Midterm Exam - Statistical Programming in R

Andrew Shao (as13381)

2024-10-26

Question 1 (4 points):

You are analyzing air quality data across cities for public health research. The dataset `city_air_quality` consists of two vectors: city names and their corresponding Air Quality Index (AQI) values recorded over a week for five cities. The AQI for `CityD` is missing.

```
city_names <- c("CityA", "CityB", "CityC", "CityD", "CityE")
aqi_values <- c(70, 120, 95, NA, 45)
```

a) **Object Assignment and NA Handling:**
Assign `aqi_values` to a new object `clean_aqi`, replacing the missing value (`NA`) with the median of the remaining values. Then print the updated `clean_aqi`.

b) **Logical Comparisons and Subsetting:**
Using logical operators, return the names of the cities where the AQI is either above 100 or below 50, in alphabetical order. Do this in a single line of code.

c) **Vector Set Operations:**
Create a new vector `high_pollution_cities` for cities with AQI greater than 60, and `low_pollution_cities` for cities with AQI less than 80. Use a set operation to find which cities fall in both categories.

d) **Statistical Summary:**
Calculate the following statistics for the `clean_aqi` vector:

i) The standard deviation.
ii) The inter quartile range (IQR).

Answer:

a)

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
## Warning: package 'tidyr' was built under R version 4.3.2
## Warning: package 'readr' was built under R version 4.3.2
## Warning: package 'dplyr' was built under R version 4.3.2
## Warning: package 'stringr' was built under R version 4.3.2
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
```

```
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
clean_aqi <- ifelse(is.na(aqi_values), median(aqi_values, na.rm = T), aqi_values)
clean_aqi
```

```
## [1]  70.0 120.0  95.0  82.5  45.0
```

b)

```
sort(city_names[clean_aqi > 100 | clean_aqi < 50])
```

```
## [1] "CityB" "CityE"
```

c)

```
high_pollution_cities <- city_names[clean_aqi > 60]
low_pollution_cities <- city_names[clean_aqi < 80]
intersect(high_pollution_cities, low_pollution_cities)
```

```
## [1] "CityA"
```

d)

```
sd(clean_aqi)
```

```
## [1] 27.95085
```

```
IQR(clean_aqi)
```

```
## [1] 25
```

Question 2 (4 points):

You are working on patient health records to understand the relationship between age, gender, smoking habits, and cholesterol levels. The dataset `patients` contains information about patient demographics and health metrics.

```
library(tidyverse)
patients <- tibble(
  ID = 1:5,
  Age = c(34, 29, 45, 53, 41),
  Gender = factor(c("Male", "Female", "Male", "Female", "Male")),
  Smokes = factor(c("Yes", "No", "No", "Yes", "Yes")),
  Cholesterol = c(190, 230, 180, NA, 220)
)
```

a) **Handling Missing Values:**
Replace the missing `Cholesterol` value with the mean cholesterol of all smokers, ignoring the missing value. Assign this modified dataset to a new object called `updated_patients`.

b) **Data Manipulation and Factor Levels:**
Change the factor levels of the `Smokes` column to "Non-smoker" and "Smoker". Recode the factor so that "Smoker" comes before "Non-smoker".

c) **Advanced List Manipulation:**
Create a list `patient_summary` that contains the following elements, and print its contents.

i) The `updated_patients` data frame.

ii) A summary vector with the mean age, median cholesterol, and number of smokers.

iii) A character vector that lists the patient IDs for males older than 40.

Answer:

a)

```
updated_patients <- patients %>% mutate(Cholesterol = replace_na(Cholesterol,
↪  mean(Cholesterol, na.rm = T)))
updated_patients
```

```
## # A tibble: 5 x 5
##      ID   Age Gender Smokes Cholesterol
##   <int> <dbl> <fct>  <fct>        <dbl>
## 1     1    34 Male   Yes            190
## 2     2    29 Female No             230
## 3     3    45 Male   No             180
## 4     4    53 Female Yes            205
## 5     5    41 Male   Yes            220
```

b)

```
updated_patients <- updated_patients %>% mutate(Smokes = factor(Smokes, ordered = T,
↪  levels = c('Yes', 'No'), labels = c('Smoker', 'Non-smoker')))
updated_patients$Smokes
```

```
## [1] Smoker     Non-smoker Non-smoker Smoker     Smoker
## Levels: Smoker < Non-smoker
```

c)

```
patient_summary <- list(updated_patients,
                        c(mean_age = mean(updated_patients$Age, na.rm = T),
                          median_cholesterol = median(updated_patients$Cholesterol, na.rm
                          ↪  = T),
                          number_smokers = sum(updated_patients$Smokes == 'Smoker')),
                        as.character(updated_patients$ID[updated_patients$Age > 40]))
patient_summary
```

```
## [[1]]
## # A tibble: 5 x 5
##      ID   Age Gender Smokes     Cholesterol
##   <int> <dbl> <fct>  <ord>            <dbl>
## 1     1    34 Male   Smoker             190
## 2     2    29 Female Non-smoker         230
## 3     3    45 Male   Non-smoker         180
## 4     4    53 Female Smoker             205
## 5     5    41 Male   Smoker             220
##
## [[2]]
##           mean_age median_cholesterol     number_smokers
```

```
##              40.4             205.0               3.0
##
## [[3]]
## [1] "3" "4" "5"
```

Question 3 (4 points):

You are tasked with visualizing exercise data to examine its relationship with body mass index (BMI) and cholesterol levels. The dataset `exercise_data` contains weekly exercise hours, BMI, and cholesterol values for a group of individuals.

```
exercise_data <- data.frame(
  Hours_Exercise = c(5, 7, 3, 2, 8, 4, 6),
  BMI = c(23.5, 25.2, 27.8, 31.1, 22.1, 29.6, 24.3),
  Cholesterol = c(190, 220, 230, 240, 200, 250, 210)
)
```

a) **Multiple Geometries and Custom Aesthetics:**
Create a scatterplot of `Hours_Exercise` vs `BMI` with `ggplot2`, save it as `my_plot` and display the plot. Add the following elements:

i) Color points based on `Cholesterol` values, using a continuous color spectrum with green color for the low values and red color for the high values.

ii) Distinguish between `BMI` values above 25 (overweight) and those not above 25 by altering the shape of the points.

b) **Smoothline Fit and Local Mappings:**
Add a smooth line (`geom_smooth()`) to `my_plot` that fits a linear model, but apply this fit only to the individuals with a `BMI` greater than 25.

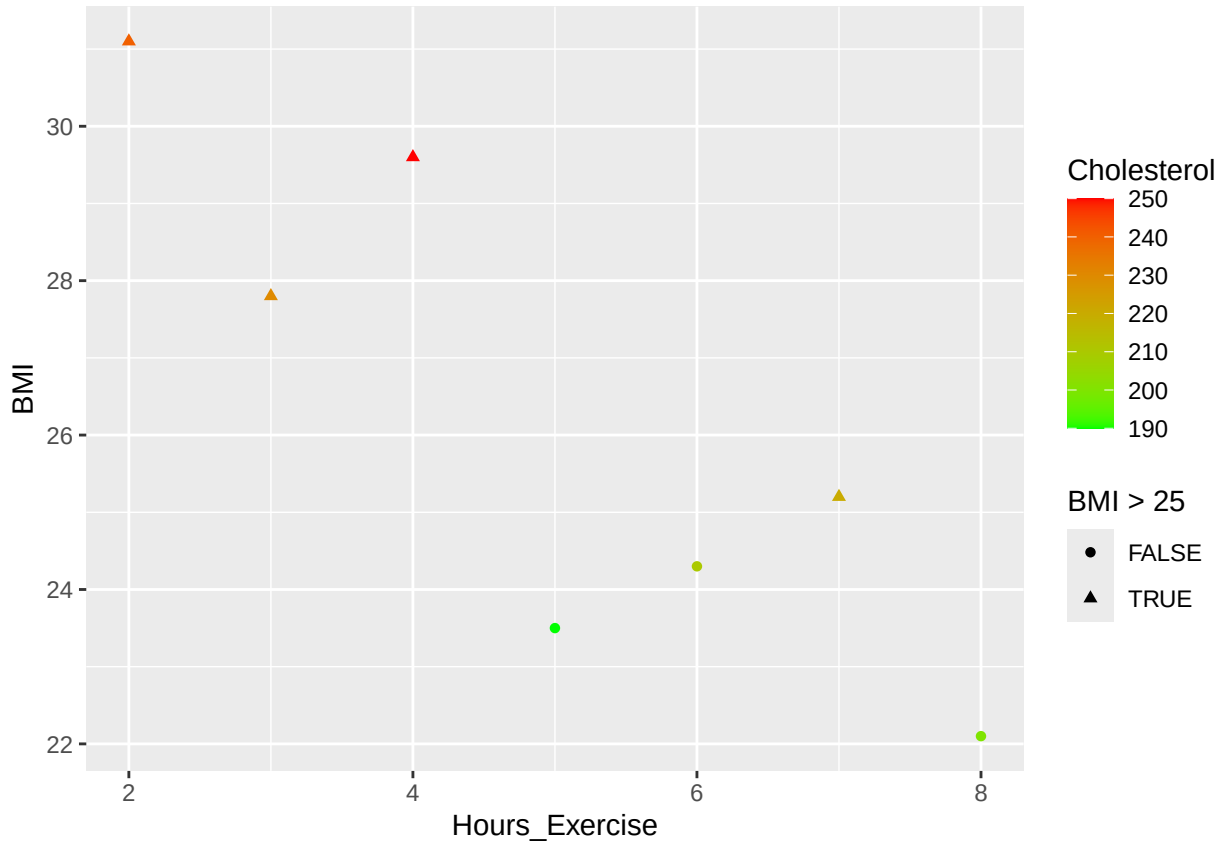
c) **Customizing Axes and Titles:**
On top of the plot in b), customize the axes by adding meaningful labels, and add a title `"Exercise, BMI, and Cholesterol Relationship"`.

d) **Facet and Complex Customization:**
Create a faceted plot where each facet represents the scatterplot of `Cholesterol` vs. `Hours_Exercise` for a BMI category ("Above 25" or "Not above 25").

Answer:

a)

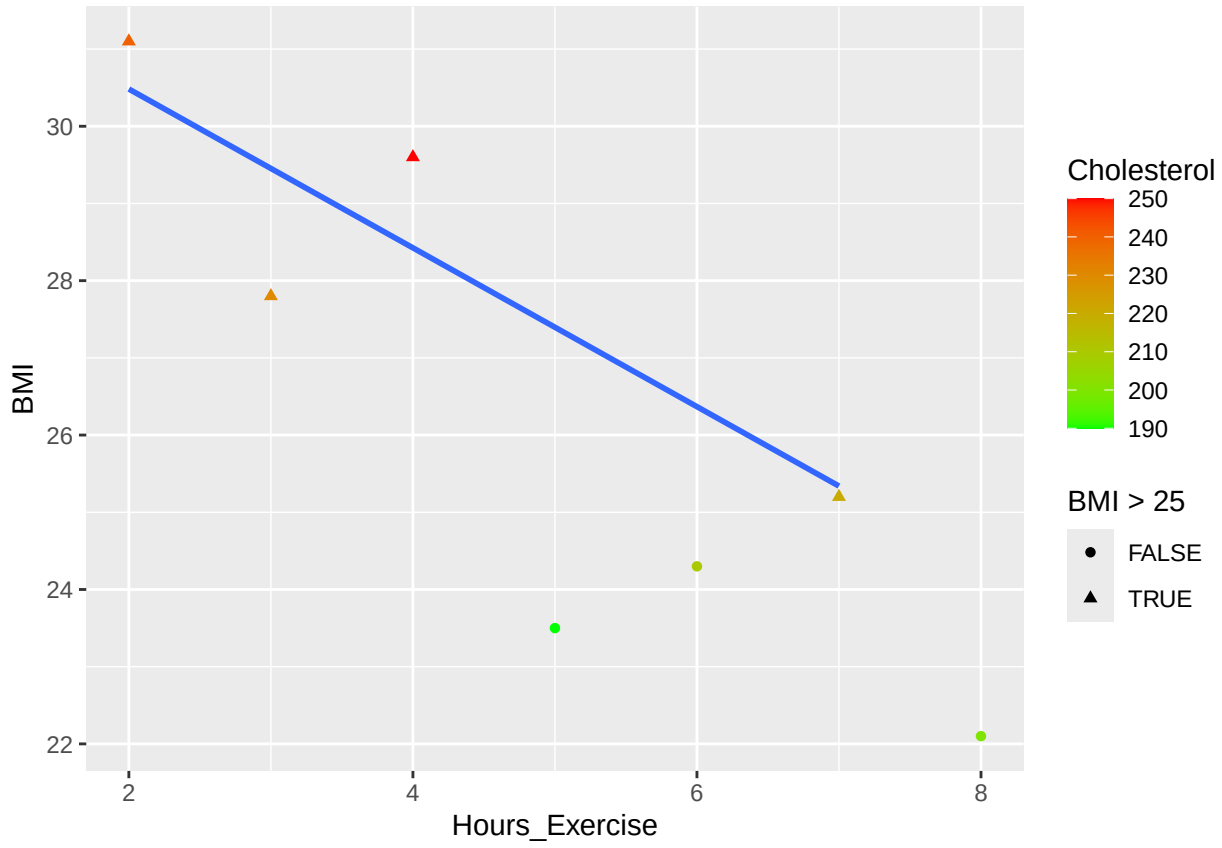
```
my_plot <- ggplot() +
  geom_point(data = exercise_data,
             aes(x = Hours_Exercise,
                 y = BMI,
                 color = Cholesterol,
                 shape = BMI > 25)) +
  scale_color_gradient(low = 'green', high = 'red')
my_plot
```



b)

```
my_plot_b <- my_plot +
  geom_smooth(data = subset(exercise_data, BMI > 25),
              aes(x = Hours_Exercise,
                  y = BMI),
              method = 'lm',
              se = F)
my_plot_b
```

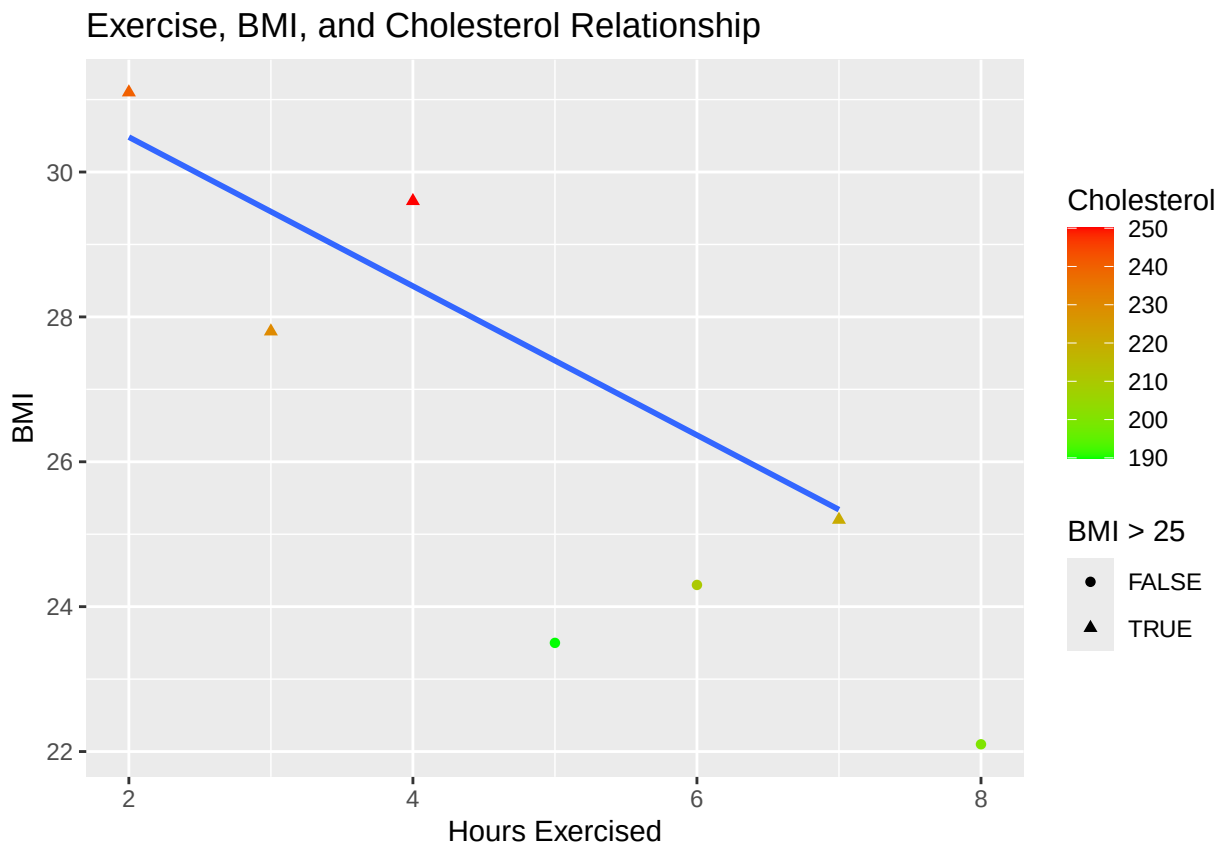
```
## `geom_smooth()` using formula = 'y ~ x'
```



c)

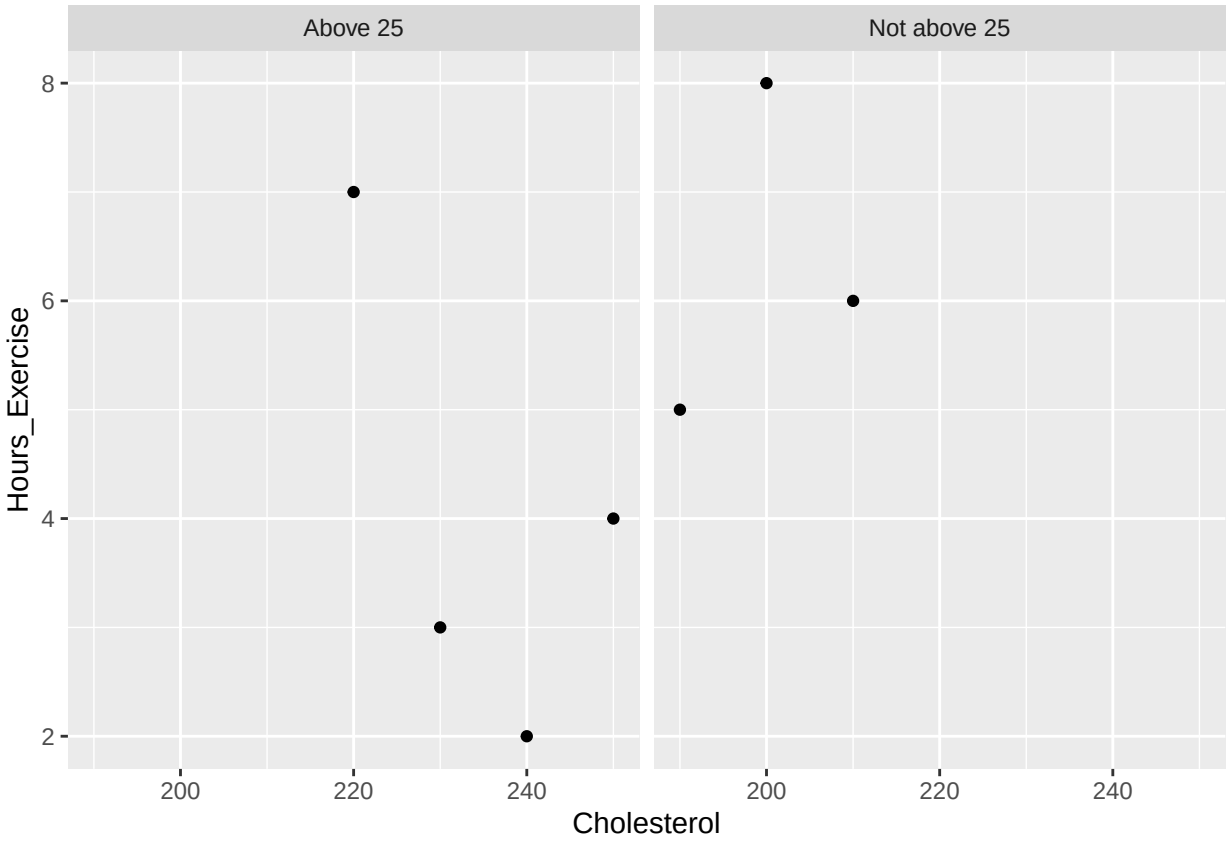
```
my_plot_b +
  ggtitle("Exercise, BMI, and Cholesterol Relationship") +
  xlab('Hours Exercised') +
  ylab('BMI')
```

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



d)

```
ggplot(data = exercise_data %>%
         mutate(BMI_category = ifelse(BMI > 25, 'Above 25', 'Not above 25'))) +
  geom_point(aes(x = Cholesterol,
                 y = Hours_Exercise)) +
  facet_wrap('BMI_category')
```



Question 4 (4 points):

You are working with the `mtcars` dataset, and you are required to perform data manipulations and visualizations to gain insights. Note: load the data by the code `data(mtcars)`.

a) **Data Manipulation:**

- Filter the `mtcars` dataset to include only cars with `mpg` greater than the median `mpg` of the entire dataset.
- Create a new column called `performance_score` which is calculated as the ratio of horsepower (`hp`) to weight (`wt`).
- Create a new column called `performance_category` which categorizes cars based on their `performance_score` into three categories:
  - `"High Performance"` for cars with a score greater than 60.
  - `"Moderate Performance"` for cars with a score between 40 and 60.
  - `"Low Performance"` for cars with a score less than 40.

b) **Data Visualization:**

- Create a **density plot** to show the distribution of `performance_score` across different `cyl` (cylinders) categories. Use different colors to represent the density curves for each cylinder category.
- Generate a **violin plot (with boxplot inside)** to compare the distribution of `mpg` across the `performance_category` categories. Display `performance_category` on the x-axis, arranging the categories from left to right in the order of `Low Performance`, `Moderate Performance`, and `High Performance`. Fill the the violin plots with different colors for each `performance_category` category.

c) **Data Aggregation:**

- Group the filtered data by the `cyl` and `performance_category` categories and calculate the mean `mpg` and median `hp` for each group. Display the result as a dataframe or tibble.

Answer:

a)

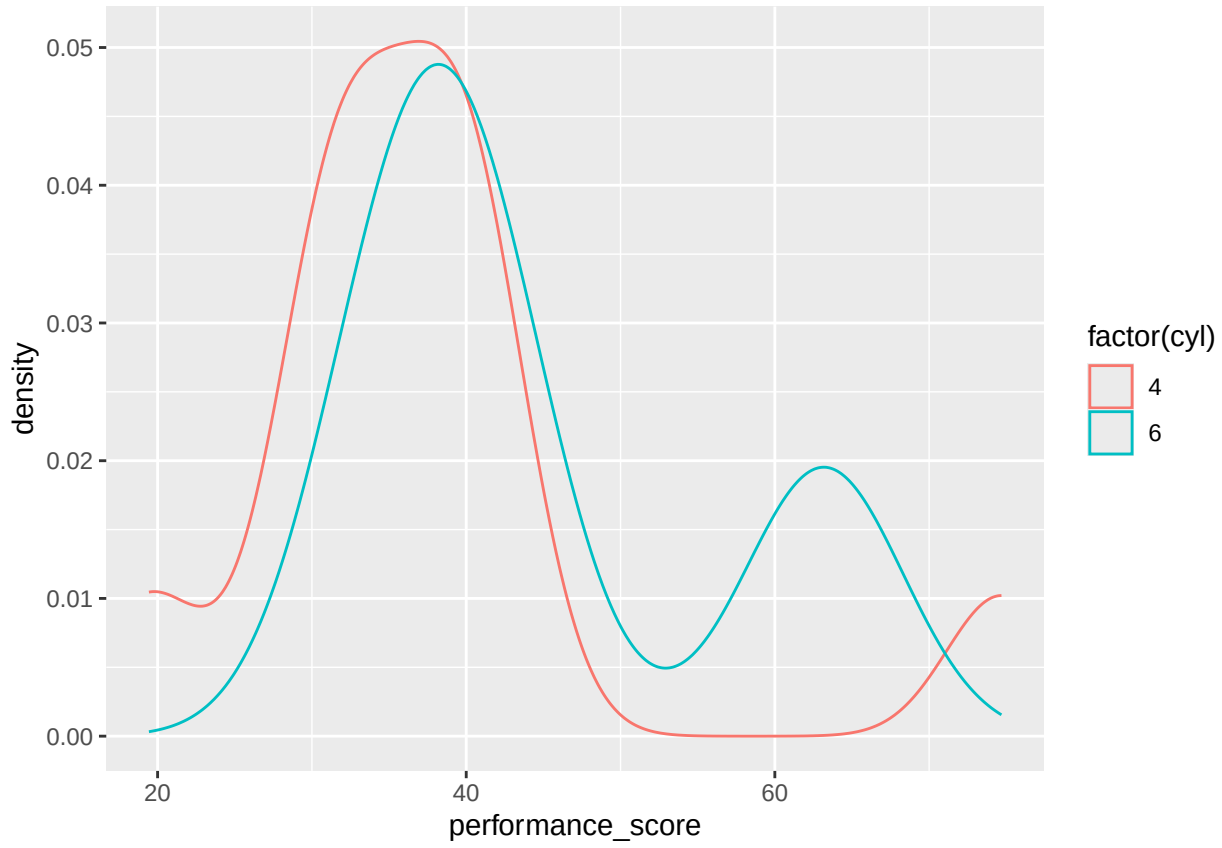
```
data("mtcars")

mtcars <- mtcars %>%
  subset(mpg > median(mpg, na.rm = T)) %>%
  mutate(performance_score = hp / wt,
         performance_category = case_when(performance_score > 60 ~ 'High Performance',
                                          performance_score < 40 ~ 'Low Performance',
                                          .default = 'Moderate Performance'))
head(mtcars)
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
##                 performance_score performance_category
## Mazda RX4                41.98473 Moderate Performance
## Mazda RX4 Wag            38.26087      Low Performance
## Datsun 710               40.08621 Moderate Performance
## Hornet 4 Drive           34.21462      Low Performance
## Merc 240D                19.43574      Low Performance
## Merc 230                 30.15873      Low Performance
```

b)

```
ggplot(mtcars) +
  geom_density(aes(x = performance_score,
                   color = factor(cyl)))
```



c)

```
mtcars %>%
  group_by(cyl, performance_category) %>%
  summarise(mean_mpg = mean(mpg, na.rm = T),
            median_hp = median(hp, na.rm = T))
```

```
## `summarise()` has grouped output by 'cyl'. You can override using the `.groups`
## argument.
```

```
## # A tibble: 6 x 4
## # Groups:   cyl [2]
##     cyl performance_category mean_mpg median_hp
##   <dbl> <chr>                   <dbl>     <dbl>
## 1     4 High Performance         30.4       113
## 2     4 Low Performance          26.8        66
## 3     4 Moderate Performance     24.4        92
## 4     6 High Performance         19.7       175
## 5     6 Low Performance          21.2       110
## 6     6 Moderate Performance     21         110
```

Question 5 (4 points):

You are provided with two datasets:

- `large_sales_data.txt`: A tab-delimited text file containing sales data with the columns `TransactionID`, `CustomerID`, `ProductID`, `SalesAmount`, `TransactionDate`.

- `customer_info.json`: A JSON file containing customer information with fields: `CustomerID`, `Name`, `Age`, `Region`, and `LoyaltyScore`.

a) **Data Import and Initial Transformation:**

- Import the `large_sales_data.txt` file as a dataframe or tibble.
- Import the `customer_info.json` file as a dataframe or tibble. Hint: google how to import a JSON file.
- For the sales data:
  - Convert the `TransactionDate` column into an appropriate date format. Hint: use the `as.Date()` function.
  - Create a new column `TransactionMonth` which extracts the month from `TransactionDate`. Hint: use the `month()` function from the package `lubridate`.

b) **Data Filtering:**

- Filter the sales data to include only transactions where the `SalesAmount` is greater than 5000, and exclude transactions that occurred in the first quarter (January, February, March). Display the first 10 observations of the result with columns `TransactionID`, `SalesAmount`, `TransactionDate` and `TransactionMonth`.
- Filter the customer data to include only customers who are at least 30 years old and have a `LoyaltyScore` of 8 or more. Display the first 7 observations of the result.

c) **Data Analysis:**

- In the filtered sales data, calculate the total number of transactions and the total `SalesAmount` for each month. Display the result as a dataframe or tibble.
- In the filtered customer data, count how many customers belong to each region, and calculate the average `LoyaltyScore` for each region. Display the result as a dataframe or tibble.

d) **Data Export:**

- Export the filtered sales data and the filtered customer data (from Sub-question (b)) as sheet `sales` and sheet `customers` into a single `.xlsx` file named `filtered_sales_customer_data.xlsx`.

Answer:

a)

```
# install.packages('jsonlite')
library(jsonlite)
```

```
## Warning: package 'jsonlite' was built under R version 4.3.3
```

```
##
## Attaching package: 'jsonlite'
```

```
## The following object is masked from 'package:purrr':
##
##     flatten
```

```
large_sales_data <- read.table('large_sales_data.txt', header = T)
customer_info <- jsonlite::fromJSON('customer_info.json')


large_sales_data <- large_sales_data %>%
  mutate(TransactionDate = as.Date(TransactionDate),
         TransactionMonth = month(TransactionDate))

head(large_sales_data)
```

```
##   TransactionID CustomerID ProductID SalesAmount TransactionDate
## 1             1         92        12     4973.08      2023-03-27
## 2             2         43        25     1491.20      2023-09-30
## 3             3         56        20     7736.19      2023-04-24
## 4             4         59        39     2198.68      2023-08-22
## 5             5         35        49     9027.76      2023-09-20
## 6             6          2         6     8978.02      2023-12-12
##   TransactionMonth
## 1                3
## 2                9
## 3                4
## 4                8
## 5                9
## 6               12
```

```
head(customer_info)
```

```
##   CustomerID       Name Age Region LoyaltyScore
## 1          1 Customer 1  45   West         1.15
## 2          2 Customer 2  28  North         1.40
## 3          3 Customer 3  37   West         7.05
## 4          4 Customer 4  68  North         5.57
## 5          5 Customer 5  57   West         2.66
## 6          6 Customer 6  24  South         4.35
```

b)

```
large_sales_data_filtered <- large_sales_data %>%
  subset(SalesAmount > 5000 & TransactionMonth > 3)
head(select(large_sales_data_filtered, `TransactionID`, `SalesAmount`, `TransactionDate`,
↪  `TransactionMonth`), 10)
```

```
##    TransactionID SalesAmount TransactionDate TransactionMonth
## 3              3     7736.19      2023-04-24                4
## 5              5     9027.76      2023-09-20                9
## 6              6     8978.02      2023-12-12               12
## 7              7     8579.19      2023-09-29                9
## 11            11     5173.79      2023-06-30                6
## 20            20     8176.81      2023-09-14                9
## 25            25     8940.59      2023-12-12               12
## 27            27     8380.22      2023-10-31               10
## 32            32     9571.07      2023-11-22               11
## 33            33     7948.00      2023-06-02                6
```

```
customer_info_filtered <- customer_info %>%
  subset(Age >= 30 & LoyaltyScore >= 8)
head(customer_info_filtered, 7)
```

```
##    CustomerID        Name Age Region LoyaltyScore
## 10         10 Customer 10  67   West         9.58
## 12         12 Customer 12  63  North         9.45
## 15         15 Customer 15  47   West         8.74
## 25         25 Customer 25  46  South         8.41
## 27         27 Customer 27  54  North         9.57
## 32         32 Customer 32  44   West         9.17
## 48         48 Customer 48  55   East         8.47
```

c)

```
large_sales_data_filtered %>%
  group_by(TransactionMonth) %>%
  summarize(transaction_total = n(),
            total_sales = sum(SalesAmount))
```

```
## # A tibble: 9 x 3
##   TransactionMonth transaction_total total_sales
##              <dbl>             <int>       <dbl>
## 1                4                 9      67852.
## 2                5                 5      36609.
## 3                6                10      71794.
## 4                7                 9      68718.
## 5                8                 7      50031.
## 6                9                11      87487.
## 7               10                10      66402.
## 8               11                 5      36402.
## 9               12                 6      47421.
```

```
customer_info_filtered %>%
  group_by(Region) %>%
  summarize(customer_total = n(),
            average_loyalty = mean(LoyaltyScore, na.rm = T))
```

```
## # A tibble: 4 x 3
##   Region customer_total average_loyalty
##   <chr>           <int>           <dbl>
## 1 East                1            8.47
## 2 North               2            9.51
## 3 South               1            8.41
## 4 West                3            9.16
```

d)

```
# install.packages('writexl')
library(writexl)
```

```
## Warning: package 'writexl' was built under R version 4.3.3
```

```
write_xlsx(list(sales = large_sales_data_filtered, customers = customer_info_filtered),
↪  'filtered_sales_customer_data.xlsx')
```