STA 141A - Midterm exam 1

Instructor: Akira Horiguchi

Date: Apr 25, 2025, 01:20 – 02:00 PM (PT)

Instructions: This midterm exam is a closed-book exam, it is scheduled for 40 minutes and written in class. Except for a pen/pencil and a two-sided cheat sheet, no other materials are allowed (unless you have SDC accommodations that I have already agreed to). This exam is out of 100 points. Make sure your name and your student ID is written on the first page.

For questions that ask you to write code to perform calculations, please do not simply hard code these calculations. For example, if I ask you to write code to compute the sum of a vector `x <- c(4,5,6)`, do not simply write `4+5+6`. Instead, write something like `sum(x)`. You many also assume that all R packages we have used in class and in homeworks are installed and loaded.

Name:

Student ID:

Score

1.:

2.:

3.:

4.:

5.:

Total:

Q1.

(a) What is the result of the following R code?

```r
cbind(c(6,4,-9), c(3,-2,7))
```

Your Solution:

(b) Consider the following data frame:

```r
df
```

```
  x  y name
1 3 -1  Bug
2 4  0  Cat
3 5  1  Bug
4 6  2  Ant
```

What is the result of the following R code?

```r
df[df$y==2, ]$y
```

Your Solution:

(c) What is the result of the following R code? (Here `df` is the same as in part (c) above.)

```r
df[df$y==2 | df$name=="Bug", ]$y
```

Your Solution:

Q2.

(a) What does the following R code do?

```
x <- sample(c("f", "g", "e", NA), size=10, replace=TRUE)
```

Your Solution:

(b) Using `if`, `else if`, and `else`, write R code that prints `"ant"` if the number of "f"s in `x` is less than 5, prints `"bug"` if the number of "f"s in `x` is greater than 5, and prints `"neither"` otherwise.

- Here `x` is the vector created in part (a).

Your Solution:

(c) Using the function `if_else()`, write R code that returns a vector of the same length as `x` that contains `"fox"` if the corresponding element of `x` is "f", contains `"goat or ewe"` if the corresponding element of `x` is "g" or "e", and contains `"yuck"` if the corresponding element of `x` is `NA`. (Here `x` is the vector created in part (a).)

Your Solution:

(d) Using a `for` loop, write R code that returns the value of $-1 + 2\sum_{k=1}^{5} 4^k$.

Your Solution:

Q3.

(a) Using `flist` below and the `sapply()` function, write R code that returns a numeric vector (of length 4) containing the mean and variance of the numeric vector `x` below.

Your Solution:

(b) Write a function called `personColor` whose arguments are the name (default name shall be "Kate") and favorite color (default color shall be "green") of a person, and returns the string "`name` likes the color `color`!" for a given `name` and `color` (see examples below). *Hint*: The function `paste0()` returns the following for the strings "a", "b" and "c!":

```
paste0("a ", "b ", "c!")
```

```
[1] "a b c!"
```

```
personColor("Sam", "yellow")
```

```
[1] "Sam likes the color yellow!"
```

```
personColor("Ant", "magenta")
```

```
[1] "Ant likes the color magenta!"
```

Your Solution:

(c) Describe the plot created by the following `R` code. (A thorough explanation of the single aesthetics and how the stated functions work is not necessary).

```
library(ggplot2)
ggplot(data = mpg, mapping = aes(x=displ, y=hwy, color=class)) +
  geom_point() +
  geom_smooth(method = 'lm', color='black')
```

Your Solution:

Q4.

Consider the following data frame:

```
scores
```

```
  id math english
1  a   30      85
2  b   80      95
3  c   70      45
```

For the following, the data frames you draw must include column names and values, but do not need to include data types.

(a) Draw the data frame that results from the following R code.

```
scores |>
  filter(math > 75) |>
  select(id, english)
```

Your Solution:

Q5.

For this problem we will use the `flights` data set, where we recall that each row represents a flight. For your convenience, below we show the structure of `flights` using `str()`.

```r
str(flights)
```

```
tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
 $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
 $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
 $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
 $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
 $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
 $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
 $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
 $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
 $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
 $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
 $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
 $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
 $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
 $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
 $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
 $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
 $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
 $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

Make sure you understand examples in the "Complex calculations" section of slide deck 03.