STA 445 - Homework #1

Richard McCormick

2023-10-02

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

```r
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?

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

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

```r
vec_d
```

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

The warning message given is: Warning: longer object length is not a multiple of shorter object length.

The vector adds the first two values together normally, but because vec_d is shorter than vec_a, the first term of vec_d is "recycled" and added to the third value in the vector, in order to make the two have equal lengths.

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?

```
vec_a <- vec_a + 5

vec_a
```

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

The result of adding 5 to vec_a is a new vector where all the original values are added to 5. R added 5 to each value in the vector.

There was no warning message for this problem, because a vector can be modified by a constant. In mathematics, it is possible to add, subtract, multiply, or divide a vector or one-dimensional matrix by a constant value, as the constant value is equally applied to all elements of the vector.

Exercise #4

Generate the vector of even numbers {2,4,6,. . . ,20}

a. Using the seq() function and

```
even_nums <- seq( from=2, to=20, by=2 )
even_nums
```

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

b. Using the a:b shortcut and some subsequent algebra. Hint: Generate the vector 1-10 and then multiple it by 2.

```
even_nums <- c(1:10) * 2
even_nums
```

```
##  [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( 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
```

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_vec <- rep( c( 2, 4, 8 ), each=4 )
rep_vec
```

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

Exercise #11

Create and manipulate a data frame.

a. Create a data.frame named my.trees that has the following columns:

Girth = {8.3, 8.6, 8.8, 10.5, 10.7, 10.8, 11.0}

Height= {70, 65, 63, 72, 81, 83, 66}

Volume= {10.3, 10.3, 10.2, 16.4, 18.8, 19.7, 15.6}

```
my.trees <- data.frame( Girth=c( 8.3, 8.6, 8.8, 10.5, 10.7, 10.8, 11.0 ),
                        Height=c( 70, 65, 63, 72, 81, 83, 66 ),
                        Volume=c( 10.3, 10.3, 10.2, 16.4, 18.8, 19.7, 15.6 ) )
```

b. Without using dplyr functions, extract the third observation (i.e. the third row)

```
third_obs <- my.trees[3,]
third_obs
```

```
##   Girth Height Volume
## 3   8.8     63   10.2
```

c. Without using dplyr functions, extract the Girth column referring to it by name (don't use whatever order you placed the columns in).

```
girth_col <- my.trees['Girth']
girth_col
```

```
##   Girth
## 1   8.3
## 2   8.6
## 3   8.8
## 4  10.5
## 5  10.7
## 6  10.8
## 7  11.0
```

d. Without using dplyr functions, print out a data frame of all the observations except for the fourth observation. (i.e. Remove the fourth observation/row.)

```
all_but_four <- my.trees[-4,]
all_but_four
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 5  10.7     81   18.8
## 6  10.8     83   19.7
## 7  11.0     66   15.6
```

e. Without using dplyr functions, use the which() command to create a vector of row indices that have a girth greater than 10. Call that vector index.

```
index <- which( my.trees$Girth > 10, arr.ind=TRUE )
index
```

```
## [1] 4 5 6 7
```

f. Without using dplyr functions, use the index vector to create a small data set with just the large girth trees.

```
small_data <- my.trees[index,]
small_data
```

```
##   Girth Height Volume
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
## 7  11.0     66   15.6
```

g. Without using dplyr functions, use the index vector to create a small data set with just the small girth trees.

```r
smol_data <- my.trees[-index,]
smol_data
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
```

Exercise #13

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 dataframe is using the which command to select all of the rows with NA values, and then keeping only the negation of the which command (i.e., the opposite of all rows which have NA in the Grade column).

The second dataframe is selecting all the rows which do not have NA for the Grade column, and then keeping all of them (using the which command).

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

```
my.test <- 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.test
```

```
## $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.

```
second_val <- my.test[2]

second_val
```

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

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

```
pval <- my.test['p.value']
pval
```

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