STA 141A - Fall 2024 - Homework 3

Instructor: Dr. Akira Horiguchi

Student name: ABCDE FGHIJ; Student ID: 123456789

Due date: Oct 18, 2024 at 07:59 PM (PT)

The assignment has to be done in an R Markdown document. The assignment has to be submitted electronically on Canvas 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. Creating Functions (12 points in total)

This question will test your understanding of certain built-in `R` functions and to create custom functions. You might also need `Sys.time()` and loops.

The Fibonacci sequence is a well known sequence in mathematics which is defined as follows:

Recursive definition: $F_n = F_{n-1} + F_{n-2}$;

Explicit Definition: $F_n = (F_2 - \frac{F_1}{2})\frac{\phi^{n-1} - (1-\phi)^{n-1}}{2\phi - 1} + \frac{F_1}{2}(\phi^{n-1} + (1-\phi)^{n-1})$, where $\phi = \frac{1+\sqrt{5}}{2}$.

a) (8 points (3+3+2[4*0.5])) Write both functions:

i) Write the recursively defined Fibonacci sequence as a function which takes the number of terms n, the 1st term F_1 and the 2nd term F_2 as input, and which returns up to the nth term F_n of the entire Fibonacci sequence. By default, $F_1 = F_2 = 1$.

```
fibonacci_recursive <- function(n, f1 = 1, f2 = 1) {
  if (n == 1) {
    return(c(f1))
  } else if (n == 2) {
    return(c(f1, f2))
  }

  prev_terms <- fibonacci_recursive(n - 1, f1, f2)
  last_two_terms <- prev_terms[(length(prev_terms) - 1):length(prev_terms)]

  return(c(prev_terms, sum(last_two_terms)))
}
```

ii) Write the explicitly defined Fibonacci sequence as a function with the same inputs, returns and defaults in i).

```
fibonacci_explicit <- function(n, f1 = 1, f2 = 1) {
  if (n == 1) {
    return(f1)
  } else if (n == 2) {
    return(f2)
  }

  phi <- (1 + sqrt(5)) / 2
  term1 <- ((f2 - f1 / 2)
            * (phi ** (n - 1) - (1 - phi) ** (n - 1))
            / (2 * phi - 1))
  term2 <- f1 / 2 * (phi ** (n - 1) + (1 - phi) ** (n - 1))
  return(term1 + term2)
}
```

iii) Confirm both functions work for multiple different inputs. Specifically, test the following by printing each of their outputs for each of the following inputs:

i) $n = 5, F_1 = 1, F_2 = 2$;

ii) $n = 10, F_1 = 20, F_2 = -12$;

iii) $n = 25$ and unspecified F_1, F_2;

iv) $n = 30, F_1 = -1$ and unspecified F_2.

```
# i) n = 5, F_1 = 1, F_2 = 2
cat("Recursive: ", fibonacci_recursive(5, 1, 2), "\n")
```

```
## Recursive:  1 2 3 5 8
```

```
cat("Explicit:  ", fibonacci_explicit(5, 1, 2), "\n\n")
```

```
## Explicit:   8
```

```
# ii) n = 10, F_1 = 20, F_2 = -12
cat("Recursive: ", fibonacci_recursive(10, 20, -12), "\n")
```

```
## Recursive:  20 -12 8 -4 4 0 4 4 8 12
```

```
cat("Explicit:  ", fibonacci_explicit(10, 20, -12), "\n\n")
```

```
## Explicit:   12
```

```
# iii) n = 25 with default F_1, F_2
cat("Recursive: ", fibonacci_recursive(25), "\n")
```

```
## Recursive:  1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 463
```

```
cat("Explicit:  ", fibonacci_explicit(25), "\n\n")
```

```
## Explicit:   75025
```

```
# iv) n = 30, F_1 = -1 with unspecified F_2 (default F_2 = 1)
cat("Recursive: ", fibonacci_recursive(30, -1), "\n")
```

```
## Recursive:  -1 1 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28
```

```
cat("Explicit:  ", fibonacci_explicit(30, -1), "\n")
```

```
## Explicit:   196418
```

b) (4 points (3+1)) Compare the two functions:

i) Call the recursively and explicitly defined functions in a) i)-ii) 30 times and compare which one runs faster.

```
# Test with n = 20
n <- 20

# This one runs slower
system.time({
  for (i in 1:5000) {
    fibonacci_recursive(n)
  }
})
```

```
##    user  system elapsed
##   0.073   0.003   0.076
```

```r
# This one runs faster
system.time({
  for (i in 1:5000) {
    fibonacci_explicit(n)
  }
})
```

```
##    user  system elapsed
##   0.003   0.000   0.003
```

ii) Explain which algorithm is more precise.

The recursive algorithm is more precise because it's the exact calculation compared to the explicit formula which approximates the answer using irrational numbers

2. Using `ggplot()` (8 points in total)

a) (4 points) Load the built-in `mtcars` dataset from `R`. By using `ggplot2`, create a scatterplot with horsepower on the x-axis, and `mpg` on the y-axis, separated by the number of cylinders. Make sure the following properties are met:

i) Each point is colored based on the number of cylinders;

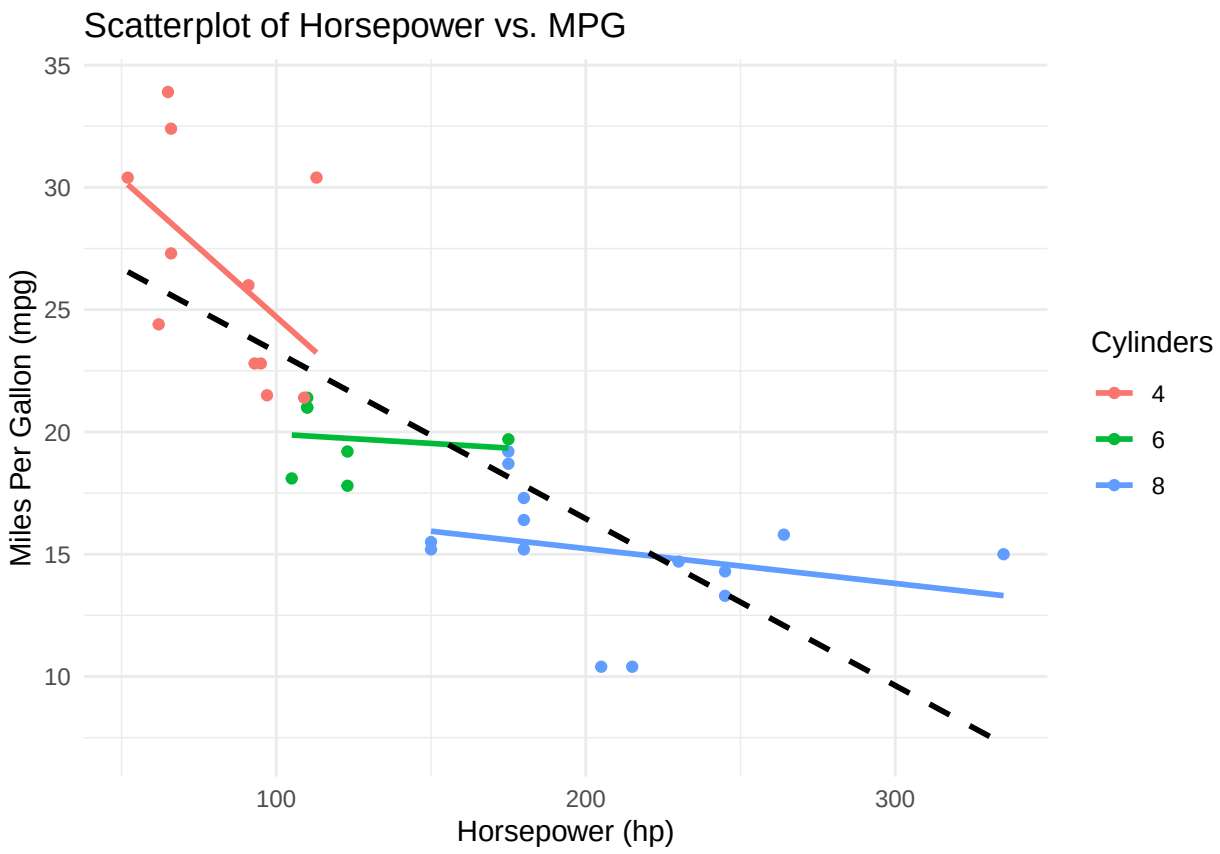
ii) the scatter plot is well labeled (include a legend for the cylinder number);

iii) one line of best fit is drawn for each cylinder number;

iv) add an overall line of best fit for all cylinder types (you can do this in a separate graph or on the same axes).

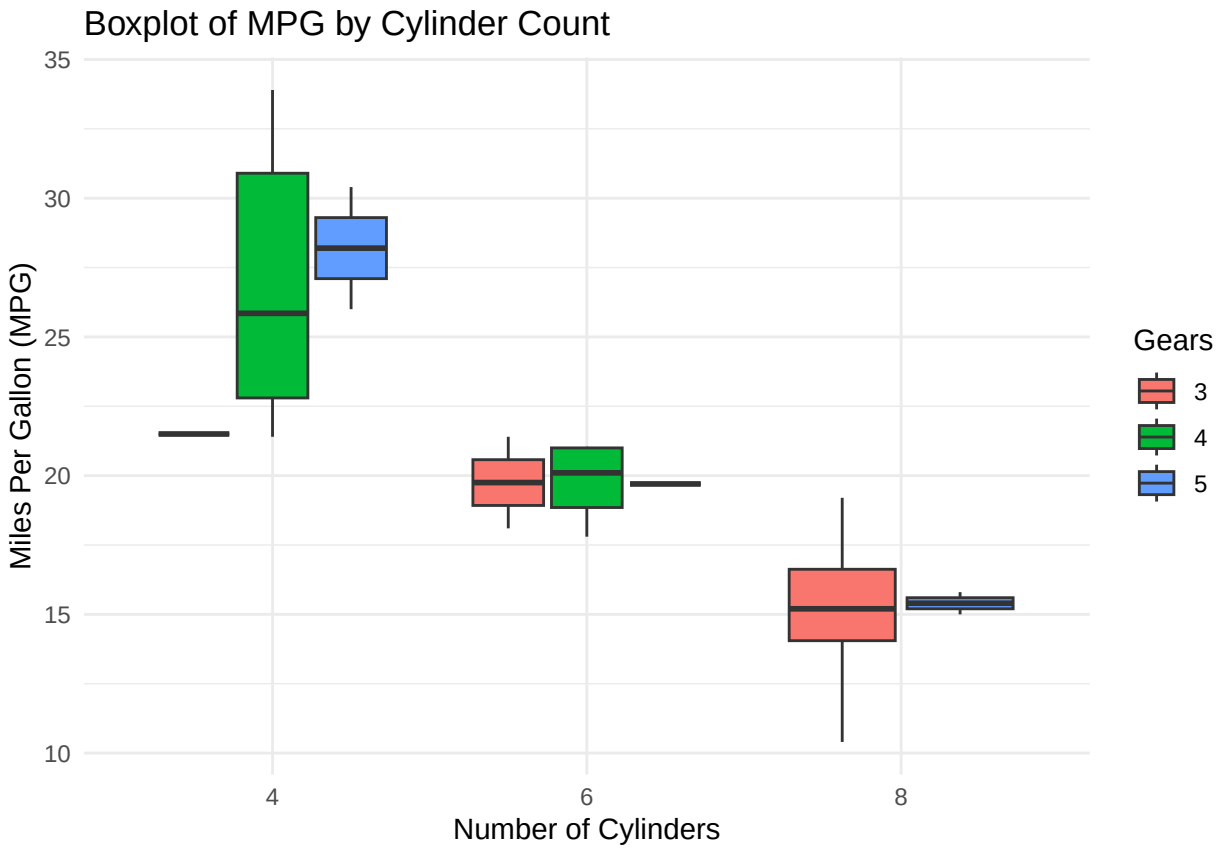
```
# Load necessary libraries
library(ggplot2)

# Load the built-in mtcars dataset
data(mtcars)

ggplot(mtcars, aes(x = hp, y = mpg, color = as.factor(cyl))) +
  # scatterplot points
  geom_point() +
  # best line of fit by cylinder group
  geom_smooth(method = "lm", aes(group = cyl), se = FALSE) +
  # overall best line of fit
  geom_smooth(method = "lm", color = "black", se = FALSE, linetype = "dashed") +
  labs(
    title = "Scatterplot of Horsepower vs. MPG",
    x = "Horsepower (hp)",
    y = "Miles Per Gallon (mpg)",
    color = "Cylinders"
  ) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



b) (2 points) Create a boxplot with `ggplot2`. The number of cylinders should be on the x-axis and `mpg` on the y-axis. Color by the number of gears.

```
ggplot(mtcars, aes(x = as.factor(cyl), y = mpg, fill = as.factor(gear))) +
  geom_boxplot() +
  labs(
    title = "Boxplot of MPG by Cylinder Count",
    x = "Number of Cylinders",
    y = "Miles Per Gallon (MPG)",
    fill = "Gears"
  ) +
  theme_minimal()
```



c) (2 points) Comment on the overall relationships you observed. How do the analyzed variables seem to affect `mpg`?

On average, cars with less cylinders had better mpg than cars with more cylinders. Gears did not seem to change the mpg much. The only exception is that in the four cylinder category, 3 gear cars had lower mpg than the 4 gear or 5 gear cars.

3. Expected value and variance (1 Bonus point in total)

(Submit the solution of this task either by submitting an additional hand-written page attached at the end of your knitted .rmd-file, or by incorporating `LaTeX` code in this .rmd-file).

Let X be a random variable with pdf $f_X(x) = \frac{32}{15}x^{-3}, x \in (1, 4)$.

a) (0.5 Bonus points) Calculate the expected value $E(X)$ by hand.

$$E(X) = \int_1^4 x f_X(x)\, dx$$

$$E(X) = \int_1^4 x * \frac{32}{15}x^{-3}\, dx$$

$$E(X) = \frac{32}{15}\int_1^4 x^{-2}\, dx$$

$$E(X) = \frac{32}{15}\left[-x^{-1}\right]_1^4$$

$$E(X) = \frac{32}{15}\left(-\frac{1}{4} + 1\right)$$

$$E(X) = \frac{32}{15} * \frac{3}{4}$$

$$E(X) = \frac{8}{5} = 1.6$$

b) (0.5 Bonus points) Calculate the variance $Var(X)$ by hand.

$$\sigma(X^2) = E(X^2) - [E(X)]^2$$

Lets solve $E(X^2)$ first:

$$E(X^2) = \int_1^4 x^2 f_X(x)\, dx$$

$$E(X^2) = \int_1^4 x^2 * \frac{32}{15} x^{-3}\, dx$$

$$E(X^2) = \frac{32}{15} \int_1^4 x^{-1}\, dx$$

$$E(X^2) = \frac{32}{15} \left[ln(x)\right]_1^4\, dx$$

$$E(X^2) = \frac{32}{15} ln(4)\, dx$$

Now plug it in:

$$\sigma(X^2) = \frac{32}{15} ln(4) - 1.6^2$$

$$\sigma(X^2) = 0.397428$$