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
#Repetitive execution of instructions
#Instruction outside brackets = repeat the instructions enclosed within the brackets
*for*
#Values that x can take belong to the vector for every iteration

for (x in c(1, 2,7)) {

  print(x)

}
```

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 1:7)
{
  print(x)
}
```

```
# Right-hand side code: for loop for passing indices
#*for* (index in list_of_indices) {do something (based on index)}
#eliciting entries of vector y by indexing ie. x iterations of the index seq y
for (x in 1:7)
{
  y <- seq(from=10, to=100, by=10)
  print(y[x])
}
```

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

```r
# Enter code here
#1. determine what to loop over - aggregate in vector
sample_sizes <- c(5, 10, 20, 5000, 10000)
#2. pre-allocate space to store output - need another empty vector initialized to hol
d vector that is same length as sample_sizes vector [since ratio of sample : mean is
1:1]
sample_means <- double(length(sample_sizes))
#3. seq_along is used to generate indices beginning from 1 as long as the length of t
he vector.
## "i" will take the value of the indice generated.
### the "i"-th number in the vector of sample sizes generated will then be passed thr
ough the function to get its sample mean and be stored in the sample mean vector
for (i in seq_along(sample_sizes)) {

  sample_means[[i]] <- mean(rnorm(sample_sizes[[i]])) }

sample_means
```

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

```r
# Example 3 for data_type=double
#4.1 defining through calling vector and defining double and length

sample_means <- vector("double", length = 5)

#4.2 shorter version, immediately defining double and length
sample_means <- double(5)

#4.3 starting with empty vector instead, defining length by a variable
sample_means <- rep(0, length(sample_sizes))
```

```r
# Initialisation of data_list for data of different types
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
```

```
# Taking advantage of vectorization - two same length and fully defined vectors
a <- 7:11
b <- 8:12

out <- a+b

out
```

B. Functionals

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

```
# Slide 23
sample_sizes <- c(5, 10, 20, 25, 50)
fmed <- function(sample_sizes){

  sample_medians <- rep(0, length(sample_sizes))
  for (i in seq_along(sample_sizes)) {

    sample_medians[i] <- median(rnorm(sample_sizes[i]))
  }
}

sample_sizes <- c(5, 10, 20, 25, 50)
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[i]))
  }
}
```

```
# Slide 24
# Switching out functions with every iteration is possible - the function is passed a
s an argument or input to another (functionals)
#Compute mean
sample_sizes <- c(5, 10, 25, 50, 200)
sample_summary <- function(sample_sizes, fun) {

  output <- vector("double", length(sample_sizes))
  for(i in seq_along(sample_sizes)) {
    output[i] <- fun(rnorm(sample_sizes[i]))
  }
return(output)
}

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

C. `while` loop

“while” loops are used when the length of an input sequence is unknown and is more general than a for loop. execute for as long as a certain condition is satisfied.

structure: while(condition) { body of the while loop}

7. `while` loop (Slides #27)

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

  print(i)

}
```

```
# Right-hand side code: while loop
i <- 1
#'While i is <=5' simply means 'until i reaches 5' --> then the function adds 1 to i
to manually go through the sequence
while (i <= 5) {

  print (i)
  i <- i+1
}
```