ECOM20001 Econometrics 1

Tutorial 1 Semester 1, 2022

Chin Quek

Department of Economics

Welcome

Econometrics 1 is an ***active learning*** subject which means that if you want to succeed in this subject, you will need to ***PARTICIPATE***.

Do NOT fall behind with the content.

Welcome

Econometrics 1 is an ***active learning*** subject which means that if you want to succeed in this subject, you will need to ***PARTICIPATE***.

Do NOT fall behind with the content.

- Before the tutorial, you are expected to watch the pre-recorded lecture videos and complete the necessary readings. There are pre-tutorial tasks for you to complete every week.
- In the tutorial, we will mainly discuss econometric concepts and may sometimes discuss some R commands. You are encouraged to actively ask questions and clarify any doubts. Tutorial attendance and participation contributes to 5% of the subject grade.
- After the tutorial, you are expected to revise the tutorial work. I strongly encourage that you form study groups and study together.
- Strongly recommend that you set up a separate folder for each tutorial containing all the tutorial files, e.g. Tutorial 1, Tutorial 2,. . . , Tutorial 12.
- Each week, you should also complete a short quiz.

Quick poll

- https://flux.qa/4G9BW4

RStudio panels

- During a programming session in R, any variables we define or data we import and save in a dataframe are stored in our **global environment** (think of it as our workspace).

  *These objects appear in the **Environment tab** at the top right of the interface*

- R-Scripting Window:
  - Working with R files that are provided to you, and also R codes that you will develop for your assignments
- Output Window:
  - Most of the output from your R-Scripting Window will be displayed
- Files, Plots, Packages Window:
  - Multi-function window that lists directories and files,
  - Presents graphs that you generate from your R-Scripting Window,
  - Allows you to install additional packages in R,
  - View help files

Working directories

- Method 1: Session > Set Working Directory
- Method 2: Locate the folder in the Files panel > More > Set As Working Directory

```
## Checking the working directory to make sure it is right
getwd()
```

R scripts and comments

While entering and running your code in the `Console` window is effective and simple. However, this technique has its limitations.

- Each time you want to execute a set of commands, you have to re-enter them.
- Complex codes are potentially subject to typographical errors

R scripts are that solution.

An R script is just a bunch of R code (with comments) in a single file, with the file extension '.R'

To start writing a new R script in RStudio, click

- `File -> New File -> R Script`.

To save the R script, click

- `File -> Save`. Give it a name `tute1_R`.

Comments

- The comment character in R is #, anything to the right of a # in a script will be ignored by R.
  - Useful to leave notes and explanations in your scripts.
  - Must include R script with comments in your assignment submissions.

Tips

- The `help()` function and `?` help operator provide access to the documentation pages for R functions, data sets, and other objects in packages
- R is a case-sensitive language. So, variable `x` is not the same as `X`.
- Sometimes, having too many named objects in the global environment creates confusion. To remove all objects, click the **broom icon** at the top of the window.
  - If we want to remove selected objects from the workspace, select the **grid view** from the dropdown menu OR
  - Use the `rm()` function and specify the object(s) for removal

Print Statements - `print()`

We start with the `print()` command, which simply prints either strings or numbers out in the console window.

Example: Printing a string of characters ‘Hello world!’

```
## Print Hello world
print("Hello world!")
```

```
## [1] "Hello world!"
```

Print Statements - `print()`

We start with the `print()` command, which simply prints either strings or numbers out in the console window.

Example: Printing a string of characters 'Hello world!'

```
## Print Hello world
print("Hello world!")
```

```
## [1] "Hello world!"
```

> *How do you print a string of characters 'R says: Hello! How are you?' and a number '20001'?*

```
## [1] "R says: Hello! How are you?"
```

```
## [1] 20001
```

> *Do we need quotations when printing numbers?*

Loading .csv files into R - read.csv()

Data files can easily be loaded from the R session's **working directory**.

A popular data file format is the text file .csv format where columns are separated by a tab, space or comma. These files are "format independent" data files, which basically means it does not matter if you save and open .csv files on a Mac computer, Windows computer, or any other type of computer. They will always work.

The dataset we will work with is named tute1_tutors.csv which should exist in your **working directory**.

```r
## Load the .csv file dataset which creates the dataset with the name data
data = read.csv(file = "tute1_tutors.csv")
```

This creates a dataframe named data (you can use whatever **name** you like to name the dataframe)

Viewing data - `View()`, `names()`, `dim()`, `head()`, `tail()`

To check for potential data entry errors and have a sense of the kind of variables we are working with, how do we view the data?

- You can view the data with the `View()` command - Recall that R is ***case sensitive***.

As an alternative to using the `View()` command, you can just click on `data` in the **Global Environment** window on the far right hand side of your R-Studio window OR the (most) right icon of the dataframe.

```r
## View your dataset data
View(data)
```

instructor	nationality	fav_icecream	fav_number
Marc	Australia	Strawberry	0
Chin	Singapore	Pistachio	21
Silvia	Italy	Vanilla	3
Thao	Australia	Chocolate	7
Huan	China	Cookie and cream	666
Richard	Australia	Vanilla	4a
Thai	Australia	Chocolate	7
Saqib	Australia	Mango	8

- You can also view the names of the variables and dimensions of the dataset with the `names()` and `dim()` commands respectively.

```
## List the names of the variables in your dataset data
names(data)
```

```
## [1] "instructor"   "nationality"  "fav_icecream" "fav_number"
```

```
## Dimensions of your dataset data
dim(data)
```

```
## [1] 8 4
```

Read the first and last n rows of a dataset

There are some times where you required to read large datasets and analyse them. It is really hard to digest a huge dataset which have 20+ columns or even more and have thousands of rows.

- It is always good practice to check whether all the data has been loaded, so you can use the following commands `head()` and `tail()`

```
## Read the first few rows of a dataset
head(data)
```

```
##   instructor nationality     fav_icecream fav_number
## 1       Marc   Australia       Strawberry          0
## 2       Chin   Singapore        Pistachio         21
## 3     Silvia       Italy          Vanilla          3
## 4       Thao   Australia        Chocolate          7
## 5       Huan       China Cookie and cream        666
## 6    Richard   Australia          Vanilla         4a
```

```r
## Read the last few rows of a dataset
tail(data)
```

```
##   instructor nationality     fav_icecream fav_number
## 3     Silvia       Italy          Vanilla          3
## 4       Thao   Australia        Chocolate          7
## 5       Huan       China Cookie and cream        666
## 6    Richard   Australia          Vanilla         4a
## 7       Thai   Australia        Chocolate          7
## 8      Saqib   Australia            Mango          8
```

How do you display the first 3 and last 2 lines of the `data` dataframe?

How do you display the first 3 and last 2 lines of the `data` dataframe?

```
##   instructor nationality fav_icecream fav_number
## 1       Marc   Australia   Strawberry          0
## 2       Chin   Singapore    Pistachio         21
## 3     Silvia       Italy      Vanilla          3
```

```
##   instructor nationality fav_icecream fav_number
## 7       Thai   Australia    Chocolate          7
## 8      Saqib   Australia        Mango          8
```

Determining variable type

You can tell the variable type by using the `sapply()` command and `class` argument.

```
## Variable types in data
sapply(data, class)
```

```
##   instructor  nationality fav_icecream   fav_number
##  "character"  "character"  "character"  "character"
```

You may be surprised that the variable `fav_number` is a character variable!!

Shouldn't `fav_number` be of a numeric type rather than a character type?

Why is `fav_number` a character?

R creates a resulting vector with a mode that can most easily accommodate all the elements it contains. This conversion between modes of storage is called “coercion”. When R converts the mode of storage based on its content, it is referred to as “implicit coercion”.

```
## Computing the mean of fav_number
mean(data$fav_number)
```

```
## Warning in mean.default(data$fav_number): argument is not numeric or log
## returning NA
```

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

```
## This would work
max(data$fav_number)
```

```
## [1] "8"
```

Clearing the Global Environment

If you ever want to clear the dataset in your Environment and start all over, you can use the following command.

```r
## Clear dataset data in the environment window
rm(list = ls())
```

Data manipulation in R

```
## Find Richard's favourite ice cream observation number
which(data$fav_number == "4a")
```

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

```
data$fav_number[6]
```

```
## [1] "4a"
```

How do we change the observation for Richard's favourite number to numeric?

```
## Remove the letter "a" for this observation e.g. [6]
data$fav_number[6] <- 4

sapply(data,class)
```

```
##   instructor  nationality fav_icecream   fav_number
##  "character"  "character"  "character"  "character"
```

Changing class type

```r
# Coerce R into treating fav_number as numeric
data$fav_number <- as.numeric(data$fav_number)

## fav_number is now a numeric variable (vector)
sapply(data, class)
```

```
##   instructor  nationality fav_icecream   fav_number
##  "character"  "character"  "character"    "numeric"
```

Indexing

There are multiple ways to access or replace values in vectors or other data structures. The most common approach is to use "indexing" by using brackets `[ ]`.

Example: Changing the column name from instructor to tutor

```
colnames(data)[1] <- "tutor"
```

Alternate solution provided:

```
names(data)[names(data) == "instructor"] <- "tutor"
```