Week-6: Code-along

Sie Choon Hong

2023-09-15

II. Code to edit and execute using the Code-along-6.Rmd file

A. `for` loop

1. Simple `for` loop (Slide #6)

```
# Enter code here
for(x in c(3, 6, 9)) {
  print(x)
}
```

```
## [1] 3
## [1] 6
## [1] 9
```

2. `for` loops structure (Slide #7)

```
# Left-hand side code: for loop for passing values
#for (value in list_of_values) {do something (based on value)}
for (x in 1:8) {
  print(x)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
```

```
# Right-hand side code: for loop for passing indices
# for(index in list_of_indices) {do something (based on index)}

for (x in 1:8) {
  y <- seq(from = 100, to = 200, by = 5)
  print(y[x])
}
```

```
## [1] 100
## [1] 105
## [1] 110
## [1] 115
## [1] 120
## [1] 125
## [1] 130
## [1] 135
```

3. Example: find sample means (Slide #9)

```
# Enter code here
# 1. Determine what to loop over
sample_sizes <- c(5, 10, 15, 20, 25000)

# 2. Pre-allocate space to store output
sample_means <- double(length(sample_sizes))

# 3. Loop over sample sizes and calculate sample means
for (i in seq_along(sample_sizes)) {
  sample_means[i] <- mean(rnorm(sample_sizes[i]))
}

# Print the sample means
sample_means
```

```
## [1] -0.007277533  0.084657261 -0.156170302  0.078277445 -0.009533910
```

4. Alternate ways to pre-allocate space (Slide #12)

```
# Example 3 for data_type=double
sample_means <- rep(0, length(sample_sizes))
```

```
# Initialisation of data_list
data_list <- vector("list", length = 5)
```

5. Review: Vectorized operations (Slide #18)

```
# Example: bad idea!
# Vector with numbers from 7 to 11
a <- 7:11

# Vector with numbers from 8 to 12
b <- 8:12

# Vector of all zeros of length 5
out <- rep(0L, 5)

# Loop along the length of vector a
for (i in seq_along(a)) {
  # Each entry of out is the sum of the corresponding elements of a and b
  out[i] <- a[i] + b[i]
}

out
```

```
## [1] 15 17 19 21 23
```

```
# Taking advantage of vectorization
# Vector with numbers from 7 to 11
a <- 7:11

# Vector with numbers from 8 to 12
b <- 8:12

out <- a + b
out
```

B. Functionals

6. `for` loops vs Functionals (Slides #23 and #24)

```
# Slide 23
# Initialise a vector with the size of 5 different samples
sample_sizes <- c(5, 10, 15, 20, 25000)

# Create a functional- function inside a function
sample_summary <- function(sample_sizes, fun) {

  # Initialise a vector of the same size as sample_sizes
  out <- vector("double", length(sample_sizes))

  # Run the for loop for as long as the length of sample_sizes
  for (i in seq_along(sample_sizes)) {
    # Perform operations indicated fun
    out[i] <- fun(rnorm(sample_sizes[i]))
  }
  return(out)
}
```

```
# Slide 24
#Compute mean
sample_summary(sample_sizes,mean)
```

```
## [1] -0.33491953 -0.36591849  0.60867713  0.02373294 -0.01070351
```

```
# Compute median
sample_summary(sample_sizes,median)
```

```
## [1]  0.250880824 -0.148725728  0.060610847 -0.156596960 -0.008003124
```

```
# Compute sd
sample_summary(sample_sizes,sd)
```

```
## [1] 0.8779618 1.0463170 0.9980815 0.9440599 0.9943535
```

C. `while` loop

7. `while` loop (Slides #27)

```
# Left-hand side code: for loop
# General Structure
for(i in 1:5) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
# Right-hand side code: while loop
# Example
i <- 1
while(i <= 5) {
  print(i)
  i <- i + 1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```