STA 445 Assignment #1

Chip Haskins

2024-10-02

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`.

```
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
```

The result is `[1] 16 24 20`, meaning R started repeating the shorter vector when adding it to the longer vector. Specifically, it extended `vec_d` from (14,20) to (14,20,14) to match the length of `vec_a`. The warning message informs us that `vec_a` does not have a length that is a clean multiple of `vec_d`'s length, so the recycling does not repeat `vec_d` completely.

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 + 5
```

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

R created the result, `[1] 7 9 11`, by adding 5 to each element in `vec_a`. This is the same as adding a vector of the same length as `vec_a` which is filled with 5s as each element.

Exercise 5

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

a) Using the `seq()` function and

```
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. *Hint: Generate a sequence of integers then multiple by 2.*

```
1:10 * 2
```

```
##  [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,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()`. 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 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)
)
my.trees
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
## 7  11.0     66   15.6
```

Complete the following without using `dplyr` functions

b) Extract the third observation (i.e. the third row)

```
my.trees[3,]
```

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

c) Extract the Girth column referring to it by name (don't use a numerical value based on column position).

```
my.trees['Girth'] # or my.trees$Girth
```

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

d) Print out a data frame of all the observations *except* for the fourth observation. (i.e. Remove the fourth observation/row.)

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

```
##   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) 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)
index
```

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

f) Use the `index` vector to create a small data set with just the large girth trees.

```
data.frame(my.trees[index,])
```

```
##   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) Use the `index` vector to create a small data set with just the small girth trees.

```
data.frame(my.trees[-index,])
```

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

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

The first method returns `df` **without** the rows that cause the expression `is.na(df$Grade)` to evaluate to `TRUE`, removing the rows with `NA` in the `Grade` column. The second method instead returns `df` **with** the rows that cause the negation of the previous expression (`!is.na(df$Grade)`, note the `!` negation) to evaluate to `TRUE`. This functionally does the same thing, but these methods differ in *how* they remove the desired rows.

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.

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

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

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

```
my.test['p.value'] # or my.test$p.value
```

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