STA 445 HW2

Nicholas Larson

2/27/24

Problem 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
```

Problem 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_a+vec_d
```

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

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

R warned me that vec_a is three objects long while vec_d is only 2 objects long.

Problem 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?

```
vec_a+5
```

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

It didnt give a warning presumably because R knew I wanted to add 5 to all numbers, but because there were two numbers in the previous problem, R didnt know what to do. R couldnt add two lists/vector that are incompatible with each other.

Problem 4

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

a. First using the `seq()` function

```
seq(from=1, to=5, by=1)
```

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

b. Using the `a:b` shortcut.

```
1:5
```

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

Problem 5

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

a. Using the seq() function

```
seq(from=2, to=20, by=2)
```

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

b. Using the a:b shortcut and some subsequent algebra.

```
y <- 1:10*2
y
```

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

Problem 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(from=0, to=1, length.out=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
```

Problem 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
```

Problem 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
```

Problem 9

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`.

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

```
m <- seq(1:15)*2
matrix(m, ncol=5, nrow=3, byrow=TRUE)
```

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

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

```
r1 <- seq(1:5)*2
r2 <- seq(from=12, to=20, by=2)
r3 <- seq(22, 30, length.out=5)
nmat <- rbind(r1,r2,r3)
nmat
```

```
##    [,1] [,2] [,3] [,4] [,5]
## r1    2    4    6    8   10
## r2   12   14   16   18   20
## r3   22   24   26   28   30
```

b. Extract the second row out of M.

```
nmat[2,]
```

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

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

```
nmat[3,2]
```

```
## r3
## 24
```

Problem 10

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) ), ]

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

In the first strategy, the negation is in the 'which()' function, the code selects the N/A value, then uses everything else. While in the second strategy, the negation is in the 'is.na' function, the code selects all values not N/A, and uses those values selected.

Problem 11

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

```
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(xcoord = x, ycoord = y, slope = slope, p_value = p.value)
my.test
```

```
## $xcoord
## [1]  4  5  6  7  8  9 10
##
## $ycoord
## [1] 34 35 41 40 45 47 51
##
## $slope
## [1] 2.82
##
## $p_value
## [1] 0.000131
```

b. Extract the second element in the list.

```
my.test[2]
```

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

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

```
my.test$p_value
```

```
## [1] 0.000131
```