STA 445 HW2

Breelyn Cocke

February 27th, 2024

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
```

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 gives the warning message "Warning: longer object length is not a multiple of shorter object length[1] 16 24 20". In this situation, R knew that the two vectors could generally not be added due to their length differences. However, it still gives us the result of [16, 24, 20]. By the recycling rule in R, we can assume it added vec_a(1)+vec_d(1) to get 16, vec_a(2)+vec_d(2) to get 24, and vec_a(3)+vec_d(1) to get 20.

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?

```
5+vec_a
```

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

In this problem, we are adding a scalar to the vector, so each component of vec_a gets an addition of 5. This is different because we are not adding different length vectors together, we are adding a scalar to the entire vector. This does not give an error message because each component of vec_a is being manipulated. ## Problem 4

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

a. First using the `seq()` function

```
seq(1, 5)
```

```
## [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(2,20,2)
```

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

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

```
(1:10)*2
```

```
##  [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(0,1,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), times=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 = matrix(seq(2,30,2), nrow = 3, ncol = 5, 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
```

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

```
c1 = seq(2, 10, 2)
c2 = seq(12, 20, 2)
c3 = seq(22, 30, 2)

M = rbind(c1, c2, c3)
```

b. Extract the second row out of M.

```
M[2,]
```

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

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

```
M[3,2]
```

```
## c3
## 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 model, we have a negative sign which tells us we are extracting some value from df. From reading the code, we can tell we are extracting the grade values which have NA = TRUE. We do not want NA values in our data set, so we are getting rid of the ones that are in fact NA. In the second model, we are not necessarily performing an extraction in terms of getting rid of unwanted data, but are looking for the valuable data. The upside down exclamation point serves as a "not" command. So, reading off the code, we are looking for which values in df do not have NA = TRUE. We want the data with values, not the ones without.

Both methods yield the same result, but can be interpreted in different ways. Method 1 focuses on getting rid of data and Method 2 focuses on highlighting wanted data.

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

```
my.list = list(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.list
```

```
## $x
## [1]  4  5  6  7  8  9 10
##
## $y
## [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.list[[2]]
```

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

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

```
my.list$p.value
```

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