STA 141A - Fall 2024 - Homework 1

Instructor: Dr. Akira Horiguchi

Student name: Andrew Jowe; Student ID: 919586453

Due date: Oct 04, 2024 at 07:59 PM (PT)

The assignment must be done in an R Markdown document. The assignment must be submitted by the due date above by uploading two files:

1. a .rmd or .qmd source file in CANVAS;
2. a .pdf file in GRADESCOPE (if you can knit/compile your .rmd to a .html file only, please save the created .html file as a .pdf file (by opening the .html file -> print -> save to .pdf)).

Email submissions will not be accepted.

Each answer has to be based on `R` code that shows how the result was obtained. The code has to answer the question or solve the task. For example, if you are asked to find the largest entry of a vector, the code has to return the largest element of the vector. If the code just prints all values of the vector, and you determine the largest element by hand, this will not be accepted as an answer. No points will be given for answers that are not based on `R`. This homework already contains chunks for your solution (you can also create additional chunks for each solution if needed, but it must be clear to which tasks your chunks belong).

There are many possible ways to write `R` code that is needed to answer the questions or do the tasks, but for some of the questions or tasks you might have to use something that has not been discussed during the lectures or the discussion sessions. You will have to come up with a solution on your own. Try to understand what you need to do to complete the task or to answer the question, feel free to search the Internet for possible solutions, and discuss possible solutions with other students. It is perfectly fine to ask what kind of an approach or a function other students use. However, you are not allowed to share your code or your answers with other students. Everyone has to write the code, do the tasks and answer the questions on their own.

During the discussion sessions, you may be asked to present and share your solutions.

Good luck!

1. Vectors and lists (7 points)

Suppose that we have:

- four types of animals: `cat`, `dog`, `cow`, `squirrel`;
- four possible colors: `white`, `black`, `brown`, `red`;
- five possible attributes: `big`, `small`, `angry`, `cute`, `finicky`.

a) (2 points) Generate three random samples of size `50` from each of the three groups, so that you have a vector containing 50 animals, a vector containing 50 colors and a vector containing 50 attributes. Call the resulting vectors of character strings as: `Animal`, `Color`, `Attribute`.

```
# Define categories
animals <- c("cat", "dog", "cow", "squirrel")
colors <- c("white", "black", "brown", "red")
attributes <- c("big", "small", "angry", "cute", "finicky")

# Generate samples
set.seed(123)
Animal <- sample(animals, 50, replace = TRUE)
Color <- sample(colors, 50, replace = TRUE)
Attribute <- sample(attributes, 50, replace = TRUE)

print(Animal)
```

```
##  [1] "cow"      "cow"      "cow"      "dog"      "cow"      "dog"
##  [7] "dog"      "dog"      "cow"      "cat"      "squirrel" "dog"
## [13] "dog"      "cat"      "dog"      "cow"      "squirrel" "cat"
## [19] "cow"      "cow"      "cat"      "squirrel" "cat"      "cat"
## [25] "cat"      "cow"      "squirrel" "dog"      "cow"      "dog"
## [31] "cat"      "dog"      "cow"      "squirrel" "dog"      "cat"
## [37] "cow"      "cow"      "cat"      "squirrel" "cow"      "squirrel"
## [43] "dog"      "cat"      "cow"      "cat"      "cat"      "dog"
## [49] "cow"      "cow"
```

```
print(Color)
```

```
##  [1] "red"   "white" "brown" "white" "brown" "red"   "black" "white" "black"
## [10] "white" "white" "red"   "red"   "brown" "white" "black" "white" "white"
## [19] "brown" "white" "black" "white" "brown" "white" "brown" "black" "red"
## [28] "brown" "red"   "red"   "black" "black" "brown" "red"   "black" "black"
## [37] "brown" "brown" "red"   "white" "black" "black" "white" "black" "red"
## [46] "white" "white" "black" "brown" "brown"
```

```
print(Attribute)
```

```
##  [1] "big"     "cute"    "big"     "big"     "angry"   "cute"    "big"
##  [8] "angry"   "finicky" "angry"   "small"   "finicky" "finicky" "angry"
## [15] "small"   "small"   "small"   "cute"    "small"   "small"   "cute"
## [22] "cute"    "big"     "angry"   "angry"   "big"     "angry"   "finicky"
## [29] "small"   "angry"   "small"   "finicky" "finicky" "angry"   "cute"
## [36] "cute"    "cute"    "finicky" "angry"   "big"     "small"   "big"
## [43] "small"   "finicky" "angry"   "cute"    "cute"    "big"     "cute"
## [50] "big"
```

b) (1 point) By using the `sum()` function and logical operations, compute the number of animals that are cats or dogs.

```
cat_or_dog_count <- sum(Animal == "cat" | Animal == "dog")
cat_or_dog_count
```

```
## [1] 26
```

c) (1 point) Compute the relative frequency of cats, dogs, cows and squirrels in the sample.

```
animal_table <- table(Animal)
relative_frequency <- animal_table / length(Animal)
relative_frequency
```

```
## Animal
##      cat      cow      dog squirrel
##     0.26     0.34     0.26     0.14
```

d) (1 point) Create a contingency table between `Animal` and `Attribute`.

```
table(Animal, Attribute)
```

```
##           Attribute
## Animal     angry big cute finicky small
##   cat          5   1    5       1     1
##   cow          2   4    3       3     5
##   dog          2   3    2       4     2
##   squirrel     2   2    1       0     2
```

e) (2 points) Put the three vectors together in a list of three elements called `mylist`, so that each vector is an element of the list. Use the command `length(mylist[1])` to print the length of the first vector. Is this code actually printing the length of the vector? Explain, and write the correct code to print the length of the first vector of the list.

```
# Create list
mylist <- list(Animal = Animal, Color = Color, Attribute = Attribute)

# Incorrect code
# mylist[1] is a sublist containing the first vector
# Hence, the length is 1
length(mylist[1])
```

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

```
# Correct code
# This returns the length of the first vector
length(mylist[[1]])
```

```
## [1] 50
```

```
mylist
```

```
## $Animal
##  [1] "cow"      "cow"      "cow"      "dog"      "cow"      "dog"
##  [7] "dog"      "dog"      "cow"      "cat"      "squirrel" "dog"
## [13] "dog"      "cat"      "dog"      "cow"      "squirrel" "cat"
```

```
## [19] "cow"      "cow"      "cat"      "squirrel" "cat"      "cat"
## [25] "cat"      "cow"      "squirrel" "dog"      "cow"      "dog"
## [31] "cat"      "dog"      "cow"      "squirrel" "dog"      "cat"
## [37] "cow"      "cow"      "cat"      "squirrel" "cow"      "squirrel"
## [43] "dog"      "cat"      "cow"      "cat"      "cat"      "dog"
## [49] "cow"      "cow"
##
## $Color
##  [1] "red"   "white" "brown" "white" "brown" "red"   "black" "white" "black"
## [10] "white" "white" "red"   "red"   "brown" "white" "black" "white" "white"
## [19] "brown" "white" "black" "white" "brown" "white" "brown" "black" "red"
## [28] "brown" "red"   "red"   "black" "black" "brown" "red"   "black" "black"
## [37] "brown" "brown" "red"   "white" "black" "black" "white" "black" "red"
## [46] "white" "white" "black" "brown" "brown"
##
## $Attribute
##  [1] "big"     "cute"    "big"     "big"     "angry"   "cute"    "big"
##  [8] "angry"   "finicky" "angry"   "small"   "finicky" "finicky" "angry"
## [15] "small"   "small"   "small"   "cute"    "small"   "small"   "cute"
## [22] "cute"    "big"     "angry"   "angry"   "big"     "angry"   "finicky"
## [29] "small"   "angry"   "small"   "finicky" "finicky" "angry"   "cute"
## [36] "cute"    "cute"    "finicky" "angry"   "big"     "small"   "big"
## [43] "small"   "finicky" "angry"   "cute"    "cute"    "big"     "cute"
## [50] "big"
```

2. Data frames (4 points)

a) (2 points) Create a data frame `df` with one numeric vector `indices` and two character vectors `firstNames` and `surnames` with length 5, respectively. The vector `indices` consists of randomly chosen values from 1 to 10 (the same values are NOT allowed to appear several times). The entries of `firstNames` are randomly chosen from the names 'Alex', 'Peter', 'Tom' (the same values are allowed to appear several times), and the entries of `surnames` are randomly chosen from the names 'James' and 'Miller' (the same values are allowed to appear several times).

```r
set.seed(123)

# Create columns
indices <- sample(1:10, 5, replace = FALSE)
firstNames <- sample(c("Alex", "Peter", "Tom"), 5, replace = TRUE)
surnames <- sample(c("James", "Miller"), 5, replace = TRUE)

# Create df
df <- data.frame(indices = indices, firstNames = firstNames, surnames = surnames)

df
```

```
##   indices firstNames surnames
## 1       3        Tom   Miller
## 2      10       Alex    James
## 3       2      Peter   Miller
## 4       8      Peter    James
## 5       6       Alex    James
```

b) (1 point) Select all elements of the data frame `df`, where the value of `indices` is greater than 7. Also, select all elements of the data frame `df`, where `firstNames` is 'Alex'.

```r
print(subset(df, indices > 7))
```

```
##   indices firstNames surnames
## 2      10       Alex    James
## 4       8      Peter    James
```

```r
print(subset(df, firstNames == "Alex"))
```

```
##   indices firstNames surnames
## 2      10       Alex    James
## 5       6       Alex    James
```

c) (1 point) Select all elements of the data frame `df`, where both the value of `firstNames` is 'Tom', and the `surnames` is 'Miller'.

```r
subset(df, firstNames == "Tom" & surnames == "Miller")
```

```
##   indices firstNames surnames
## 1       3        Tom   Miller
```

3. Matrices (7 points)

a) (2 points) Create two matrices `A`, `B`, with two rows and two columns each, where `A` contains the values 1 to 4, and `B` the values 5 to 8. Further, create a vector consisting of 6 elements which are randomly chosen from the values 1 to 100 (the same values are allowed to appear several times). Based on this vector, define a matrix `C` with 2 rows and 3 columns.

```
# Create matrices
A <- matrix(1:4, nrow = 2, ncol = 2)
B <- matrix(5:8, nrow = 2, ncol = 2)

# Create vector
set.seed(123)
vec <- sample(1:100, 6, replace = TRUE)

# Create matrix from vector
C <- matrix(vec, nrow = 2, ncol = 3)

print(A)
```

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

```
print(B)
```

```
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```
print(C)
```

```
##      [,1] [,2] [,3]
## [1,]   31   51   67
## [2,]   79   14   42
```

b) (1 point) Return the matrices which are obtained by element-wise addition (`+`) and multiplication (`*`) of the matrices `A` and `B`.

```
print(A + B)
```

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

```
print(A * B)
```

```
##      [,1] [,2]
## [1,]    5   21
## [2,]   12   32
```

c) (1 point) Determine the matrix produts of `A` and `B`, and `B` and `C`.

```
print(A %*% B)
```

```
##      [,1] [,2]
## [1,]   23   31
```

```
## [2,]   34   46
```

```
print(B %*% C)
```

```
##      [,1] [,2] [,3]
## [1,]  708  353  629
## [2,]  818  418  738
```

d) (2 points) Return the values in the 1st row of `A`, in the 2nd row of `B`, in the 1st column of `C`, and in the 1st row and 2nd column of `C`.

```
print(A[1, ])
```

```
## [1] 1 3
```

```
print(B[2, ])
```

```
## [1] 6 8
```

```
print(C[, 1])
```

```
## [1] 31 79
```

```
print(C[1, 2])
```

```
## [1] 51
```

e) (1 point) Use the `apply()` function to determine the row-wise sum of `C`.

```
apply(C, 1, sum)
```

```
## [1] 149 135
```

4. System of linear equations (2 points + 1 Bonus point)

Consider the following system of linear equations

$$\begin{aligned} x_1 + 2x_2 + 3x_3 + 2x_4 &= 3, \\ 2x_1 + x_2 + 2x_3 + 7x_4 &= -1, \\ x_1 + 2x_2 + x_3 + 2x_4 &= 0, \\ -x_1 + 3x_2 + 2x_3 + x_4 &= 6. \end{aligned}$$

a) (2 points) Create the matrix `A` and the vector `y` corresponding to the matrix equation $Ax = y$, where $A \in \mathbb{R}^{4\times4}$ and $x, y \in \mathbb{R}^4$ (Hint: it might be useful to make use of the commands `cbind()` or `rbind()`).

```
# Create matrix
A <- rbind(
  c(1, 2, 3, 2),
  c(2, 1, 2, 7),
  c(1, 2, 1, 2),
  c(-1, 3, 2, 1)
)

# Create vector
y <- c(3, -1, 0, 6)

print(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    2
## [2,]    2    1    2    7
## [3,]    1    2    1    2
## [4,]   -1    3    2    1
```

```
print(y)
```

```
## [1]  3 -1  0  6
```

b)* (1 Bonus point) Find the solution of the system of the linear equations by using the `solve()` function.

The solution of $Ax = y$ is

```
solve(A, y) # x_1, x_2, x_3, x_4
```

```
## [1] -2.08333333  0.31250000  1.50000000 -0.02083333
```