Week-5: Code-along

Wong Wei Qi

2023-09-11

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)

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

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

```
print(say_hello_to('Wei Qi'))
```

```
## [1] "Hello Wei Qi!"
## [1] "Hello Wei Qi!"
```

3. typeof primitive functions (Slide #16)

```
typeof(sum)
```

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

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

```
typeof(say_hello_to)
```

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

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

```
calc_sample_mean <- function(sample_size) {
  mean(rnorm(sample_size))
}
```

6. Test your function (Slide #22)

```
calc_sample_mean(100)
```

```
## [1] 0.09649184
```

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

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.3     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
```

```
# creating a vector to test our function
sample_tibble <- tibble(
  sample_sizes =
    c(100, 300, 3000)
)
# using rowwise groups the data by row, # allowing calc_sample_mean
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     -0.149
## 2          300      0.00653
## 3         3000      0.0103
```

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)
}

calc_sample_mean(100)
```

```
## [1] -0.1126928
```

```
# Call the function
```

9. Different input combinations (Slide #26)

```
calc_sample_mean(10, our_sd = 2)
```

```
## [1] -0.29388
```

10. Different input combinations (Slide #27)

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

```
## Error in calc_sample_mean(our_mean = 5): argument "sample_size" is missing, with n
o default
```

11. Some more examples (Slide #28)

```
add_two <- function(x) { x+2
}
## B. Scoping

add_two(-34)
```

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

```
### 12. Multiple assignment of z (Slide #36)
```

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

13. Multiple assignment of z (Slide #37)

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

foo(4)
```

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