Week-6: Code-along

NM2207: Computational Media Literacy- Tang Ching Xian

13 September 2023

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

A. `for` loop

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

```
for (x in c (3,6,9)) {
  print(x)
}
```

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

```
#action occurs between the curly brackets (print etc.)
```

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

```
# Left-hand side code: for loop for passing values
for (x in 1:8) {print(x)}
```

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

```
# for (value in list_of_values) {do soemthing (based on value)}
```

```
# Right-hand side code: for loop for passing indices
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
```

```
#for (index in list_of_indices){give y variable with specific sequnece and do something eg
print(based on index)}
```

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

```
# Find means of increasingly large samples
#Repeating code in order to find mean
#mean1 <- mean(rnorm(5))
#mean2 <- mean(rnorm(10))
#mean3 <- mean(rnorm(15))
#mean4 <- mean(rnorm(20))
#mean5 <- mean(rnorm(25000))
#print(c(mean1, mean2, mean3, mean4, mean5))

#determine what to loop over
sample_sizes <- c(5,10,15,20,25000)
#pre-allocate space to store output
sample_means <- double(length(sample_sizes))
for (i in seq_along(sample_sizes)){
  sample_means[i] <- mean(rnorm(sample_sizes[i]))
}

sample_means
```

```
## [1] -0.522685188  0.196257403  0.229178859 -0.054861885  0.004042964
```

```
#seq_along(x) is synonymous to 1:length(x)
```

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

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

```r
# Initialisation of data_list
sample_sizes <- c(5, 10, 15, 20, 25000)
# Generate indices using seq_along
##seq_along(sample_sizes)

# Initialize a vector of zeros of the same size as sample_sizes
sample_means <- rep(0, length(sample_sizes))

# # Another way to initialize a vector of zeros is here below
# sample_means <- numeric(length(sample_sizes))
# Loop over each sample size and calculate the mean
for (i in seq_along(sample_sizes)) {
  # Generate a random sample from a normal distribution and calculate its mean
  sample_means[i] <- mean(rnorm(sample_sizes[i]))
}

# Print the calculated sample means
print(sample_means)
```

```
## [1]  0.042534001  0.605541468 -0.465733582  0.023593758 -0.009106214
```

5. Review: Vectorized operations (Slide #18)

```r
# 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 corres
out[i] <- a[i] + b[i]
}
out
```

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

```r
# 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
```

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

B. Functionals

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

```
# 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] -0.437262319  0.325996397 -0.088337364  0.026913014 -0.007932444
```

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

```
## [1]  0.550292456 -0.207610754 -0.472706165 -0.074775887  0.009581361
```

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

```
## [1] 1.5706114 1.3045495 0.8542948 0.8336187 0.9990272
```

C. `while` loop

7. `while` loop (Slides #27)

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

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

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

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