07 Don't repeat yourself

Dien Giau (Richard) Bui

4/20/2022

Load library

```
library(dplyr)
library(skimr)
library(stringr)
library(broom)
```

Introduction

- Life is full of repeated routine
  - Repeat calculation or analysis
- This note provides some techniques to iterate/repeat things in a programming way such as:
  - Define your own function to repeat things
  - Loop or mapping

Example

- We may ask if how much is the ranking score of universities across different size
- and for each group size, if the number of international students, faculties, and student-faculty-ratio can affect the `score`
- Load data:

```
data_path = "data/raw/qs-world-university-rankings-2017-to-
Qs = readr::read_csv(data_path)

Qs = Qs %>%
  mutate(research_output = str_replace(research_output, "Ve
Qs = Qs %>% filter(!is.na(research_output))
Qs = Qs %>% filter(!is.na(type))
Qs = Qs %>%
  mutate(private = ifelse(type=="Private", 1, 0))
```

How many size groups?

```
Qs %>% count(size)
```

```
## # A tibble: 4 x 2
##   size      n
##   <chr> <int>
## 1 L      3061
## 2 M      1392
## 3 S       386
## 4 XL     1631
```

First, focus on L size

```
Qs_L = Qs %>% filter(size == "L")
mean(Qs_L$score, na.rm = TRUE)
```

```
## [1] 47
```

```
m1 = lm(score ~ international_students + faculty_count +
          student_faculty_ratio, data=Qs_L)
tidy(m1)
```

```
## # A tibble: 4 x 5
##   term                  estimate std.error statistic
##   <chr>                    <dbl>     <dbl>     <dbl>
## 1 (Intercept)             41.0      1.05        38.9 1
## 2 international_students   0.00273  0.000123    22.1 1
## 3 faculty_count            0.00303  0.000209    14.4 4
## 4 student_faculty_ratio   -0.898    0.0779     -11.5 2
```

Then, we continue with M size

```
Qs_M = Qs %>% filter(size == "M")
mean(Qs_M$score, na.rm = TRUE)
```

```
## [1] 41
```

```
m2 = lm(score ~ international_students + faculty_count +
          student_faculty_ratio, data=Qs_M)
tidy(m2)
```

```
## # A tibble: 4 x 5
##   term                  estimate std.error statistic  p
##   <chr>                    <dbl>     <dbl>     <dbl>
## 1 (Intercept)             27.5      1.97        14.0  7.
## 2 international_students   0.00484  0.000392    12.3  7.
## 3 faculty_count            0.00835  0.000868     9.62 2.
## 4 student_faculty_ratio   -0.171    0.142       -1.21 2.
```

Discussion

- The process to analyze for L and M groups is very similar
  - Filter: one filter "L" and one filter "M"
  - Take mean of `score`: same
  - Run a regression: same formula
  - Clean/tidy the model coefficients: same
- So, actually, they are only different at one input in filter data in the first step

Define own function

- So, we can think about create your own function so that we can repeat this process easily
- A function needs to have:
  - Input: input of the function, in our case, should be the value to filter size, it could receive “L”, “M”, “S”, “XL”
  - Process with this input
  - Return output by the end
- Look like this:

```
function_name = function(Input){
  # PROCESS DATA HERE to get Output

  # RETURN OUTPUT HERE
  return(Output)
}
```

For example

- Write a function to take square of a number

```
calculate_square = function(x){
  x_square = x*x
  return(x_square)
}
calculate_square(4)
```

```
## [1] 16
```

Try to write your function here

- AN EMPTY SLIDE HERE WITH INTENTION

```
analyze_subsize = function(size_type){
  ## YOU CONTINUE HERE
}
```

Apply the function for each group

```
analyze_subsize(size_type = "L")
```

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

```
## # A tibble: 4 x 5
##   term                      estimate std.error statistic
##   <chr>                        <dbl>     <dbl>     <dbl>
## 1 (Intercept)                41.0      1.05          38.9 1
## 2 international_students      0.00273  0.000123      22.1 1
## 3 faculty_count               0.00303  0.000209      14.4 4
## 4 student_faculty_ratio      -0.898    0.0779       -11.5 2
```

```
#analyze_subsize(size_type = "M")
#analyze_subsize(size_type = "S")
#analyze_subsize(size_type = "XL")
```

Discussion

- After having a magic function, we only need to use/apply the function 4 times
  - And if we need later, we also just simply apply more, no need to re-write the whole long process
- If we add more analysis (t-test or so), we simply fix the source code of function
  - So very easy to maintain the analysis process

But . . .

- Do you notice that we aim to avoid repeating things in programming
- So even we now only have a magic function, we still need to repeat 4 times (and even more, e.g., 1,000 groups)

```
#analyze_subsize(size_type = "L")
#analyze_subsize(size_type = "M")
#analyze_subsize(size_type = "S")
#analyze_subsize(size_type = "XL")
```

How to avoid this repeatation?

- In R, we can use some loop/mapping to avoid this kind of repeat
- Let me show you with loop then with mapping later

Use loop to avoid repeat a function many times

```
all_size_type = c("M", "L", "S", "XL")
output = vector(mode = "list", length = 4)
for (i in 1:4){
  output[[i]] = analyze_subsize(size_type = all_size_type[i
}
```

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

See our output vector

```
# for L size
output[[2]]
```

```
## # A tibble: 4 x 5
##   term                      estimate std.error statistic
##   <chr>                        <dbl>     <dbl>     <dbl>
## 1 (Intercept)                41.0      1.05          38.9 1
## 2 international_students      0.00273  0.000123      22.1 1
## 3 faculty_count               0.00303  0.000209      14.4 4
## 4 student_faculty_ratio      -0.898    0.0779       -11.5 2
```

Another way is mapping or apply

- `lapply` is apply and store output as a list

```
output = lapply(X = all_size_type, FUN = analyze_subsize)
```

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

```
# for L size
output[[2]]
```

```
## # A tibble: 4 x 5
##   term                   estimate std.error statistic
##   <chr>                     <dbl>     <dbl>     <dbl>
## 1 (Intercept)             41.0      1.05          38.9 1
## 2 international_students   0.00273  0.000123      22.1 1
## 3 faculty_count            0.00303  0.000209      14.4 4
## 4 student_faculty_ratio   -0.898    0.0779       -11.5 2
```

Thank you

- Thank you for today
- We will have more practices in class