PA 434 Week11

1. Loop

```r
output_1 <- vector("numeric", 10)
# since our sequence should be the length of our vector
# we should make the contruction of our loop based on the length directly
for (i in seq(length(output_1))){
  output_1[[i]] <- i + 6
}
output_1
```

```
##  [1]  7  8  9 10 11 12 13 14 15 16
```

2. Matrix

```r
mat_x <- matrix(data = 1:120, nrow = 20, ncol = 6)
```

a.

```r
# create the vector to store our output
output_2a <- vector("numeric", ncol(mat_x))
# get the number of columns of the matrix and use that as the basis for the length of the loop
for (j in seq(ncol(mat_x))) {
  output_2a[[j]] <- sum(mat_x[,j])
}
output_2a
```

```
## [1]  210  610 1010 1410 1810 2210
```

b.

```r
# setting margin to 2 tells the apply function to apply the given function on the columns
output_2b <- apply(X = mat_x, FUN = sum, MARGIN = 2)
output_2b
```

```
## [1]  210  610 1010 1410 1810 2210
```

3. Data Frame

```r
df <- data.frame(1:10, c(letters[1:10]), rnorm(10, sd = 10), stringsAsFactors = FALSE)
# iterate over the columns of the df as objects
# this puts the column in the namespace of each iteration of the loop so that it may be
# treated a normal variable
for (col in df) {
  if (is.numeric(col)) {
    print(mean(col))
  } else if (is.character(col)) {
    print(length(col))
  } else {
    print("column is not numeric or character like")
  }
}
```

```
## [1] 5.5
## [1] 10
## [1] 0.9542233
```

4. Matrix of Distributions

```
mat_distributions <- matrix(nrow = 10, ncol = 4)
means <- c(-10, 0, 10, 100)
# looping over means and mat_distributions in parallel
# with means providing the mean paramenter for the rnorm function
# to generate columns for mat_distributions
for (j in seq(ncol(mat_distributions))) {
  mat_distributions[,j] <- rnorm(10, mean = means[j])
}
mat_distributions
```

```
##             [,1]         [,2]      [,3]      [,4]
##  [1,]  -9.552419  0.013449136  9.558039 100.47536
##  [2,] -11.239835  0.454714438  9.104455 100.11661
##  [3,]  -9.692841 -0.728241290  8.916972  99.50165
##  [4,] -10.642653  0.776399010  9.759161 101.29289
##  [5,] -12.482203 -2.033686290 11.475414 101.82047
##  [6,]  -8.719605  0.099873536  8.321118  99.60163
##  [7,] -10.834658 -1.322112299 10.694659 100.55442
##  [8,] -10.708915 -0.571053236 10.349693  99.45009
##  [9,] -10.889874  0.004890049  9.500734 100.92585
## [10,]  -9.762151  0.259638414 10.055917  98.85339
```

5. Ifelse

```
respondent.df = data.frame(
  name = c("Sue", "Eva", "Henry", "Jan", "Mary", "John"),
  sex = c("f", "f", "m", "m", "f", "m"),
  years = c(21, 31, 29, 19, 23, 33)
)

# create new column and assign values based on sex and years columns
respondent.df$male.teen <-
  ifelse(test = respondent.df$sex == "m" & respondent.df$years < 20,
         yes = 1,
         no = 0)
respondent.df
```

```
##    name sex years male.teen
## 1   Sue   f    21         0
## 2   Eva   f    31         0
## 3 Henry   m    29         0
## 4   Jan   m    19         1
## 5  Mary   f    23         0
## 6  John   m    33         0
```

6. Ifelse versus If_else

```
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 4.0.4
```

```
respondent.df %>%
  mutate(under30 = ifelse(years > 30, NA, years),
         under30.tidy = if_else(years > 30, as.double(NA), years))
```

```
##    name sex years male.teen under30 under30.tidy
## 1   Sue   f    21         0      21           21
## 2   Eva   f    31         0      NA           NA
## 3 Henry   m    29         0      29           29
## 4   Jan   m    19         1      19           19
## 5  Mary   f    23         0      23           23
## 6  John   m    33         0      NA           NA
```

7. tapply mean, min and max

```
for (fun in c("mean", "min", "max")) {
  # print the name of the function that we iterated to
  print(fun)
  # apply function with the name given by variable "fun" across the sexes
  print(tapply(X = respondent.df$years, INDEX = respondent.df$sex, FUN = get(fun)))
}
```

```
## [1] "mean"
##  f  m
## 25 27
## [1] "min"
##  f  m
## 21 19
## [1] "max"
##  f  m
## 31 33
```

8. Tidy

```
author = c(
  "Author1",
  "Author1",
  "Author2",
  "Author3",
  "Author3",
  "Author3",
  "Author4",
  "Author5"
)
pub = c("Pub1", "Pub2", "Pub3", "Pub4",
        "Pub5", "Pub6", "Pub7", "Pub8")
type = c(
  "preprint",
  "article",
  "preprint",
  "article",
  "article",
  "preprint",
  "preprint",
  "article"
)
data <- as_tibble(cbind(author, pub, type))
print(data)
```

```
## # A tibble: 8 x 3
##   author  pub   type
##   <chr>   <chr> <chr>
## 1 Author1 Pub1  preprint
## 2 Author1 Pub2  article
## 3 Author2 Pub3  preprint
## 4 Author3 Pub4  article
## 5 Author3 Pub5  article
## 6 Author3 Pub6  preprint
## 7 Author4 Pub7  preprint
## 8 Author5 Pub8  article
```

```
# split into groups based on author
data.list <- split(data, author)

# Create a loop that will number the publications for each authors.
# "Tidy" the data so that each row represents one author only.
num_pub = vector("numeric", length(data))

for (i in 1:(length(data.list))) {
}
```