Week-5: Code-along

Tan Ee Xuan

2023-09-13

II. Code to edit and execute using the Code-along.Rmd file

A. Writing a function

1. Write a function to print a “Hello” message (Slide #14)

```
# Enter code here
say_hello_to <- function(name) {
  print(paste0("Hello ", name, "!"))
}
```

2. Function call with different input names (Slide #15)

```
# Enter code here
say_hello_to("Alice")
```

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

```
say_hello_to("Peter")
```

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

3. typeof primitive functions (Slide #16)

```
# Enter code here
typeof(`+`)
```

```
## [1] "builtin"
```

```
typeof(sum)
```

```
## [1] "builtin"
```

4. typeof user-defined functions (Slide #17)

```
# Enter code here
typeof(say_hello_to)
```

```
## [1] "closure"
```

```
typeof(mean)
```

```
## [1] "closure"
```

5. Function to calculate mean of a sample (Slide #19)

```
# Enter code here
calc_sample_mean <- function(sample_size) {
  mean(rnorm(sample_size))
}
```

6. Test your function (Slide #22)

```
# With one input
calc_sample_mean(1000)
```

```
## [1] 0.03663756
```

```
# With vector input
calc_sample_mean(c(12,14,32))
```

```
## [1] -1.235766
```

7. Customizing the function to suit input (Slide #23)

```
# Enter code here
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6     ✔ purrr   0.3.4
## ✔ tibble  3.1.8     ✔ dplyr   1.0.9
## ✔ tidyr   1.2.0     ✔ stringr 1.4.0
## ✔ readr   2.1.2     ✔ forcats 0.5.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
sample_tibble <- tibble(sample_sizes =
                          c(100,300,3000))

sample_tibble %>%
  group_by(sample_sizes) %>%
  mutate(sample_means =
           calc_sample_mean(sample_sizes))
```

```
## # A tibble: 3 × 2
## # Groups:   sample_sizes [3]
##   sample_sizes sample_means
##          <dbl>        <dbl>
## 1          100     -0.00497
## 2          300     -0.0447
## 3         3000      0.00246
```

8. Setting defaults (Slide #25)

```
# First define the function
calc_sample_mean <- function(sample_size,
                             our_mean = 0,
                             our_sd = 1) {
  sample <- rnorm(sample_size,
                  mean = our_mean,
                  sd = our_sd)
  mean(sample)
}
# Call the function
calc_sample_mean(sample_size = 10)
```

```
## [1] -0.4775845
```

9. Different input combinations (Slide #26)

```
# Enter code here
calc_sample_mean(10, our_sd = 2)
```

```
## [1] -0.4374543
```

```
calc_sample_mean(10, our_sd = 6)
```

```
## [1] -1.855392
```

```
calc_sample_mean(10, 6, 2)
```

```
## [1] 6.059555
```

10. Different input combinations (Slide #27)

```
# set error=TRUE to see the error message in the output
# Enter code here
calc_sample_mean(our_mean = 5)
```

```
## Error in rnorm(sample_size, mean = our_mean, sd = our_sd): argument "sample_size"
is missing, with no default
```

11. Some more examples (Slide #28)

```
# Enter code here
add_two <- function(x) {
  x+2
}

add_two(4)
```

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

```
add_two(-34)
```

```
## [1] -32
```

```
add_two(5.684)
```

```
## [1] 7.684
```

B. Scoping

12. Multiple assignment of z (Slide #36)

```
# Enter code here
z <- 1
sprintf("The value assigned to z outside the function is %d", z)
```

```
## [1] "The value assigned to z outside the function is 1"
```

```
foo <- function(z = 2) {
  z <- 3
  return(z+3)
}

foo()
```

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

13. Multiple assignment of z (Slide #37)

```
# Enter code here
z <- 1
foo <- function(z = 2) {
  z <- 3
  return(z+3)
}

foo(z = 4)
```

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