ETC1010: Introduction to Data Analysis

Week 6, part B

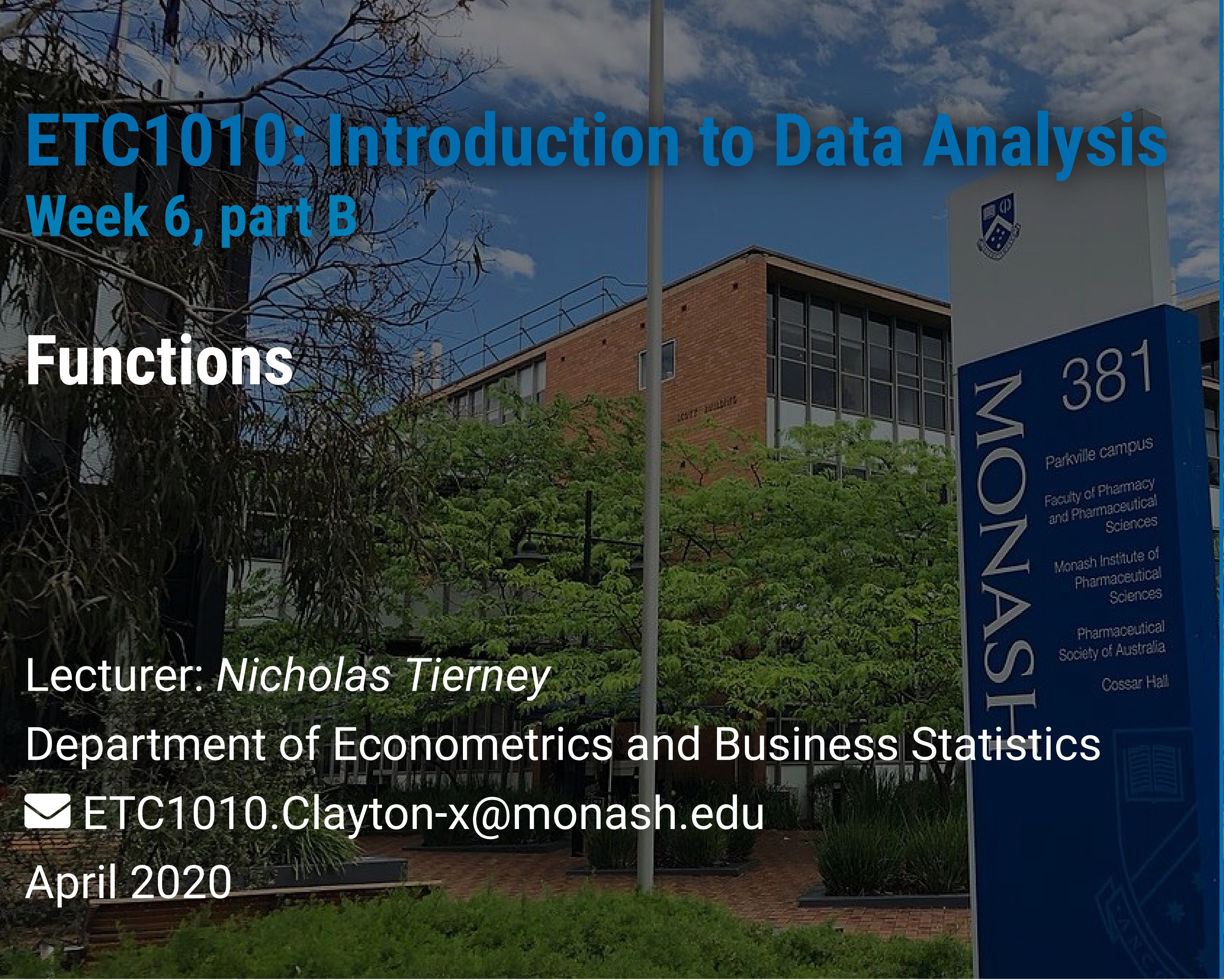
Functions

Lecturer: *Nicholas Tierney*

Department of Econometrics and Business Statistics

✉ ETC1010.Clayton-x@monash.edu

April 2020


MONASH
381
Parkville campus
Faculty of Pharmacy and Pharmaceutical Sciences
Monash Institute of Pharmaceutical Sciences
Pharmaceutical Society of Australia
Cossar Hall

Recap

- File Paths
- functions

Assignment 1

- You all did really well! I'm so proud.
- Sherry marked the exams and had the following to say:

Assignment 1

- Good:
  - People have successfully knitted the .rmd document into an html file that combines all the code, text and R outputs - This is a big achievement for people using R the first time!
  - Most people answer questions well
  - In answering the second last question, we do have 2-3 groups go beyond the question and analyse the extra spreadsheet using tidy verbs and produce some insights - well done those groups!

Assignment 1

- To be improved:
  - Most groups forget to factorise the year in the n_incident vs. year plot. It doesn't make sense to display a year label of "2017.5" in the plot.
- Clarification:
  - In the question "Can you tell me what each row represents, and what each of the columns measure", I give marks to groups that answer "each row represents an observation, ...", but this is a sloppy answer from my view because in our dataset, it is really "each row represents an observation of the number of incidents" rather than "each row represents an observation of one incident". Those who answer the latter lose 0.5 marks.

Assignment 2

- Currently under review from the tutors, this will be released by Friday
- You will all get Two weeks from the release date - this will be updated in the assessment.

Motivating Functions

Remember web scraping?

How many episodes in Stranger Things?

```r
st_episode <-
  bow("https://www.imdb.com/title/tt4574334/") %>%
  scrape() %>%
  html_nodes(".np_right_arrow .bp_sub_heading") %>%
  html_text() %>%
  str_remove(" episodes") %>%
  as.numeric()

st_episode
## [1] 33
```

How many episodes in Stranger Things? And Mindhunter?

```r
st_episode <- bow("https://www.imdb.com/title/tt4574334/") %>%
  scrape() %>%
  html_nodes(".np_right_arrow .bp_sub_heading") %>%
  html_text() %>%
  str_remove(" episodes") %>%
  as.numeric()
st_episode
## [1] 33

mh_episodes <- bow("https://www.imdb.com/title/tt4574334/") %>%
  scrape() %>%
  html_nodes(".np_right_arrow .bp_sub_heading") %>%
  html_text() %>%
  str_remove(" episodes") %>%
  as.numeric()
mh_episodes
## [1] 33
```

Why functions?

- Automate common tasks in a power powerful and general way than copy-and-pasting:
  - Give a functions an evocative name that makes code easier to understand.
  - As requirements change, **you only need to update code in one place, instead of many**.
  - You eliminate the chance of making incidental mistakes when you copy and paste (i.e. updating a variable name in one place, but not in another).

Why functions?

- Down the line: Improve your reach as a data scientist by writing functions (and packages!) that others use

Setup

```r
library(tidyverse)
library(rvest)
library(polite)

st <- bow("http://www.imdb.com/title/tt4574334/") %>% scrape()

twd <- bow("http://www.imdb.com/title/tt1520211/") %>% scrape()

got <- bow("http://www.imdb.com/title/tt0944947/") %>% scrape()
```

When should you write a function?

Whenever you’ve copied and pasted a block of code more than twice.

When you want to clearly express some set of actions

(there are many other reasons as well!)

Do you see any problems in the code below?

```r
st_episode <- st %>%
  html_nodes(".np_right_arrow .bp_sub_heading") %>%
  html_text() %>%
  str_replace(" episodes", "") %>%
  as.numeric()

got_episode <- got %>%
  html_nodes(".np_right_arrow .bp_sub_heading") %>%
  html_text() %>%
  str_replace(" episodes", "") %>%
  as.numeric()

twd_episode <- got %>%
  html_nodes(".np_right_arrow .bp_sub_heading") %>%
  html_text() %>%
  str_replace(" episodes", "") %>%
  as.numeric()
```

Inputs

How many inputs does the following code have?

```r
st_episode <- st %>%
  html_nodes(".np_right_arrow .bp_sub_heading") %>%
  html_text() %>%
  str_replace(" episodes", "") %>%
  as.numeric()
```

Turn the code into a function

Pick a short but informative **name**, preferably a verb, but not always

```
episodes <-
```

Turn your code into a function

- Pick a short but informative **name**, preferably a verb.
- List inputs, or **arguments**, to the function inside `function`. If we had more the call would look like `function(x, y, z)`.

```
episodes <- function(x){



}
```

Turn your code into a function

- Pick a short but informative **name**, preferably a verb.
- List inputs, or **arguments**, to the function inside `function`. If we had more the call would look like `function(x, y, z)`.
- Place the **code** you have developed in body of the function, a { block that immediately follows `function(...)`.

```r
episodes <- function(x){
  x %>%
    html_nodes(".np_right_arrow .bp_sub_heading") %>%
    html_text() %>%
    str_replace(" episodes", "") %>%
    as.numeric()
}
```

Turn your code into a function

```r
episodes <- function(x){
  x %>%
    html_nodes(".np_right_arrow .bp_sub_heading") %>%
    html_text() %>%
    str_replace(" episodes", "") %>%
    as.numeric()
}

episodes(st)
## [1] 33
```

Check your function

- Number of episodes in The Walking Dead

```
episodes(twd)
## [1] 148
```

- Number of episodes in Game of Thrones

```
episodes(got)
## [1] 73
```

Naming functions (it's hard)

> *"There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton*

- Names should be short but clearly evoke what the function does
- Names should be verbs, not nouns
- Multi-word names should be separated by underscores (`snake_case` as opposed to `camelCase`)
- A family of functions should be named similarly (`title`, `episodes`, `genre`, etc.)
- Avoid overwriting existing (especially widely used) functions (e.g., `ggplot`)

Scraping show info

```r
scrape_show_info <- function(x){

  title <- x %>%
    html_node("#title-overview-widget h1") %>%
    html_text() %>%
    str_trim()

  runtime <- x %>%
    html_node("time") %>%
    html_text() %>%
    str_replace("\\n", "") %>%
    str_trim()

  genres <- x %>%
    html_nodes(".txt-block~ .canwrap a") %>%
    html_text() %>%
    str_trim() %>%
    paste(collapse = ", ")

  tibble(title = title, runtime = runtime, genres = genres)
}
```

Scraping show info

```
scrape_show_info(st)
## # A tibble: 1 x 3
##   title           runtime genres
##   <chr>           <chr>   <chr>
## 1 Stranger Things 51min   Drama, Fantasy, Horror, Mystery, Sci-Fi, Thriller
scrape_show_info(twd)
## # A tibble: 1 x 3
##   title            runtime genres
##   <chr>            <chr>   <chr>
## 1 The Walking Dead 44min   Drama, Horror, Thriller
```

How to update this function to use page URL as argument?

```r
scrape_show_info <- function(x){

  title <- x %>% html_node("#title-overview-widget h1") %>%
    html_text() %>%
    str_trim()

  runtime <- x %>% html_node("time") %>%
    html_text() %>%
    str_replace("\\n", "") %>%
    str_trim()

  genres <- x %>% html_nodes(".txt-block~ .canwrap a") %>%
    html_text() %>%
    str_trim() %>%
    paste(collapse = ", ")

  tibble(title = title, runtime = runtime, genres = genres)
}
```

How to update this function to use page URL as argument?

```r
scrape_show_info <- function(x){

y <- bow(x) %>% scrape()

  title <- y %>% html_node("#title-overview-widget h1") %>%
    html_text() %>%
    str_trim()

  runtime <- y %>% html_node("time") %>%
    html_text() %>%
    str_replace("\\n", "") %>%
    str_trim()

  genres <- y %>% html_nodes(".txt-block~ .canwrap a") %>%
    html_text() %>%
    str_trim() %>%
    paste(collapse = ", ")

  tibble(title = title, runtime = runtime, genres = genres)
}
```

Let's check

```
st_url <- "http://www.imdb.com/title/tt4574334/"
twd_url <- "http://www.imdb.com/title/tt1520211/"

scrape_show_info(st_url)
## # A tibble: 1 x 3
##   title           runtime genres
##   <chr>           <chr>   <chr>
## 1 Stranger Things 51min   Drama, Fantasy, Horror, Mystery, Sci-Fi, Thriller
scrape_show_info(twd_url)
## # A tibble: 1 x 3
##   title            runtime genres
##   <chr>            <chr>   <chr>
## 1 The Walking Dead 44min   Drama, Horror, Thriller
```

Automation

Automation

- You now have a function that will scrape the relevant info on shows given its URL.
- Where can we get a list of URLs of top 100 most popular TV shows on IMDB?
- Write the code for doing this in your teams.

Automation

```
urls <- bow("http://www.imdb.com/chart/tvmeter") %>%
  scrape() %>%
  html_nodes(".titleColumn a") %>%
  html_attr("href") %>%
  paste("http://www.imdb.com", ., sep = "")
```

```
##   [1] "http://www.imdb.com/title/tt6468322/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##   [2] "http://www.imdb.com/title/tt5071412/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##   [3] "http://www.imdb.com/title/tt3032476/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##   [4] "http://www.imdb.com/title/tt10293938/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb92
##   [5] "http://www.imdb.com/title/tt6040674/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##   [6] "http://www.imdb.com/title/tt0475784/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##   [7] "http://www.imdb.com/title/tt1439629/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##   [8] "http://www.imdb.com/title/tt12004280/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb92
##   [9] "http://www.imdb.com/title/tt3502248/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##  [10] "http://www.imdb.com/title/tt0944947/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##  [11] "http://www.imdb.com/title/tt0903747/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##  [12] "http://www.imdb.com/title/tt1520211/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##  [13] "http://www.imdb.com/title/tt1796960/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
##  [14] "http://www.imdb.com/title/tt2442560/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=332cb927
```

Automation: Go to each page, scrape show info

- Programatically direct R to each page on the `urls` list and run `scrape_show_info`

```
scrape_show_info(urls[1])
## # A tibble: 1 x 3
##   title        runtime   genres
##   <chr>        <chr>     <chr>
## 1 Money Heist 1h 10min ""
scrape_show_info(urls[2])
## # A tibble: 1 x 3
##   title runtime genres
##   <chr> <chr>   <chr>
## 1 Ozark 1h      Crime, Drama, Thriller
scrape_show_info(urls[3])
## # A tibble: 1 x 3
##   title            runtime genres
##   <chr>            <chr>   <chr>
## 1 Better Call Saul 46min   Crime, Drama
```

Go to each page, scrape show info

In other words, we want to **map** the `scrape_show_info` function to each element of `show_urls`:

```
top_100_shows <- map_df(urls, scrape_show_info)
```

- This will hit the `urls` one after another, and grab the info.

Passing functions to ... functions?

- The fact that we can pass a function to another is a **big idea**, and is one of the things that makes R a **functional programming language**.
- It's a bit mind-bending, but it's an idea worth practicing and comfortable with

aside: `lists` as an idea: first...vectors

- `c()` creates a **vector** of one type
- e.g., `x <- c(1, 2, 3, "A")` contains:
- `[1] "1" "2" "3" "A"`
- `class(x)` returns:
- `[1] "character"`

aside: `lists` as an idea: first...vectors

- You can look up vectors based on position with `[ ]`
- `x[1]` returns the first thing
- `x[2]` returns the second thing
- `x[1:2]` returns the first through to second thing

aside: `lists` as an idea: second...lists

- `list()` creates list, which can be any type

```r
y <- list(1,2,3,"x"); y
#> [[1]]
#> [1] 1
#>
#> [[2]]
#> [1] 2
#>
#> [[3]]
#> [1] 3
#>
#> [[4]]
#> [1] "x"
```

aside: `lists` as an idea: second...lists

- You access positions of a list with `[[]]`
- So `y[[1]]` returns: 1

aside: a `data frame` is actually a list!

calculate the mean for every column:

```
map(mtcars, mean)
## $mpg
## [1] 20.09062
##
## $cyl
## [1] 6.1875
##
## $disp
## [1] 230.7219
##
## $hp
## [1] 146.6875
##
## $drat
## [1] 3.596563
##
## $wt
## [1] 3.21725
##
## $qsec
```

calculate the mean for every column:

```
map_dbl(mtcars, mean)
##        mpg        cyl       disp         hp       drat         wt       qsec 
##  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250  17.848750 
##         vs         am       gear       carb 
##   0.437500   0.406250   3.687500   2.812500
```

Range for every column: writing a function

```
my_range <- function(x){
  max(x) - min(x)
}
map_dbl(mtcars, my_range)
##     mpg     cyl    disp      hp    drat      wt    qsec      vs      am
##  23.500   4.000 400.900 283.000   2.170   3.911   8.400   1.000   1.000
##    gear    carb
##   2.000   7.000
```

Range for every column: writing a function in map

```
map_dbl(mtcars, .f = function(x) max(x) - min(x))
##     mpg     cyl    disp      hp    drat      wt    qsec      vs      am
##  23.500   4.000 400.900 283.000   2.170   3.911   8.400   1.000   1.000
##    gear    carb
##   2.000   7.000
```

Range for every column: writing a function in map

```
map_dbl(mtcars, .f = ~(max(.) - min(.)))
##     mpg     cyl    disp      hp    drat      wt    qsec      vs      am 
##  23.500   4.000 400.900 283.000   2.170   3.911   8.400   1.000   1.000 
##    gear    carb 
##   2.000   7.000
```

Your Turn: rstudio.cloud

Take the lab quiz!

Resources

- Jenny Bryans blog post
- functions chapter of r4DS
- iteration section of r4ds
- lists section in advanced R