

# Week-6: Code-along

NM2207: Computational Media Literacy

2023-09-17

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

```
## [1] 0.411160916 0.148443645 -0.012430508 -0.052104282 -0.002369116
```

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

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

```
## [1] 0 0 0 0 0
```

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

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

## 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 corres  
  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
```

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

## 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] -0.678690585  0.229215141 -0.019293699  0.021704732 -0.003584102
```

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

```
## [1] -0.083862134 -0.399056785  0.048792110 -0.152182925 -0.005503211
```

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

```
## [1] 0.5503773 0.7581978 0.8220235 1.0387453 1.0094225
```

# 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  
i <- 1  
while (i <= 5) {  
  # body  
  print(i)  
  i <- i + 1  
}
```

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