Week-6: Code-along

NM2207: Computational Media Literacy

September 17

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 (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 (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
sample_sizes <- c(5,10,15,20,25000)
sample_means <- double(length(sample_sizes))

for (i in seq_along(sample_sizes)) {
  sample_means[i] <- mean(rnorm(sample_sizes[i]))
}

sample_means
```

```
## [1]  0.43199047  0.49081766 -0.43226022 -0.05286610 -0.00104153
```

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!
a <- 7:11
b <- 8:12
out <- rep(0L, 5)
for (i in seq_along(a)) {
  out[i]<- a[i] + b[i]
}
out
```

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

```
# Taking advantage of vectorization
a <- 7:11
b <- 8:12

out <- a+b
out
```

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

B. Functionals

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

```
# Slide 23
sample_sizes <- c(5,10,15,20,25000)
sample_summary <- function (sample_sizes, fun) {
  out <- vector ("double", length(sample_sizes))
  for (i in seq_along(sample_sizes)) {
    out[i] <- fun(rnorm(sample_sizes[i]))
  }
  return(out)
}
```

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

```
## [1] -0.497145414 -0.555697117 -0.106204417  0.014512398 -0.001025236
```

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

```
## [1]  0.295096158 -0.185361630  0.098283475 -0.056265075 -0.001749612
```

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

```
## [1] 0.5126603 1.0984311 1.0192169 0.9345635 0.9987872
```

C. `while` loop

7. `while` loop (Slides #27)

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

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

```
# 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
```