STA 445 Assignment 1

Kaylin McLiverty

2023-10-03

Chapter 8 Exercises

Exercise 1

Create a vector of three elements (2,4,6) and name that vector `vec_a`. Create a second vector, `vec_b`, that contains (8,10,12). Add these two vectors together and name the result `vec_c`.

```
vec_a <- c(2,4,6)
vec_b <- c(8,10,12)
vec_c <- vec_a + vec_b
vec_c
```

```
## [1] 10 14 18
```

Exercise 2

Create a vector, named `vec_d`, that contains only two elements (14,20). Add this vector to `vec_a`. What is the result and what do you think R did (look up the recycling rule using Google)? What is the warning message that R gives you?

```
vec_d <- c(14,20)
vec_d + vec_a
```

```
## Warning in vec_d + vec_a: longer object length is not a multiple of shorter
## object length
```

```
## [1] 16 24 20
```

Warning: longer object length is not a multiple of shorter object length[1] 16 24 20

Since the lengths of vec_d and vec_a are different, you are not able to add the two vectors. That is R's warning message

Exercise 3

Next add 5 to the vector vec_a. What is the result and what did R do? Why doesn't in give you a warning message similar to what you saw in the previous problem?

```
5 + vec_a
```

```
## [1]  7  9 11
```

R does not give a warning message similar to the one given through exericse 2. The code added 5 to each element in vec_a. I read online that R will use vectors of size 1 to be recycled to the size of any other vector. That is why the adding 5 to vec_a had no warnings.

Exercise 4

Generate the vector of integers $\{1, 2, \ldots 5\}$ in two different ways.

a) First using the `seq()` function

b) Using the `a:b` shortcut.

```
# part a
seq(1,5,1)
```

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

```
#part b
1:5
```

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

Exercise 5

Generate the vector of even numbers $\{2, 4, 6, \ldots, 20\}$

a) Using the seq() function and

b) Using the a:b shortcut and some subsequent algebra. *Hint: Generate the vector 1-10 and then multiple

```
# part a
seq(2,20,2)
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
# part b
2*c(1:10)
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

Exercise 6

Generate a vector of 21 elements that are evenly placed between 0 and 1 using the `seq()` command and name this vector `x`.

```
x <- seq(0,1, ,21)
x
```

```
##  [1] 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70
## [16] 0.75 0.80 0.85 0.90 0.95 1.00
```

Exercise 7

Generate the vector $\{2, 4, 8, 2, 4, 8, 2, 4, 8\}$ using the `rep()` command to replicate the vector c(2,4,8).

```
rep(c(2,4,8),3)
```

```
## [1] 2 4 8 2 4 8 2 4 8
```

Exercise 8

Generate the vector $\{2, 2, 2, 2, 4, 4, 4, 4, 8, 8, 8, 8\}$ using the `rep()` command. You might need to check the help file for rep() to see all of the options that rep() will accept. In particular, look at the optional argument `each=`.

```
rep(c(2,4,8), each=4)
```

```
##  [1] 2 2 2 2 4 4 4 4 8 8 8 8
```

Exercise 10

In this problem, we will work with the matrix

$$\begin{bmatrix} 2 & 4 & 6 & 8 & 10 \\ 12 & 14 & 16 & 18 & 20 \\ 22 & 24 & 26 & 28 & 30 \end{bmatrix}$$

a) Create the matrix in two ways and save the resulting matrix as `M`.

```
i. Create the matrix using some combination of the `seq()` and `matrix()` commands.

ii. Create the same matrix by some combination of multiple `seq()` commands
    and either the `rbind()` or `cbind()` command.
```

b) Extract the second row out of `M`.

c) Extract the element in the third row and second column of `M`.

```
# part ai
M <- matrix( c(seq(2,30,2)), nrow=3, byrow=TRUE)
M
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    4    6    8   10
## [2,]   12   14   16   18   20
## [3,]   22   24   26   28   30
```

```
# part aii
rbind(seq(2,10,2), seq(12,20,2), seq(22,30,2))
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    4    6    8   10
## [2,]   12   14   16   18   20
## [3,]   22   24   26   28   30
```

```
#part b
M[2,]
```

```
## [1] 12 14 16 18 20
```

```
#part c
M[3,2]
```

```
## [1] 24
```

Exercise 12

The following code creates a `data.frame` and then has two different methods for removing the rows with `NA` values in the column `Grade`. Explain the difference between the two.

```
df <- data.frame(name= c('Alice','Bob','Charlie','Daniel'),
                      Grade = c(6,8,NA,9))

    df[ -which(  is.na(df$Grade) ), ]
```

```
##     name Grade
## 1  Alice     6
## 2    Bob     8
## 4 Daniel     9
```

```
    df[  which( !is.na(df$Grade) ), ]
```

```
##     name Grade
## 1  Alice     6
## 2    Bob     8
## 4 Daniel     9
```

The first method uses a 'which(is.na())' that will return values that are labeled as "na", the use of the '-' then excludes values that are labeled as "na".

The second method uses a '!' with the "is.na" function which is read as "not is.na" and does not include the values that are not available (in other words, includes the values that are defined).

Exercise 14

Create and manipulate a list.

```
a) Create a list named my.test with elements
    + x = c(4,5,6,7,8,9,10)
    + y = c(34,35,41,40,45,47,51)
    + slope = 2.82
    + p.value = 0.000131

b) Extract the second element in the list.

c) Extract the element named `p.value` from the list.
```

```
# Part a
x = c(4,5,6,7,8,9,10)
y = c(34,35,41,40,45,47,51)
slope = 2.82
p.value = 0.000131

my.test <- list(x=x,y=y,slope=slope,p.value=p.value)
str(my.test)
```

```
## List of 4
##  $ x      : num [1:7] 4 5 6 7 8 9 10
##  $ y      : num [1:7] 34 35 41 40 45 47 51
##  $ slope  : num 2.82
##  $ p.value: num 0.000131
```

```
# Part b
my.test[2]
```

```
## $y
## [1] 34 35 41 40 45 47 51
```

```
# Part c
my.test['p.value']
```

```
## $p.value
## [1] 0.000131
```

Chapter 9 Exercises

Exercise 1

Download from GitHub the data file Example_5.xls. Open it in Excel and figure out which sheet of data we should import into R. At the same time figure out how many initial rows need to be skipped. Import the data set into a data frame and show the structure of the imported data using the `str()` command. Make sure that your data has $n = 31$ observations and the three columns are appropriately named. If you make any modifications to the data file, comment on those modifications.

```
data.1 <- read_excel('Example_5.xls', sheet=2, range='A5:C36')
str(data.1)
```

```
## tibble [31 x 3] (S3: tbl_df/tbl/data.frame)
##  $ Girth : num [1:31] 8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ Height: num [1:31] 70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num [1:31] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

Exercise 2

Download from GitHub the data file Example_3.xls. Import the data set into a data frame and show the structure of the imported data using the `tail()` command which shows the last few rows of a data table. Make sure the Tesla values are `NA` where appropriate and that both `-9999` and `NA` are imported as NA values. If you make any modifications to the data file, comment on those modifications.

```
data.2 <- read_excel('Example_3.xls', sheet=2, range='A1:L34', na=c('NA', -9999))

tail(data.2)
```

```
## # A tibble: 6 x 12
##   model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Lotus Europa  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
## 2 Ford Panter~  15.8     8 351     264  4.22  3.17  14.5     0     1     5     4
## 3 Ferrari Dino  19.7     6 145     175  3.62  2.77  15.5     0     1     5     6
## 4 Maserati Bo~  15       8 301     335  3.54  3.57  14.6     0     1     5     8
## 5 Volvo 142E    21.4     4 121     109  4.11  2.78  18.6     1     1     4     2
## 6 Tesla Model~  98      NA  NA     778 NA     4.94  10.4    NA     0     1    NA
```