Assignment One

Rebecca Chavez

2023-10-03

Chapter Eight

Question One

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
```

Question Two

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
```

The result is a vector that contains (16, 24, 20). I believe R got this vector by adding the values of vec_a and vec_d together respectively to get 16 and 24. For the last value, since there are no more values in vec_d to add to the last value of vec_a, R reused the first value of vec_d to get $6 + 14 = 20$ for the last value in the new vector. The warning message is saying that the length of vec_a is not a multiple of the length of vec_d, so the values in vec_d will not be reused the same amount of times.

Question Three

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 added 5 to each value in vec_a, resulting in (7, 9, 11). There is no warning message for this because a vector of 5 has a length of 1, which is a multiple of the length of vec_a.

Question Four

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

a) First using the ‘seq()‘ function

```
# seq(x, y) gives a vector of all the whole numbers between x and y
seq(1, 5)
```

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

b) Using the ‘a:b‘ shortcut.

```
# x:y gives a vector of all the whole numbers between x and y
1:5
```

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

Question Five

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

a) Using the seq() function and

```
# the by option gives the multiple that the vector will count by
seq(2, 20, by=2)
```

```
##  [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

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

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

Question Six

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

```r
# the length.out option makes the length of the vector this number
# evenly distributes between the two numbers, using decimals if necessary
x <- seq(0, 1, length.out=21)
```

Question Seven

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

```r
# rep(vector, int) repeats the vector int amount of times
rep(c(2, 4, 8), 3)
```

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

Question Eight

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=`.

```r
# the each option goes through each value in vector and repeats it each times
rep(c(2, 4, 8), each=4)
```

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

Question Ten

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.

```r
# create a matrix of the vector generated by seq()
# give it nrows and fill in rows first
M <- matrix(seq(2, 30, by=2), nrow=3, byrow=TRUE)
```

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

```r
# create a vector for each row
a <- seq(2, 10, by=2)
b <- seq(12, 20, by=2)
c <- seq(22, 30, by=2)

# bind the rows above into one matrix
M <- rbind(a, b, c)
```

b) Extract the second row out of ‘M‘.

```
# matrix[row, col]
M[2,]
```

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

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

```
# matrix[row, col]
M[3, 2]
```

```
##  c
## 24
```

Question Twelve

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.

```r
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 uses the `which()` function to select the rows where there is an ‘NA’ value in the Grade column. It then removes these by ‘-’ in front of the `which()` function, leaving only the rows without an ‘NA’. The second method uses the `which()` function to select rows that do not have an ‘NA’ value in the Grades column by using ‘!’, which means not, in front of the `is.na()` function.

Question Fourteen

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

```
# create list with 2 vectors (x and y) and two doubles (slope, p.value)
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)
```

b) Extract the second element in the list.

```
# the double brackets [[]] gets the number element in the list
my.test[[2]]
```

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

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

```
# putting the element name in quotes will extract the whole element from list
my.test['p.value']
```

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

Chapter Nine

Question One

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.

```
example5 <- read_excel('~/Downloads/Example_5.xls', sheet=2, range='A5:C36')
str(example5)
```

```
## 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 ...
```

Question Two

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.

```
example3 <- read_excel('~/Downloads/Example_3.xls', sheet='data', range='A1:L34', na=c('NA', -9999))
tail(example3)
```

```
## # 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
```