're going to suck. It's going to be very frustrating. But the good news is that that is typical, it's something that happens to everyone, and it's only temporary.

Unfortunately, there is no way to go from knowing nothing about a subject to knowing something about the subject ... without going through a period of great frustration and much suckiness.

But remember, when you're getting frustrated, that's a good thing, it's typical, it's temporary. **Keep pushing through** and in time it will become second nature."

- Hadley Wickham

Sucking



Karen Rinaldi, "(It's Great to) Suck at Something"

To suffer through sucking, together

- We work together
- There are no stupid questions
- Don't get lost
- Ask questions, in class and via office hours, etc.
- Help others
- Google and Stackoverflow are your best friends

Important logistics for remaining part of the course

- Two home works
- Individual Project
- Group Project (aka Final exam)

Why R?



Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

20% complete



For more information about this issue and possible fixes, visit
<https://www.windows.com/stopcode>

If you call a support person, give them this info:
Stop code: `UNEXPECTED_KERNEL_MODE_TRAP`

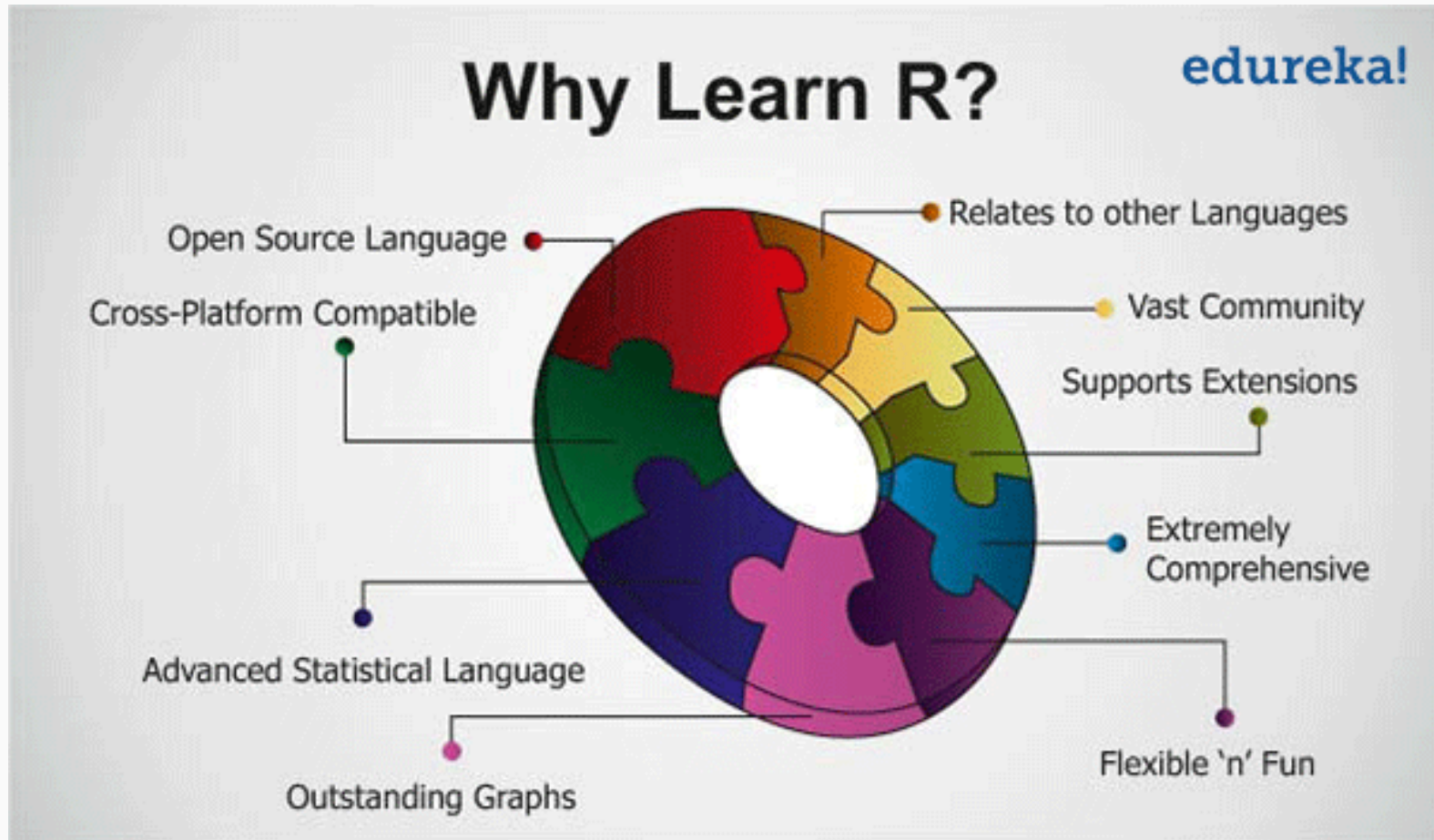
Why R?



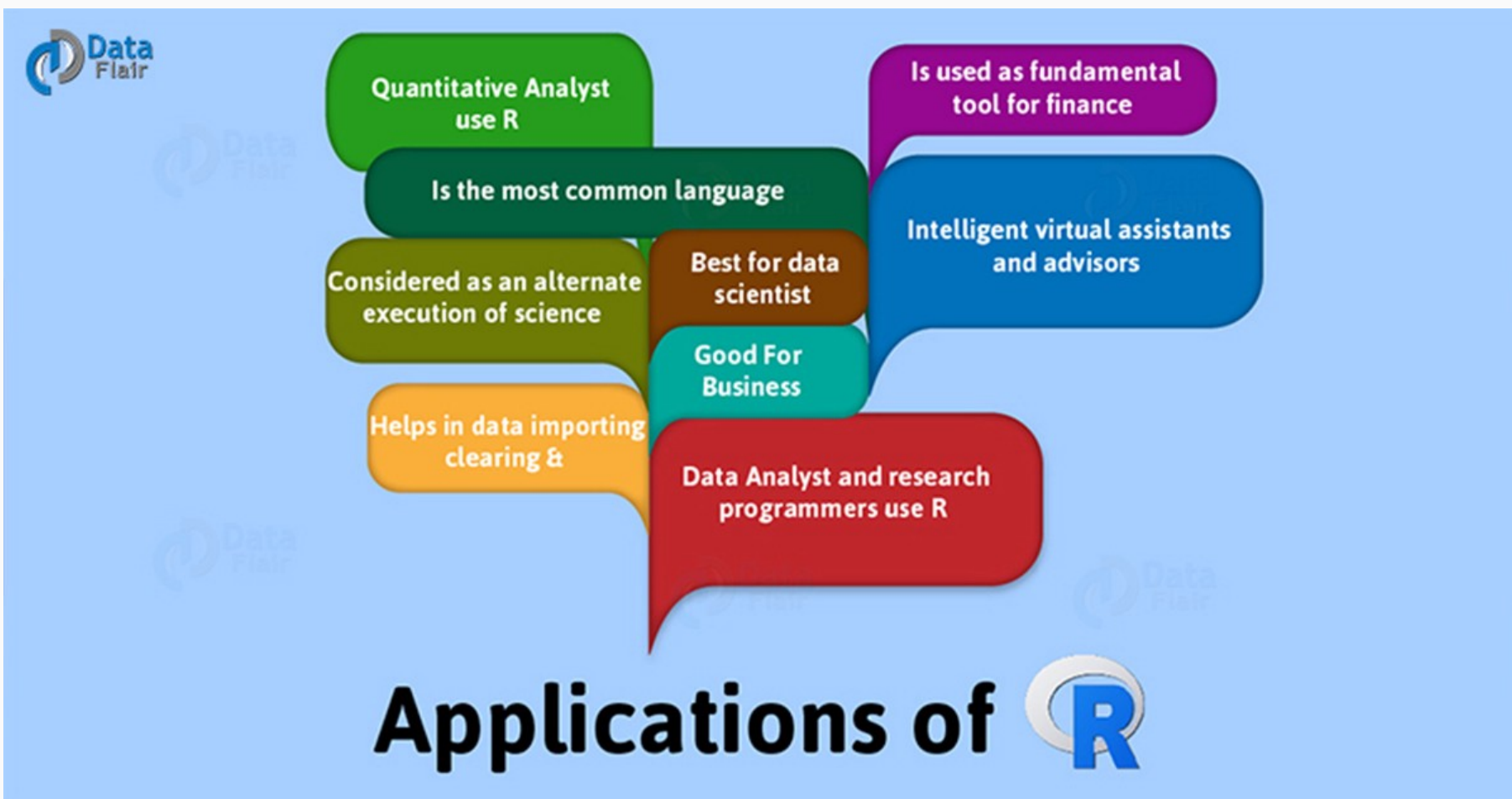
Why R?



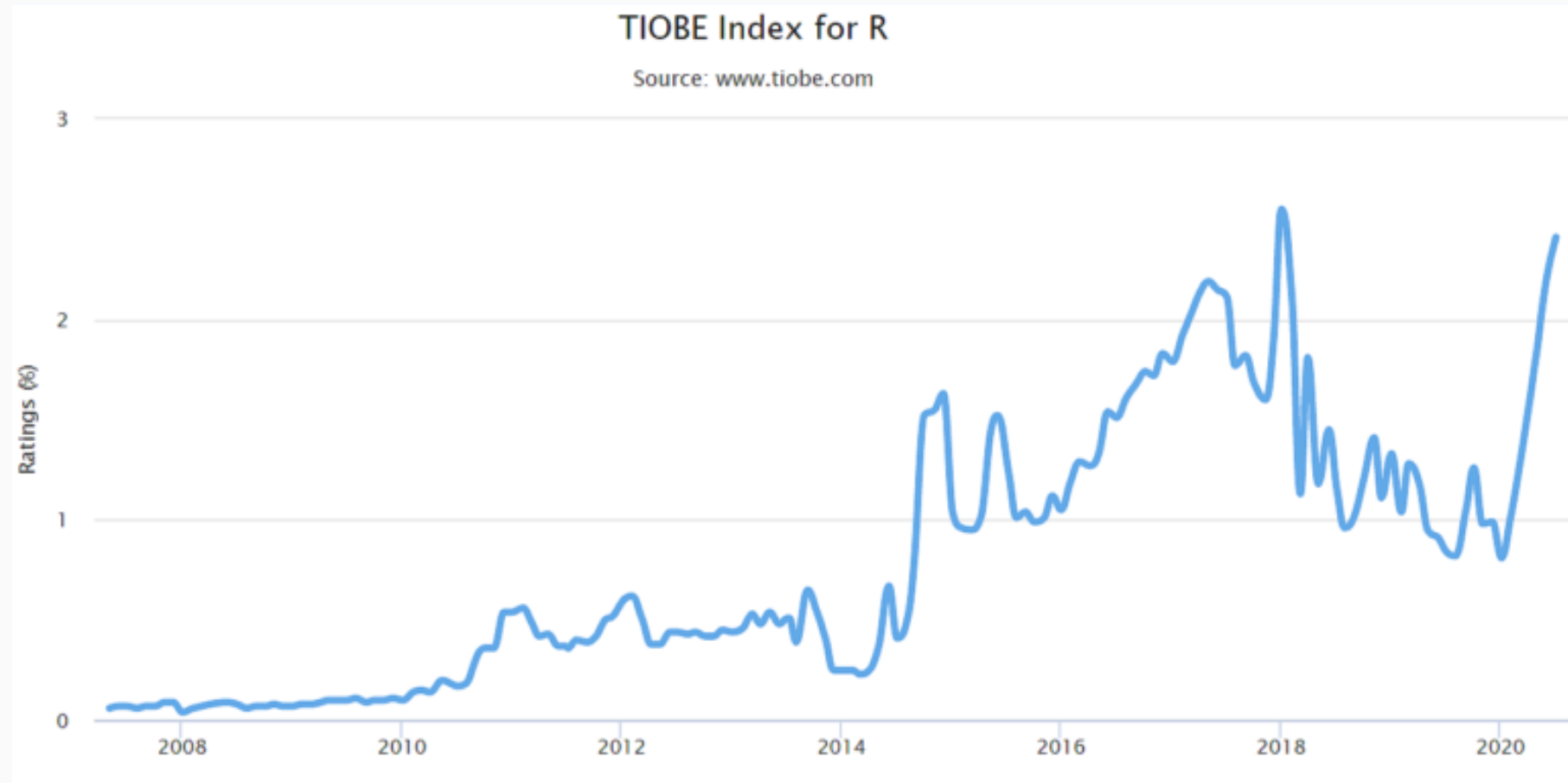
Why R?



Why R?



Why R?



Why R?

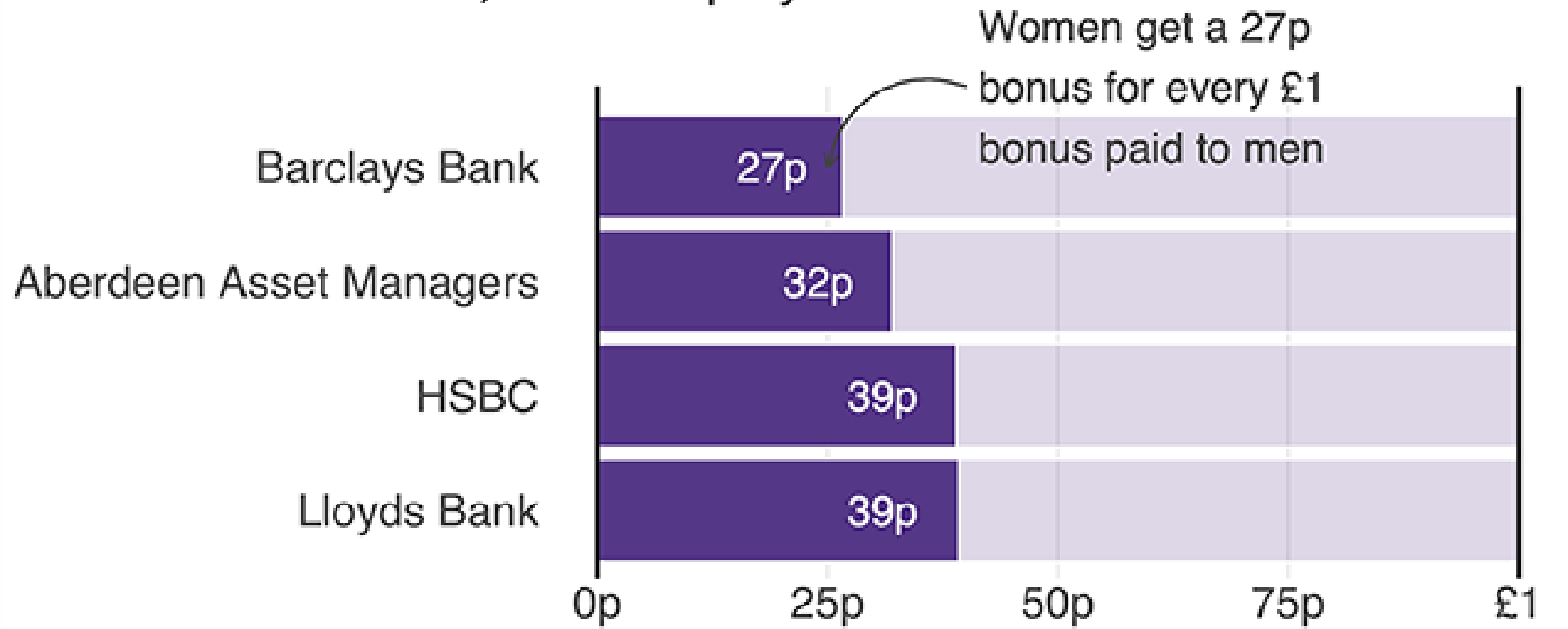
Why R?

SKILL	2017
R	\$ 105,977
Unix	\$ 105,829
HL7 (Health Level 7)	\$ 105,807
Database Testing	\$ 105,806
MVS (Multiple Virtual Storage)	\$ 105,718
ISO 9000	\$ 105,636
SoapUI	\$ 105,581
Workday	\$ 105,574
NumPy	\$ 105,526
XAML (eXtensible Application Markup Language)	\$ 105,507
NetSuite	\$ 105,488

Why R?

Finance firms have the largest bonus gaps

Women's bonuses as a proportion of men's among finance firms with 5,000+ employees



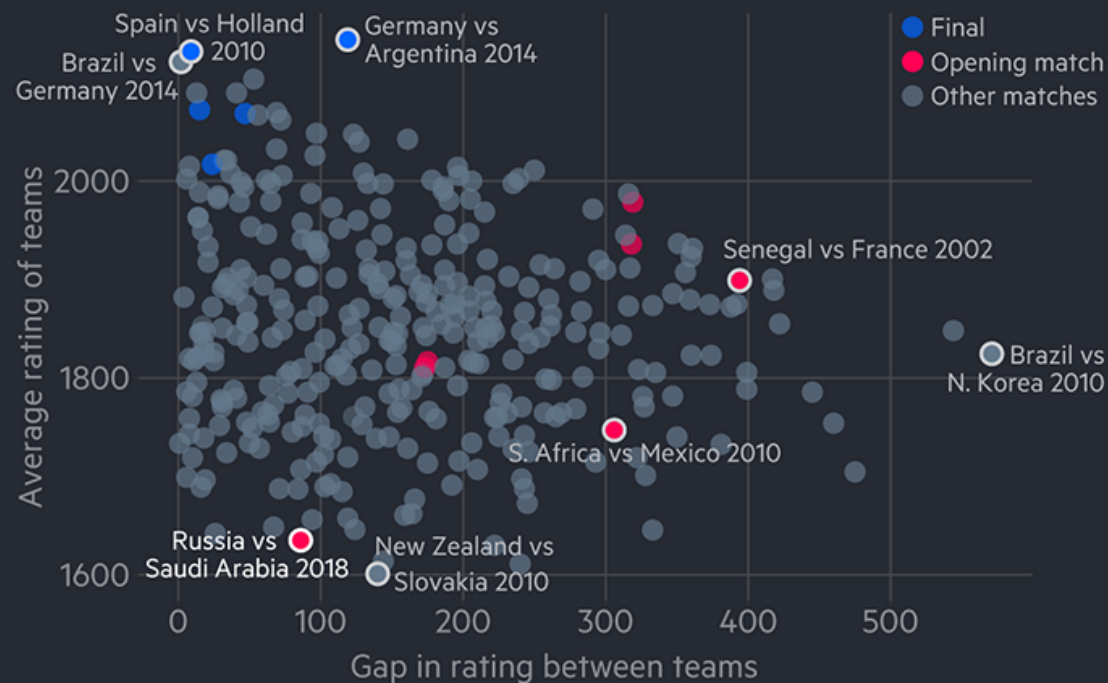
Source: 10,016 companies that reported their data



Why R?

Russia vs Saudi Arabia: where does the oil state derby rank among the weakest World Cup matches?

Circles represent every World Cup match since 1998



Source: eloratings.net

FINANCIAL TIMES

Learning R



Jesse Maegan

@kierisi

Following



My **#rstats** learning path:

1. Install R
2. Install RStudio
3. Google "How do I [THING I WANT TO DO] in R?"

Repeat step 3 ad infinitum.

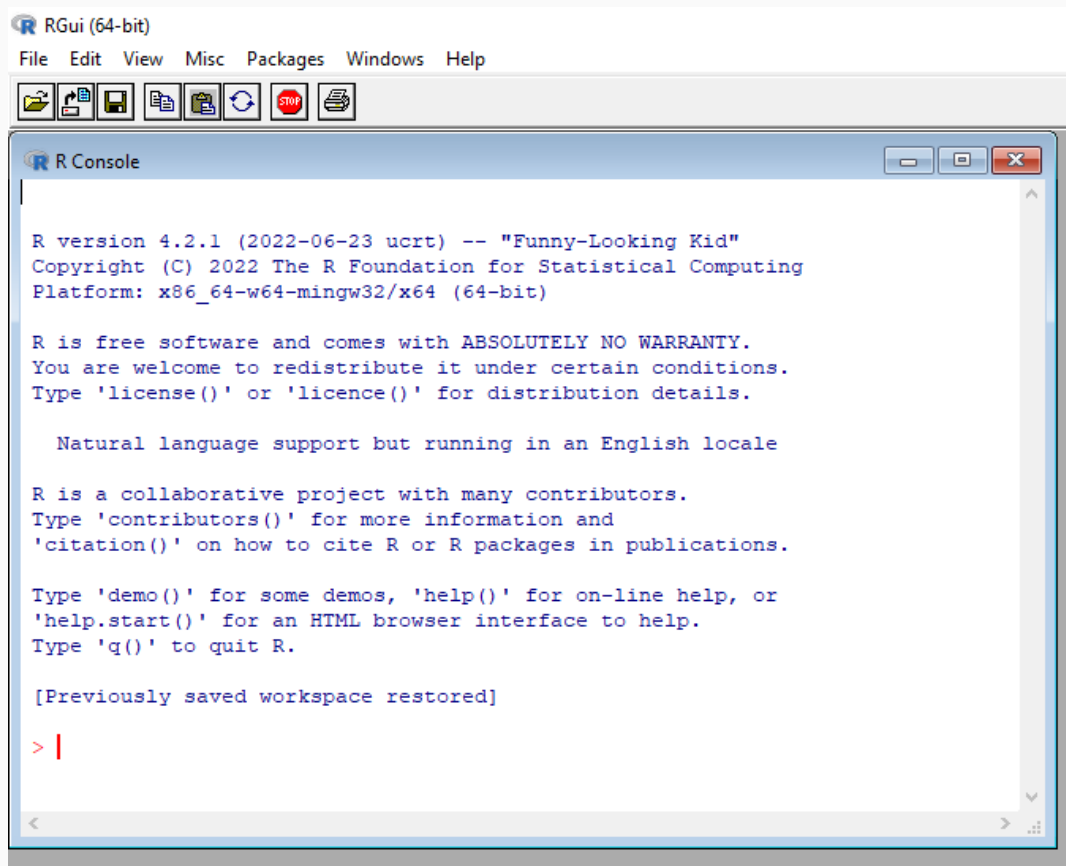
7:19 AM - 18 Aug 2017

2. Getting Readied

In this section

1. Install **R**
2. Install **Rstudio**

Get R



The screenshot shows the RGui (64-bit) window. The menu bar includes File, Edit, View, Misc, Packages, Windows, and Help. The toolbar contains icons for file operations and execution. The R Console window displays the following text:

```
R version 4.2.1 (2022-06-23 ucrt) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

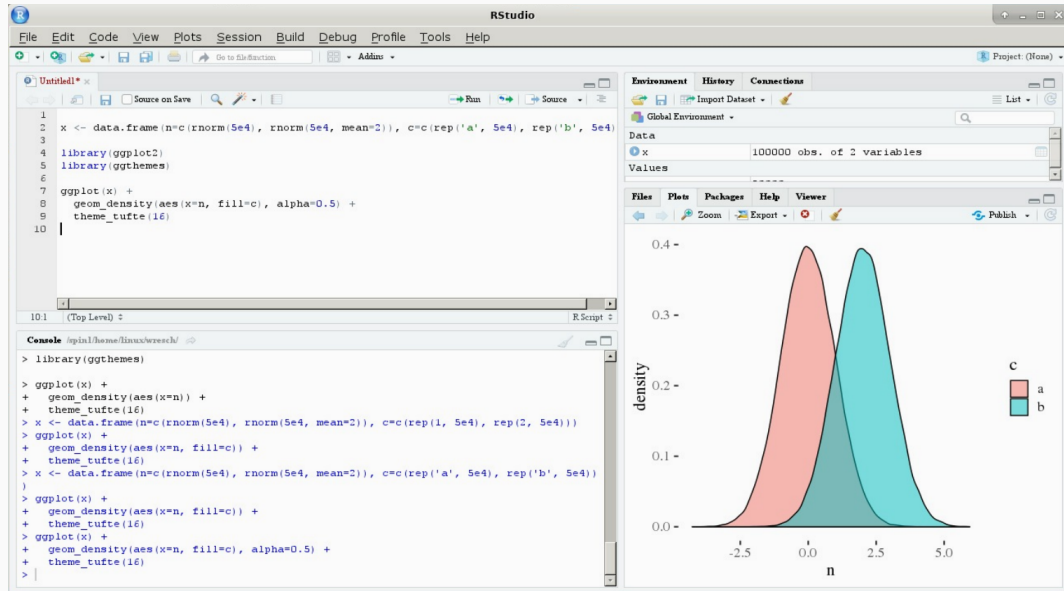
> |
```



Get R (cont.)

Go to <https://cran.r-project.org/> and choose the needed System to further download and install R

Get RStudio



Download the RStudio IDE

Go to <https://www.rstudio.com/products/rstudio/download/> and choose the needed System to further download and install RStudio IDE (integrated development environment)

In-class activity

- There will be many group activities
- When in groups: talk to one another, and help each other
- You will be informed what to upload into the system

3. Introduction to R

In this section

1. Arithmetic Operations
2. Objects
3. Vectors
4. Functions
5. Data Files
6. Saving Objects
7. Packages
8. Programming and Learning Tips

1. Arithmetic Operations

```
6 + 3
```

```
## [1] 9
```

```
6 - 3
```

```
## [1] 3
```

```
6 / 3
```

```
## [1] 2
```

```
6 ^ 3
```

```
## [1] 216
```

1. Arithmetic Operations

```
6 * (10 - 3)
```

```
## [1] 42
```

```
sqrt(4)
```

```
## [1] 2
```


2. Objects

```
result ← 6 + 3  
result
```

```
## [1] 9
```

```
print(result)
```

```
## [1] 9
```

2. Objects

```
result <- 6 - 3  
result
```

```
## [1] 3
```

2. Objects

Result

```
## Error in eval(expr, envir, enclos): object 'Result' not found
```

2. Objects

```
iegor ← "instructor"  
iegor
```

```
## [1] "instructor"
```

2. Objects

```
iegor ← "instructor and researcher"  
iegor
```

```
## [1] "instructor and researcher"
```

2. Objects

```
Result ← "6"  
Result
```

```
## [1] "6"
```

```
Result / 3
```

```
## Error in Result/3: non-numeric argument to binary operator
```

```
sqrt(Result)
```

```
## Error in sqrt(Result): non-numeric argument to mathematical function
```

2. Objects

```
result
```

```
## [1] 3
```

```
class(result)
```

```
## [1] "numeric"
```

```
Result
```

```
## [1] "6"
```

```
class(Result)
```

```
## [1] "character"
```

```
class(sqrt)
```

```
## [1] "function"
```

3. Vectors

Table 1.2. World Population Estimates.

<i>Year</i>	<i>World population (thousands)</i>
1950	2,525,779
1960	3,026,003
1970	3,691,173
1980	4,449,049
1990	5,320,817
2000	6,127,700
2010	6,916,183

Source: United Nations, Department of Economic and Social Affairs, Population Division (2013). *World Population Prospects: The 2012 Revision, DVD Edition.*

3. Vectors

- A vector or a one-dimensional array represents a collection of information stored in a specific order.
- **c()** stands for concatenate, to enter a data vector containing multiple values with commas separating different elements of the vector we are creating

```
world.pop ← c(2525779, 3026003, 3691173, 4449049, 5320817, 6127700, 6916183)
```

```
world.pop
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

3. Vectors

- `c()` can be also used to combine multiple vectors

```
pop.first ← c(2525779, 3026003, 3691173)
pop.second ← c(4449049, 5320817, 6127700, 6916183)
pop.all ← c(pop.first, pop.second)
pop.all
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

3. Vectors

- To access specific elements of a vector, we use **square brackets []**

```
world.pop[2]
```

```
## [1] 3026003
```

```
world.pop[c(2,4)]
```

```
## [1] 3026003 4449049
```

```
world.pop[c(4,2)]
```

```
## [1] 4449049 3026003
```

```
world.pop[-3]
```

```
## [1] 2525779 3026003 4449049 5320817 6127700 6916183
```

3. Vectors

```
pop.million ← world.pop / 1000  
pop.million
```

```
## [1] 2525.779 3026.003 3691.173 4449.049 5320.817 6127.700 6916.183
```

3. Vectors

```
pop.rate ← world.pop / world.pop[1]  
pop.rate
```

```
## [1] 1.000000 1.198047 1.461400 1.761456 2.106604 2.426063 2.738238
```

3. Vectors

- Arithmetic operations can be done using multiple vectors

```
pop.increase ← world.pop[-1] - world.pop[-7]  
percent.increase ← (pop.increase / world.pop[-7]) *100  
percent.increase
```

```
## [1] 19.80474 21.98180 20.53212 19.59448 15.16464 12.86752
```

3. Vectors

- Finally, we can also replace the values associated with particular indices by using the usual assignment operator (<-).
- Below, we replace the first two elements of the percent.increase vector with their rounded values.

```
percent.increase[c(1,2)] ← c(20, 22)  
percent.increase
```

```
## [1] 20.00000 22.00000 20.53212 19.59448 15.16464 12.86752
```

In-class activity

- We have 26 students who are taking the course. Until the end of semester we have 5 three-hours classes for practicing R (remaining two - individual and group project presentations). What is the total amount of expected time spent in sessions for all students combined? Use R as a calculator to derive the answer.
- Create a vector of age for all members in your group. Assign the resulting vector into an object called age
- What's the value of the second element in your age vector? The first and the third?
- Some say that in our modern times, our age should be adjusted by a factor of 0.7. Multiply your age vector with 0.7 and create a new object called new_age
- Create a vector of names for all members in your group. Assign the resulting vector into an object called names.
- Upload to the System.

4.Functions

- Functions are important objects in R and perform a wide range of tasks.
- A function often takes multiple input objects and returns an output object. We have already seen several functions: `sqrt()`, `print()`, `class()`, and `c()`.
- In R, a function generally runs as `funcname(input)` where `funcname` is the function name and `input` is the input object. In programming (and in math), we call these inputs arguments.
- For example, in the syntax `sqrt(4)`, `sqrt` is the function name and `4` is the argument or the input object.

4.Functions

```
length(world.pop)
```

```
## [1] 7
```

```
min(world.pop)
```

```
## [1] 2525779
```

```
max(world.pop)
```

```
## [1] 6916183
```

```
range(world.pop)
```

```
## [1] 2525779 6916183
```

4.Functions

```
mean(world.pop)
```

```
## [1] 4579529
```

```
sum(world.pop) / length(world.pop)
```

```
## [1] 4579529
```

4.Functions

- When multiple arguments are given, the syntax looks like `funcname(input1, input2)`.
- The order of inputs matters. That is, `funcname(input1, input2)` is different from `funcname(input2, input1)`.

```
year ← seq(from = 1950, to = 2010, by = 10)  
year
```

```
## [1] 1950 1960 1970 1980 1990 2000 2010
```

4.Functions

```
seq(to = 2010, by = 10, from = 1950)
```

```
## [1] 1950 1960 1970 1980 1990 2000 2010
```

4.Functions

```
seq(from = 2010, to = 1950, by = -10)
```

```
## [1] 2010 2000 1990 1980 1970 1960 1950
```

```
2008:2012
```

```
## [1] 2008 2009 2010 2011 2012
```

```
2012:2008
```

```
## [1] 2012 2011 2010 2009 2008
```

4.Functions

- The **names()** function can access and assign names to elements of a vector.
- Element names are not part of the data themselves, but are helpful attributes of the R object.

```
names(world.pop)
```

```
## NULL
```

```
names(world.pop) ← year  
names(world.pop)
```

```
## [1] "1950" "1960" "1970" "1980" "1990" "2000" "2010"
```

```
world.pop
```

```
##      1950      1960      1970      1980      1990      2000      2010  
## 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

4.Functions

- The **function()** function can create a new function. The syntax takes the following form.

```
## myfunction ← function(input1, input2, ..., inputN) {  
##  
##     DEFINE `output` USING INPUTS  
##  
##     return(output)  
## }
```


4.Functions

- In this example code, myfunction is the function name, input1, input2, ..., inputN are the input arguments, and the commands within the braces {} define the actual function.
- Finally, the **return()** function returns the output of the function. We begin with a simple example, creating a function to compute a summary of a numeric vector.

```
my.summary ← function(x){ # function takes one input
  s.out ← sum(x)
  l.out ← length(x)
  m.out ← s.out / l.out
  out ← c(s.out, l.out, m.out) # define the output
  names(out) ← c("sum", "length", "mean") # add labels
  return(out) # end function by calling output
}
z ← 1:10
my.summary(z)
```

```
##      sum length  mean
##   55.0   10.0    5.5
```

```
my.summary(world.pop)
```

```
##      sum  length  mean
```

In-class activity

- Here's a function named "add_ten" that takes in one input, x, and returns x+10 as a result

```
add_ten <- function(x){ # function takes one input  
  return(x + 10)  
}  
add_ten(20)
```

```
## [1] 30
```

- Create another function named "add_to_both" that takes in two inputs, x and y, and returns a vector that adds 5 to x and 10 to y.

5: Data Files

- **CSV** or comma-separated values files represent tabular data.
 - This is conceptually similar to a spreadsheet of data values like those generated by Microsoft Excel or Google Spreadsheet.
 - Each observation is separated by line breaks and each field within the observation is separated by a comma, a tab, or some other character or string.
- **RData** files represent a collection of R objects including data sets.
 - These can contain multiple R objects of different kinds.
 - They are useful for saving intermediate results from our R code as well as data files.

5: Data Files

- **Working Directory**

- Before interacting with data files, we must ensure they reside in the working directory, which R will by default load data from and save data to.
- There are different ways to change the working directory.
- In RStudio, Files > More > Set Working Directory > Choose Directory
- Better yet, use a function in the console

5: Data Files

```
## setwd("C:/Users/user/OneDrive - kdis.ac.kr/Woosong_2022/Work/2022_fall/DAfM/Lectures")  
## getwd()
```

5: Data Files

- Suppose the United Nations population data are saved as a CSV file, UNpop.csv
- In RStudio: Import Dataset > From Text File....
- Alternatively, we can use the `read.csv()` function.

```
year, world.pop
1950, 2525779
1960, 3026003
1970, 3691173
1980, 4449049
1990, 5320817
2000, 6127700
2010, 6916183
```

5: Data Files

```
UNpop ← read.csv("UNpop.csv")  
class(UNpop)
```

```
## [1] "data.frame"
```

5: Data Files

```
load("UNpop.RData")
```


5: Data Files

- Note that R can access any file on our computer if the full location is specified.
- For example, we can use syntax such as `read.csv("C:/Users/user/OneDrive - kdis.ac.kr/Woosong_2022/Work/2022_fall/DAfM/Lectures/UNpop.csv")` if the data file `UNpop.csv` is stored in the directory.
- However, setting the working directory as shown above allows us to avoid tedious typing.

5: Data Files

- A data frame object is a collection of vectors, but we can think of it like a spreadsheet.
- It is often useful to visually inspect data. We can view a spreadsheet-like representation of data frame objects in RStudio by double-clicking on the object name in the Environment tab in the upper-right window.
- Alternatively, we can use the `view()` function, which as its main argument takes the name of a data frame to be examined.

5: Data Files

```
names(UNpop)
```

```
## [1] "year"      "world.pop"
```

```
nrow(UNpop)
```

```
## [1] 7
```

```
ncol(UNpop)
```

```
## [1] 2
```

```
dim(UNpop)
```

```
## [1] 7 2
```

5: Data Files

```
summary(UNpop)
```

```
##          year      world.pop
##  Min.      :1950  Min.      :2525779
##  1st Qu.:1965  1st Qu.:3358588
##  Median :1980  Median :4449049
##  Mean    :1980  Mean    :4579529
##  3rd Qu.:1995  3rd Qu.:5724258
##  Max.    :2010  Max.    :6916183
```

5: Data Files

```
UNpop$world.pop
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

5: Data Files

- Another way of retrieving individual variables is to use indexing inside square brackets[], as done for a vector.
- Since **a data frame object is a two-dimensional array**, we need two indexes, one for rows and the other for columns.
- Using brackets with a comma [rows, columns] allows users to call specific rows and columns by either row/column numbers or row/column names.
- If we use row/column numbers, sequencing functions covered above, i.e., and `c()`, will be useful.
- If we do not specify a row (column) index, then the syntax will return all rows (columns).

5: Data Files

```
UNpop[, "world.pop"] # extract the column called "world.pop"
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

```
UNpop[c(1, 2, 3),] # extract the first three rows (and all columns)
```

```
##   year world.pop  
## 1 1950   2525779  
## 2 1960   3026003  
## 3 1970   3691173
```

```
UNpop[1:3, "year"] # extract the first three rows of the "year" column
```

```
## [1] 1950 1960 1970
```

5: Data Files

```
## take elements 1, 3, 5, ... of the "world.pop" variable  
UNpop$world.pop[seq(from = 1, to = nrow(UNpop), by = 2)]
```

```
## [1] 2525779 3691173 5320817 6916183
```


5: Data Files

- In R, missing values are represented by NA.
- When applied to an object with missing values, functions may or may not automatically remove those values before performing operations.
- Here, we note that for many functions, like `mean()`, the argument `na.rm = TRUE` will remove missing data before operations occur. I

```
world.pop ← c(UNpop$world.pop, NA)
world.pop
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183      NA
```

```
mean(world.pop)
```

```
## [1] NA
```

```
mean(world.pop, na.rm = TRUE)
```

```
## [1] 4579529
```

6: Saving Objects

- The objects we create in an R session will be temporarily saved in the workspace, which is the current working environment.
- **ls()** function displays the names of all objects currently stored in the workspace.
- In RStudio, all objects in the workspace appear in the Environment tab in the upper-right corner.
- However, these objects will be lost once we terminate the current session.

6: Saving Objects

- This can be avoided if we save the workspace at the end of each session as an RData file.
- When we quit R, we will be asked whether we would like to save the workspace.
- We should answer no to this so that we get into the habit of explicitly saving only what we need.
- If we answer yes, then R will save the entire workspace as .RData in the working directory without an explicit file name and automatically load it next time we launch R.
- **This is not recommended practice**, because the .RData file is invisible to users of many operating systems and R will not tell us what objects are loaded unless we explicitly issue the `ls()` function.

6: Saving Objects

- In RStudio, we can save the workspace by clicking the Save icon in the upper-right Environment window.
- Alternatively, from the navigation bar, click on Session > Save Workspace As..., and then pick a location to save the file.
- To load the same workspace the next time we start RStudio, click the Open File icon in the upper-right Environment window, select Session > Load Workspace..., or use the `load()` function as before.
- It is also possible to save the workspace using the `save.image()` function.
- Unless the full path is specified, objects will be saved to the working directory.

6: Saving Objects

```
## save.image("C:/Users/user/OneDrive - kdis.ac.kr/Woosong_2022/Work/2022_fall/DAfM/Lectures")
```

6: Saving Objects

- It is often better to **save only a specific object** (e.g., a data frame object) rather than the entire workspace.
- This can be done with the `save()` function as in `save(xxx, file = "yyy.RData")`, where xxx is the object name and yyy.RData is the file name.
- Multiple objects can be listed, and they will be stored as a single RData file.

```
## save(UNpop, file = "Session1.RData")  
## save(world.pop, year, file = "Session1.RData")
```

6: Saving Objects

- In other cases, we may want to save a data frame object as a CSV file rather than an RData file.
- We can use the `write.csv()` function by specifying the object name and the file name, as the following example illustrates.

```
## write.csv(UNpop, file = "UNpop.csv")
```

6: Saving Objects

- Finally, to access objects saved in the RData file, simply use the `load()` function as before.

```
## load("Chapter1.RData")
```


In-class activity

- Check your current working directory by running `getwd()`
- Create a folder for this course if you have not already and store all related files there.
- Change your working directory to the new folder location by running `setwd("path_to_your_folder")`
- Download the turnout.csv file to your working directory if you have not already done so.
- Load the csv file to your environment using the `read.csv()` function. Make sure to assign it to an object name
- Try running `head(your_object_name)` and discuss what you see.
- Now save the object as an RData file using the `save()` function

7: Packages

- One of R's strengths is the existence of a large community of R users who contribute various functionalities as R packages.
- These packages are available through the Comprehensive R Archive Network (CRAN; <http://cran.r-project.org>).

7: Packages

- For the purpose of illustration, suppose that we wish to load a data file produced by another statistical software package such as Stata or SPSS.
- The `foreign` package is useful when dealing with files from other statistical software.

```
install.packages("foreign") # install package
```

```
library("foreign") # load package
```

7: Packages

- To use the package, we must load it into the workspace using the `library()` function.
- In some cases, a package needs to be installed before being loaded.
- In RStudio, we can do this by clicking on Packages > Install in the bottomright window , where all currently installed packages are listed, after choosing the desired packages to be installed.
- Alternatively, we can install from the R console using the `install.packages()` function.
- **Package installation needs only to occur once**, though we can update the package later upon the release of a new version (by clicking Update or reinstalling it via the `install.packages()` function).

7: Packages

- Once the package is loaded, we can use the appropriate functions to load the data file.
- For example, the `read.dta()` and `read.spss()` functions can read Stata and SPSS data files, respectively (the following syntax assumes the existence of the `UNpop.dta` and `UNpop.sav` files in the working directory).

```
read.dta("UNpop.dta")  
read.spss("UNpop.sav")
```

7: Packages

- As before, it is also possible to save a data frame object as a data file that can be directly loaded into another statistical software package.
- For example, the `write.dta()` function will save a data frame object as a Stata data file.

```
write.dta(UNpop, file = "UNpop.dta")
```

In-class activity

- Install the `ggplot2` package and run the following lines. What do you see?

```
library(ggplot2)
```

```
ggplot(mpg, aes(displ, hwy, colour = class)) +  
  geom_point()
```

8: Programming and Learning Tips

- We conclude this brief introduction to R by providing several practical tips for learning how to program in the R language.
- First, we should **use a text editor like the one that comes with RStudio** to write our program rather than directly typing it into the R console.
- If we just want to see what a command does, or quickly calculate some quantity, we can go ahead and enter it directly into the R console.
- However, for more involved programming, it is always better to use the text editor and save our code as a text file with the .R file extension.
- This way, we can keep **a record of our program and run it again** whenever necessary.

8: Programming and Learning Tips

- In RStudio, use the pull-down menu File > New File > R Script
- or click the New File icon (a white square with a green circle enclosing a white plus sign) and choose R Script.
- Either approach will open a blank document for text editing in the upper-left window where we can start writing our code.
- To run our code from the RStudio text editor, simply highlight the code and press the Run icon.
 - Alternatively, in Windows, Ctrl+Enter works as a shortcut.
 - The equivalent shortcut for Mac is Command+Enter.

8: Programming and Learning Tips

- Finally, we can also run the entire code in the background (so, the code will not appear in the console) by clicking the Source icon or using the `source()` function with the code file name (including a full path if it is not placed in the working directory) as the input.

8: Programming and Learning Tips

```
source("UNpop.R")
```

8: Programming and Learning Tips

- Second, we can **annotate our R code** so that it can be easily understandable to ourselves and others.
- This is especially important as our code gets more complex.
- To do this, we use the comment character `#`, which tells R to ignore everything that follows it.
- It is customary to use a double comment character `##` if a comment occupies an entire line and use a single comment character `#` if a comment is made within a line after an R command.

8: Programming and Learning Tips

- An example is given here.

```
##  
## File: UNpop.R  
## Author: Iegor Vyshnevskyi  
## The code loads the UN population data and saves it as a STATA file  
##  
library(foreign)  
UNpop ← read.csv("UNpop.csv")  
UNpop$world.pop ← UNpop$world.pop / 1000 # population in millions  
write.dta(UNpop, file = "UNpop.dta")
```

8: Programming and Learning Tips

- Third, for further clarity it is important to **follow a certain set of coding rules**.
- For example, we should use **informative names** for files, variables, and functions.
- **Systematic spacing and indentation** are essential too.
- In today's examples, we place spaces around all binary operators such as <-, =, +, and -, and always add a space after a comma.
- While comprehensive coverage of coding style is beyond the scope of this course, I encourage you to follow a useful R style guide published by Google at <https://google.github.io/styleguide/Rguide.xml>.

8: Programming and Learning Tips

- In addition, it is possible to check our R code for potential errors and incorrect syntax.
- In computer science, this process is called linting.
- The `lintr()` function in the lintr package enables the linting of R code.

8: Programming and Learning Tips

- The following syntax implements the linting of the UNpop.R file shown above, where we replace the assignment operator <- in line 8 with the equality sign = for the sake of illustration.

```
library(lintr)  
lint("UNpop.R")
```


8: Programming and Learning Tips

- Finally, **R Markdown** via the rmarkdown package is useful for quickly writing documents using R.
- R Markdown enables us to easily embed R code and its output within a document using straightforward syntax in a plain-text format.
- The resulting documents can be produced in the form of HTML, PDF, or even Microsoft Word.
- Because R Markdown embeds R code as well as its output, the results of data analysis presented in documents are reproducible.
- R Markdown is also integrated into RStudio, making it possible to produce documents with a single click.
- For a quick start, see <http://rmarkdown.rstudio.com/>.
- later we will have a class devoted to *R Markdown*.

RMarkdown basics

Add `fontsize: 12pt` to the header (the section above `---`)

Basic commands:

`# Section`

`## Subsection`

Formatting: `*italics*` `**boldface**`

Weblink: `[Woosong](https://english.wsu.ac.kr/main/index.jsp)`

Image: ``

RMarkdown

R code:

```
```{r}
```

```
library(foreign)
```

```
```
```

Or just click on the `insert` button to add code

- Then press the `knit` button to see your document

In-class activity

- Install TinyTex by running the codes below:
 - Reference: <https://yihui.org/tinytex/>
 - Do this only once

```
install.packages('tinytex')  
tinytex::install_tinytex()
```

- Create a new RMarkdown document: File > New File > RMarkdown
- If RStudio asks you to download some packages, click yes
- Write a few paragraphs and add this code to the file:
- "Knit"

And that's a wrap!!

Assignment: R script

- Upload your working R script to the system.