Assignment1-445

2023-10-03

Chapter 8 Exercises

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 # adding vec_a & vec_b
vec_c
```

```
## [1] 10 14 18
```

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? The result added the first and second elements from each vector. But for the third element in 'vec_d' it added the third element to the first element in 'vec_a' & "recycled" it.

Warning: longer object length is not a multiple of shorter object length[1] 16 24 20

```
vec_d <- c(14,20)
vec_ad <- vec_a + vec_d # adding vec_a to vec_d
```

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

```
vec_ad
```

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

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?

result is '[1] 7 9 11'. R added 5 to each element in the 'vec_a'. R doesn't give you a warning message similar to what I saw in the previous problem because 5 is not a vector.

```
vec_a + 5 # adding 5 to each element in vec_a
```

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

4. Generate the vector of integers $\{1, 2, \ldots 5\}$ in two different ways.
   a) First using the `seq()` function

```
seq(1,5) # vector from 1 to 5
```

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

```
b) Using the ‘a:b‘ shortcut.
```

```
1:5 # also vector from 1 to 5
```

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

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

    a) Using the seq() function and

```
seq(2, 20, by=2) # vector from 2 to 20 by 2's
```

```
##  [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*.
```

```
2*(1:10) # also a vector from 2 to 20 by 2's
```

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

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) # 21 numbers with values from 0 to 1 evenly
                              # placed
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
```

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) # vector repeated 3 times
```

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

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
rep(c(2, 4, 8), each = 4) # repeat each element 4 times
```

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

10. 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.

```
M <- matrix( seq(2, 30, by=2), nrow=3, ncol=5)
M # 2 to 30 in intervals of 2 in a 3x5 matrix
```

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

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

```
row_a <- seq(2, 10, by=2)
row_b <- seq(12, 20, by=2)
row_c <- seq(22, 30, by=2)
M2 <- rbind(row_a, row_b, row_c) # bind all rows to create matrix
M2
```

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

```
b) Extract the second row out of `M`.
```

```
M[-(6:10)] # extract second row
```

```
##  [1]  2  4  6  8 10 22 24 26 28 30
```

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

```
M[-c(12)] # extract element in row 3, col 2
```

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

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.

The first line calls for everything to be included in the output except for 'NA', and the second line calls for the opposite of 'NA' to be called… so everything else will be in the output.

```
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) ), ]
```

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

```
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=x, y=y, slope=slope, p.value=p.value) # create list
str(my.test) # show structure of object
```

```
## List of 4
##  $ x      : num [1:7] 4 5 6 7 8 9 10
##  $ y      : num [1:7] 34 35 41 40 45 47 51
##  $ slope  : chr "2.82"
##  $ p.value: chr "0.000131"
```

```
b) Extract the second element in the list.
```

```
my.test[[2]] # extract second element
```

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

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

```
my.test[['p.value']] # extract p.value
```

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

Chapter 9 Exercises

1. 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.

```
# setwd("/Users/ellabuxton/Desktop/STA444")
data.5 <- read_excel( # import data set
  'Example_5.xls', # data set
  sheet='RawData', # selected sheet
  range='A5:C36') # selected range of data on sheet
str(data.5) # display structure of selected data
```

```
## 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 ...
```

2. 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.

```
data.3 <- read_excel( # import data set
  'Example_3.xls', # data set
  sheet='data', # selected sheet
  range='A1:L34', # selected range of data on sheet
  na=c('NA', -9999)) # show 'NA' instead of values in vector
tail(data.3) # display tail of data
```

```
## # 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
```