Assignment 1

Nathan Underwood

2023-10-05

Question 1 (Chapter 8)

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`.

```
vecA <- c(2, 4, 6)
vecB <- c(8, 10, 12)

vecC <-c(vecA, vecB)

vecC
```

```
## [1]  2  4  6  8 10 12
```

Question 2 (Chapter 8)

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?

```
vecD <- c(14, 20)
vecA <- vecA + vecD
```

```
## Warning in vecA + vecD: longer object length is not a multiple of shorter
## object length
```

```
vecA
```

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

R outputs a warning message regarding vector lengths. This is the result of the recycling rule, where if one vector is shorter than the other vector, the shorter vector will wrap back to the first element and continue the addition process. The error is specifically related to the fact that the longer vector is not a multiple of the shorter vector, where the wrap around process would work evenly.

Question 3 (Chapter 8)

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?

```
vecA <- vecA + 5

vecA
```

```
## [1] 21 29 25
```

This addition operation added 5 to each element of the vector. R treats 5 as a single element vector, and wraps around after every operation. There is no error because the vector size of 1 fits in with any other vector size.

Question 4 (Chapter 8)

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

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

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

```
c(1:5)
```

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

Question 5 (Chapter 8)

Generate the vector of even numbers $\{2, 4, 6, \ldots, 20\}$ a) Using the seq() function and b) Using the a:b shortcut and some subsequent algebra. *Hint: Generate the vector 1-10 and then multiple it by 2.*

```
seq(2, 20, 2)
```

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

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

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

Question 6 (Chapter 8)

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
```

Question 7 (Chapter 8)

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)
```

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

Question 8 (Chapter 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
```

Question 10 (Chapter 8)

In this problem, we will work with the matrix

```
\[ \left[\begin{array}{ccccc}
2 & 4 & 6 & 8 & 10\\
12 & 14 & 16 & 18 & 20\\
22 & 24 & 26 & 28 & 30
\end{array}\right]\]

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

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

```r
matrixA <- matrix(seq(2, 30, 2), nrow = 3, ncol = 5, byrow = TRUE)

vecZ = seq(2, 10, 2)
vecY = seq(12, 20, 2)
vecX = seq(22, 30, 2)
matrixB <- rbind(vecZ, vecY, vecX)

matrixB[2, ]
```

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

```r
matrixB[[3, 2]]
```

```
## [1] 24
```

Question 12 (Chapter 8)

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) ), ]
```

```
##     name Grade
## 1  Alice     6
## 2    Bob     8
## 4 Daniel     9
```

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

```
##     name Grade
## 1  Alice     6
## 2    Bob     8
## 4 Daniel     9
```

The first method uses the '-' sign to specify which specific elements are to be indexed – in this case, all of the elements that are not NA. The second method specifies the type of argument to be considered, rather than the indices to be evaluated. '!' is the NOT symbol.

Question 14 (Chapter 8)

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

list <- list(x = x, y = y, Slope = slope, PValue = p.value)

list[[2]]
```

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

```
list[['PValue']]
```

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

Question 1 (Chapter 9)

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.

```
excelData5 <- read_excel('Example_5.xls', sheet='RawData',
                         range = 'A5:C36', col_names = TRUE)

str(excelData5)
```

```
## 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 2 (Chapter 9)

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.

```
excelData3 <- read_excel('Example_3.xls', sheet='data',
                         range = 'A1:L34', na = c('NA', '-9999'))

tail(excelData3)
```

```
## # 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
```