Week-6: Code-along

NM2207: Computational Media Literacy

2023-09-18

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
# Create a vector of the sample_size we want to use
sample_sizes <- c(5, 10, 15, 20, 25000)

# Initialise output vector with zero, replacing each zero with the mean that we are calculati
ng
sample_means <-double(length(sample_sizes))

# Generate indices as many as the length of the vector (similar to 1:5)
for (i in seq_along(sample_sizes)) {
  sample_means[[i]] <-mean(rnorm(sample_sizes[[i]]))
}
sample_means
```

```
## [1]  0.664808058 -0.162825794 -0.120036176 -0.279163086  0.002094403
```

4. Alternate ways to pre-allocate space (Slide #12) To hold different data types

```
# 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)
fsd <-function(sample_sizes) {
  sample_sds <-rep(0, length(sample_sizes))
  for(i in seq_along(sample_sizes)) {
    sample_sds[i] <-sd(rnorm(sample_sizes[1]))
  }
  return(sample_sds)
}
fsd(sample_sizes)
```

```
## [1] 1.4206798 1.3669406 1.4721028 0.8277743 1.3425477
```

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

```
## [1] -0.708655143 -0.865177504 -0.097899340 -0.198989415  0.003513299
```

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

```
## [1]  0.79860117  0.05372785  0.27269126  0.60038645 -0.01070474
```

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

```
## [1] 0.6483913 1.3390682 0.9657760 0.9959139 0.9951155
```

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
```