



**University of Windsor**

Lab 1

COMP 8157 | Advanced Database Topics – Summer 2025

Names:

Aleena Ali Azeem - 110190830

Simranpreet Kaur - 110189426

Priyadharshan Reddy S - 110191285

## Part 1: Data Exploration (12 marks)

1. Import the **Heart Attack** datasets (2 marks).

```
# 1. Import the Heart Attack dataset
df <- read.csv("Heart_Attack.csv")
```

- Explanation: `read.csv("Heart_Attack.csv")`:  
This function **reads a CSV (Comma-Separated Values) file** named "Heart\_Attack.csv" into R.
- `df <-`: The `<-` operator is used to **assign** the result of the `read.csv()` function to a variable named `df`.  
`df` stands for **data frame**, which is a common data structure in R used to store datasets (like tables with rows and columns).
- So basically we're loading the "Heart\_Attack.csv" file into R and storing it in a variable called `df` so as to work with that dataset in the rest of your code.

2. Summarize that Heart Attack dataset and explain the output (4 marks).

```
# 2. Summarize the dataset
summary(df)
```

- `summary()` is a **built-in R function** that provides a quick **statistical summary** of each column in the data frame `df`.

For each column in `df`, it gives:

- **Numerical columns:**
  - Minimum (Min.)
  - 1st Quartile (1st Qu.)
  - Median
  - Mean
  - 3rd Quartile (3rd Qu.)
  - Maximum (Max.)
- **Categorical columns (factors or characters):**
  - A count of the number of observations in each category/level.
- This tells R to display a **summary report** of the dataset so you can quickly understand the distribution and type of data in each column.

```
> summary(df)
```

age	sex	cp	trtbps	chol
Min. :29.00	Min. :0.0000	Min. :0.000	Min. : 94.0	Min. :126.0
1st Qu.:47.50	1st Qu.:0.0000	1st Qu.:0.000	1st Qu.:120.0	1st Qu.:211.0
Median :55.00	Median :1.0000	Median :1.000	Median :130.0	Median :240.0
Mean :54.37	Mean :0.6854	Mean :0.967	Mean :131.6	Mean :246.3
3rd Qu.:61.00	3rd Qu.:1.0000	3rd Qu.:2.000	3rd Qu.:140.0	3rd Qu.:274.5
Max. :77.00	Max. :1.0000	Max. :3.000	Max. :200.0	Max. :564.0
	NA's :1			

fbs	restecg	thalachh	exng	oldpeak
Min. :0.0000	Min. :0.0000	Min. : 71.0	Min. :0.0000	Min. :0.000
1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:133.5	1st Qu.:0.0000	1st Qu.:0.000
Median :0.0000	Median :1.0000	Median :153.0	Median :0.0000	Median :0.800
Mean :0.1536	Mean :0.5333	Mean :149.6	Mean :0.3523	Mean :1.043
3rd Qu.:0.0000	3rd Qu.:1.0000	3rd Qu.:166.0	3rd Qu.:1.0000	3rd Qu.:1.600
Max. :1.0000	Max. :2.0000	Max. :202.0	Max. :1.0000	Max. :6.200
NA's :10	NA's :3		NA's :22	NA's :1

slp	caa	thall	output
Min. :0.000	Min. :0.0000	Min. :0.000	Min. :0.0000
1st Qu.:1.000	1st Qu.:0.0000	1st Qu.:2.000	1st Qu.:0.0000
Median :1.000	Median :0.0000	Median :2.000	Median :1.0000
Mean :1.399	Mean :0.7621	Mean :2.314	Mean :0.5446
3rd Qu.:2.000	3rd Qu.:1.0000	3rd Qu.:3.000	3rd Qu.:1.0000
Max. :2.000	Max. :4.0000	Max. :3.000	Max. :1.0000
	NA's :13		

3. Show the structure and dimension of the dataset and explain it (2 marks).

```
> str(df)
```

```
'data.frame': 303 obs. of 14 variables:
```

```
$ age      : int  63 37 41 56 57 57 56 44 52 57 ...
```

```
$ sex      : int  1 1 0 1 0 1 0 1 1 1 ...
```

```
$ cp       : int  3 2 1 1 0 0 1 1 2 2 ...
```

```
$ trtbps   : int  145 130 130 120 120 140 140 120 172 150 ...
```

```
$ chol     : int  233 250 204 236 354 192 294 263 199 168 ...
```

```
$ fbs      : int  1 0 0 0 0 0 0 0 1 0 ...
```

```
$ restecg  : int  0 1 0 1 1 1 0 1 1 1 ...
```

```
$ thalachh : int  150 187 172 178 163 148 153 173 162 174 ...
```

```
$ exng     : int  0 0 0 0 1 0 0 0 0 0 ...
```

```
$ oldpeak  : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
```

```
$ slp      : int  0 0 2 2 2 1 1 2 2 2 ...
```

```
$ caa      : int  0 0 0 0 0 NA 0 0 NA 0 ...
```

```
$ thall    : int  1 2 2 2 2 1 2 3 3 2 ...
```

```
$ output   : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
> dim(df)
```

```
[1] 303 14
```

## str(df)

- `str()` stands for **structure**. This function shows the **internal structure** of the df data frame. This shows us the following:
- The **type** of object (data.frame)
- The **number of observations (rows)** and **variables (columns)**
- The **name** of each column
- The **data type** of each column (e.g., int, num, Factor, chr)
- A **preview of the first few values** in each column

## dim(df)

- `dim()` stands for **dimensions**.
- It returns a **numeric vector** with two numbers:
  - **Number of rows (observations)**
  - **Number of columns (variables)**

4. Show the first 8 rows and the last 5 rows of the dataset (2 marks).

```
> colnames(df)
[1] "age"      "sex"      "cp"      "trtbps"   "chol"     "fbs"      "restecg"
[8] "thalachh" "exng"     "oldpeak" "slp"      "caa"      "thall"    "output"
> head(df, 8)
  age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
1  63  1  3   145  233  1      0      150    0    2.3    0  0    1      1
2  37  1  2   130  250  0      1      187    0    3.5    0  0    2      1
3  41  0  1   130  204  0      0      172    0    1.4    2  0    2      1
4  56  1  1   120  236  0      1      178    0    0.8    2  0    2      1
5  57  0  0   120  354  0      1      163    1    0.6    2  0    2      1
6  57  1  0   140  192  0      1      148    0    0.4    1 NA    1      1
7  56  0  1   140  294  0      0      153    0    1.3    1  0    2      1
8  44  1  1   120  263  0      1      173    0    0.0    2  0    3      1
> tail(df, 5)
  age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
299  57  0  0   140  241  0      1      123    1    0.2    1  0    3      0
300  45  1  3   110  264  0      1      132    0    1.2    1  0    3      0
301  68  1  0   144  193  1      1      141    0    3.4    1  2    3      0
302  57  1  0   130  131  0      1      115    1    1.2    1  1    3      0
303  57  0  1   130  236  0      0      174    0    0.0    1  1    2      0
```

## 1. head(df, 8)

- `head()` displays the **first few rows** of the dataset.
- By default, it shows the first **6** rows, but here we've specified 8, so it shows the **first 8 rows** of df.

## 2. tail(df, 5)

- tail() shows the **last few rows** of the dataset.
- By default, it also shows 6 rows, but here we're asking for **5 rows** — so it displays the **last 5 rows** of df.

5. Show the column names of the Heart Attack dataset (2 marks).

```
> colnames(df)
[1] "age"      "sex"      "cp"      "trtbps"   "chol"     "fbs"      "restecg"
[8] "thalachh" "exng"     "oldpeak" "slp"      "caa"      "thal"     "output"
```

### Explanation:

- colnames() stands for **column names**.
- It returns a **character vector** containing the list of **names of all the columns** in the data frame df.

## Part 2: Data Pre-Processing (28 marks)

6. What is the class variable in the Heart Attack dataset? What does it indicate (4 marks)?

```
# Checking unique values of the class variable 'output'
unique(heart_data$output)
```

It tells the output based on the variable such as cholesterol etc.

### Explanation:

The output column is used as the class variable. It contains values like 0 and 1, where:

- 0 means the person did not have a heart attack
- 1 means the person had a heart attack

```
> unique(heart_data$output)
[1] 1 0
```

7. What is the datatype of the class variable (4 marks)?

```
# Checking the data type of the 'output' column
class(heart_data$output)
```

This code checks if the class variable is stored as a number or as a category. If it is numeric (like integer or double).

```
> class(heart_data$output)
[1] "integer"
```

8. Change the class type of the class variable of Heart Attack dataset to factor. Show the output after the conversion **(4 marks)**.

So the factor is applied to classify the levels of the column and how many each level is repeated.

```
# Converting 'output' column to a factor (categorical)
heart_data$output <- as.factor(heart_data$output)
# Verifying the conversion
str(heart_data$output)
```

**Explanation:**

The output column is changed to a factor. This means it will be treated as a category, not a number. This is important when doing classification tasks, as it clearly separates the two groups: heart attack and no heart attack.

```
> heart_data$output <- as.factor(heart_data$output)
> str(heart_data$output)
Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

9. Find the sum of the missing values in Heart Attack dataset **(4 marks)**.

```
# Finding the sum of missing values in the dataset
sum(is.na(heart_data))
```

**Explanation:**

This line checks how many cells in the dataset are empty or have missing values. It's important to find missing data before starting any analysis or building models.

```
> sum(is.na(heart_data))
[1] 50
```

10. Find which columns contain missing values in the dataset. What is the total missing values for each column **(4 marks)**?

```
# Checking how many missing values exist in each column
colSums(is.na(heart_data))
```

**Explanation:**

This code shows how many missing values are in each column. It helps to know which specific variables need fixing or attention before moving forward with data analysis.

```
> # Checking how many missing values exist in each column
> colSums(is.na(heart_data))
```

age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng
0	1	0	0	0	10	3	0	22
oldpeak	slp	caa	thall	output				
1	0	13	0	0				

11. Replace the missing values in the Heart Attack by 0. Check what if the missing values was replaced successfully **(4 marks)**.

```
# Replacing all missing values in the dataset with 0
heart_data[is.na(heart_data)] <- 0
# Verifying that there are no missing values now
sum(is.na(heart_data))
```

All missing values in the dataset are replaced with 0. After this, the code checks again to make sure there are no more missing values. This step helps clean the data so that it won't cause errors during analysis.

```
> # Verifying that there are no missing values now
> sum(is.na(heart_data))
[1] 0
```

12. Rename the sex attribute from (0 and 1) to (Male and Female). Show the conversion output of the specific attribute **(4 marks)**.

```
# Converting the 'sex' column to factors with labels
heart_data$sex <- factor(heart_data$sex, levels = c(0, 1), labels = c("Female", "Male"))
# Displaying the updated values in the 'sex' column
head(heart_data$sex)
```

In the original dataset, the sex column used numbers (1 for Male, 0 for Female). These numbers are changed to text labels ("Male" and "Female") to make the data easier to read and understand when doing analysis or creating charts.

```
> # Displaying the updated values in the 'sex' column
> head(heart_data$sex)
[1] Male Male Female Male Female Male
Levels: Female Male
```

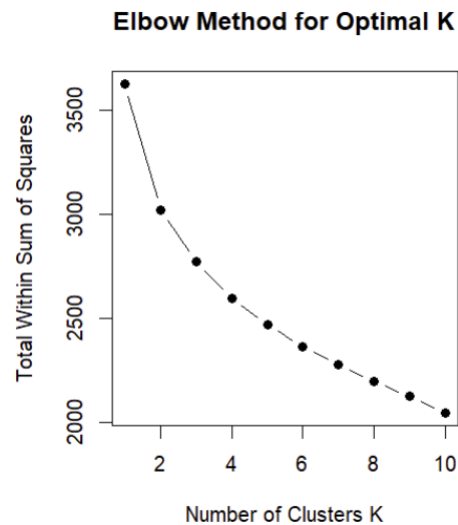
### Part 3: Clustering (14 marks)

13. Apply clustering techniques. Explain how many clusters there are and why (7 marks).

First, we use the numerical data and then we use the scale the data. Then we apply the elbow method.

```
wss <- vector()
for (k in 1:10) {
  wss[k] <- sum(kmeans(scaled_df, centers = k, nstart = 25)$within
}

plot(1:10, wss, type = "b", pch = 19,
     xlab = "Number of Clusters K",
     ylab = "Total Within Sum of Squares",
     main = "Elbow Method for Optimal K")
set.seed(123)
km_result <- kmeans(scaled_df, centers = 2, nstart = 25)
```



After which we will make clusters. Using the elbow method, we use  $K = 2$ . As the decrease becomes less after that.



```

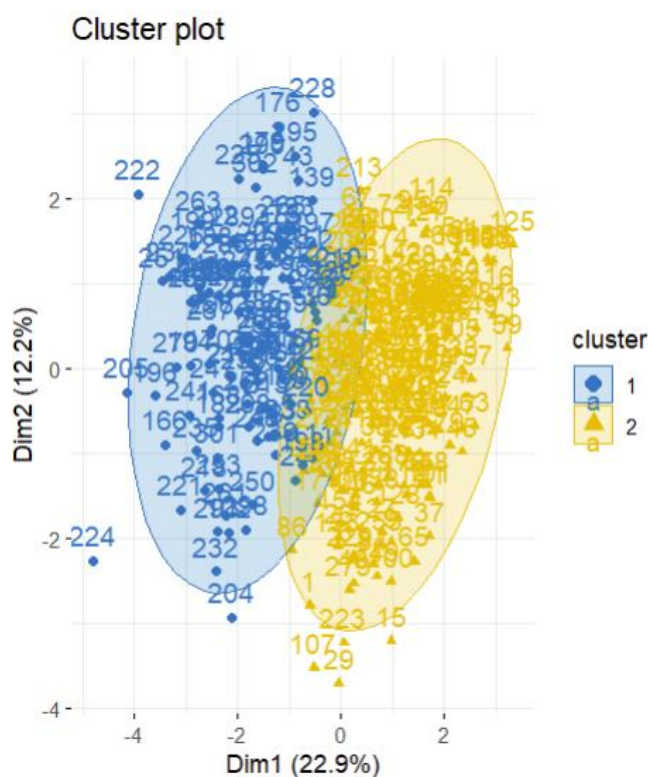
numeric_df <- df[, sapply(df, is.numeric)]
scaled_df <- scale(numeric_df)
set.seed(123)
km_result <- kmeans(scaled_df, centers = 2, nstart = 25)
fviz_cluster(km_result, data = scaled_df,
              ellipse.type = "norm",
              palette = "jco",
              ggtheme = theme_minimal())

```

```

install.packages("plotly")
library(plotly)

```



14. Which two variables have strong relationships with each other and why (7 marks)

	age	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
age	1.00	-0.07	0.28	0.21	0.12	-0.12	-0.40	0.10	0.21	-0.17	0.28	0.07
cp	-0.07	1.00	0.05	-0.08	0.09	0.04	0.30	-0.39	-0.15	0.12	-0.18	-0.16
trtbps	0.28	0.05	1.00	0.12	0.18	-0.11	-0.05	0.07	0.19	-0.12	0.10	0.06
chol	0.21	-0.08	0.12	1.00	0.01	-0.15	-0.01	0.07	0.05	0.00	0.07	0.10
fbs	0.12	0.09	0.18	0.01	1.00	-0.08	-0.01	0.03	0.01	-0.06	0.14	-0.03
restecg	-0.12	0.04	-0.11	-0.15	-0.08	1.00	0.04	-0.07	-0.06	0.09	-0.07	-0.01
thalachh	-0.40	0.30	-0.05	-0.01	-0.01	0.04	1.00	-0.38	-0.34	0.39	-0.21	-0.10
exng	0.10	-0.39	0.07	0.07	0.03	-0.07	-0.38	1.00	0.29	-0.26	0.12	0.21
oldpeak	0.21	-0.15	0.19	0.05	0.01	-0.06	-0.34	0.29	1.00	-0.58	0.22	0.21
slp	-0.17	0.12	-0.12	0.00	-0.06	0.09	0.39	-0.26	-0.58	1.00	-0.08	-0.10
caa	0.28	-0.18	0.10	0.07	0.14	-0.07	-0.21	0.12	0.22	-0.08	1.00	0.15
thall	0.07	-0.16	0.06	0.10	-0.03	-0.01	-0.10	0.21	0.21	-0.10	0.15	1.00

Above numerical representation of the variables in our data and their correlation. There are negative and positive correlations.

The two variables with the strongest correlation in the Heart Attack dataset are oldpeak and slp, showing a strong negative correlation of -0.58. This indicates that as ST depression (oldpeak) increases, the slope of the peak exercise ST segment (slp) tends to decrease. Since both features relate to the heart's response during exercise, this relationship is medically meaningful and highlights how changes in one measurement often reflect changes in the other.

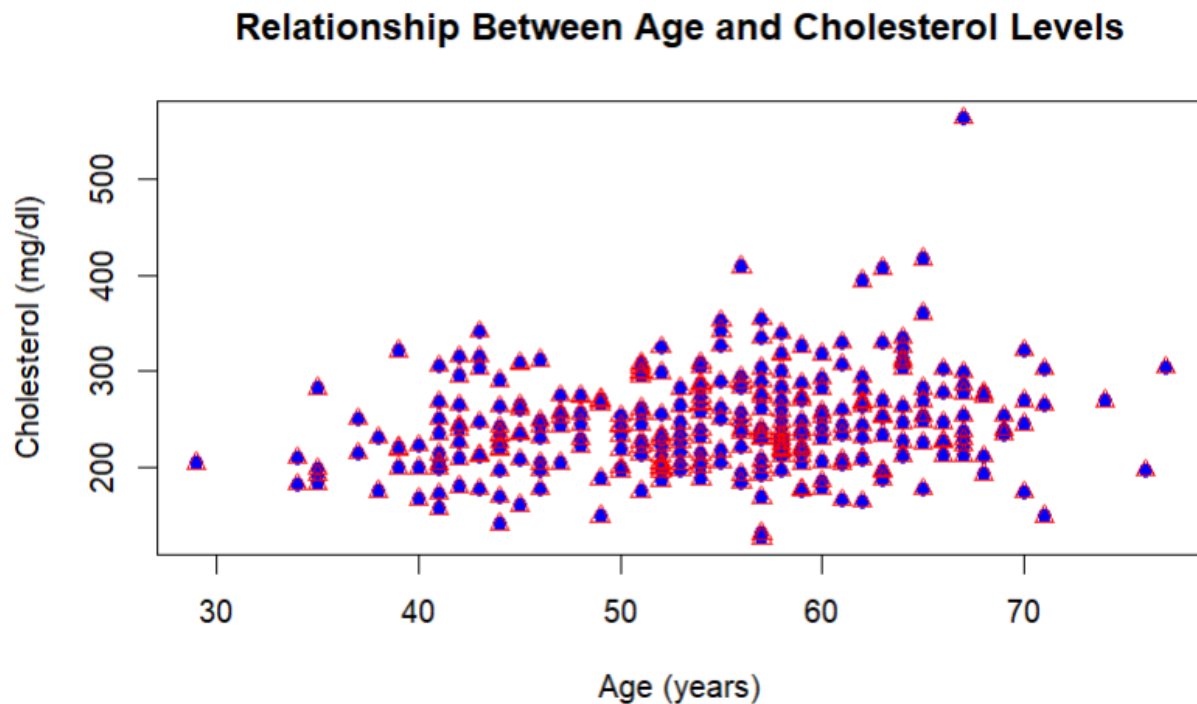
#### Part 4: Data Visualization (60 marks)

15. Create a scatter plot. The plot should show the relationship between the cholesterol and the age attributes (10 marks).

- Add labels, title, and color to the plot. The color should be blue.
- Add open red triangles to the plot.

```
plot(heart_data$age, heart_data$chol,
     main = "Relationship Between Age and Cholesterol Levels",
     xlab = "Age (years)",
     ylab = "Cholesterol (mg/dl)",
     col = "blue",      # Blue points
     pch = 19)          # pch=19 solid circles

# Overlay red open triangles (pch=2)
points(heart_data$age, heart_data$chol,
       col = "red",     # Red color for triangles
       pch = 2)         # pch=2 open triangles
```



This code creates a scatter plot showing the relationship between age and cholesterol levels. The first layer plots solid blue circles, and the second layer overlays open red triangles at the same points. Labels and title are added to describe the axes

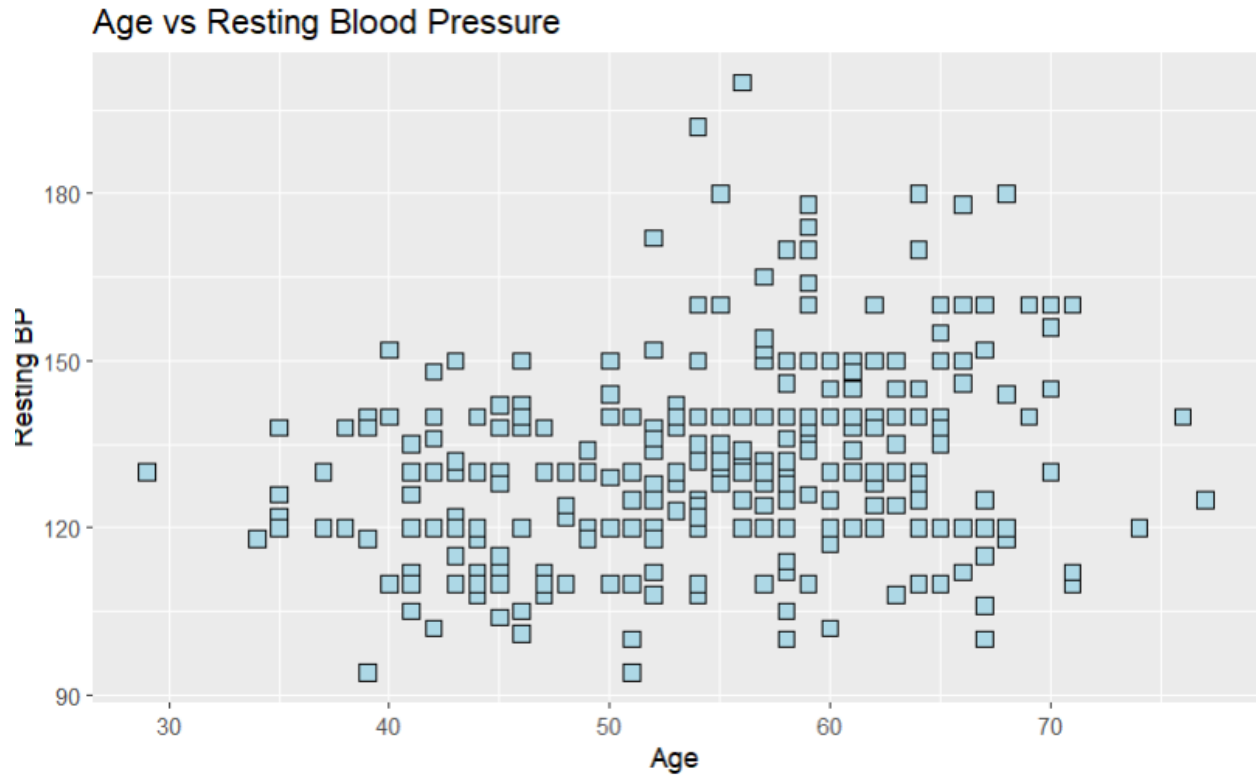
16. Use the ggplot function to plot any two variables (10 marks).

a. The points shape should be filled square.

Example 1:

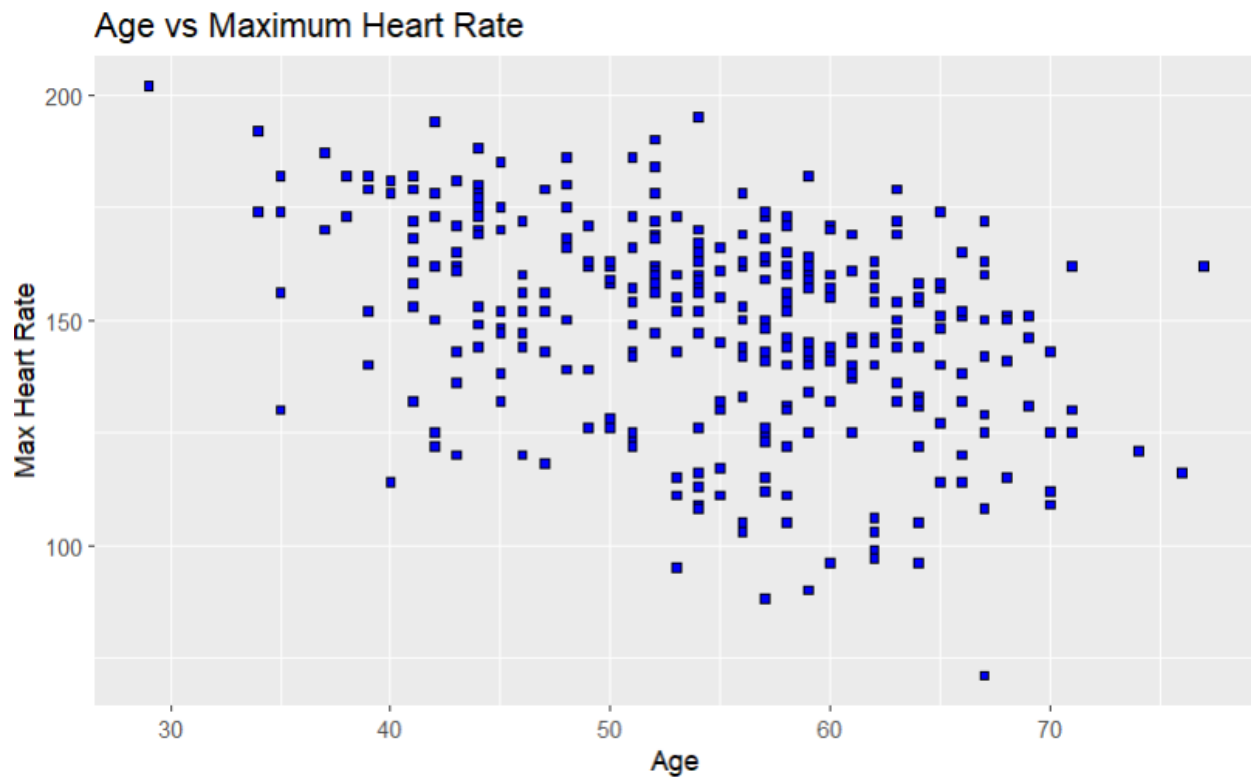
```
library(ggplot2)

# ggplot with filled squares (pch = 22)
ggplot(heart_data, aes(x = age, y = trtbps)) +
  geom_point(shape = 22, fill = "lightblue", color = "black", size = 3) +
  ggtitle("Age vs Resting Blood Pressure") +
  xlab("Age") +
  ylab("Resting BP")
```



Question 16: Example: 2

```
# Plot using ggplot2
ggplot(heart_data, aes(x = age, y = thalachh)) +
  geom_point(shape = 22, fill = "blue") +
  labs(title = "Age vs Maximum Heart Rate",
        x = "Age",
        y = "Max Heart Rate")
```

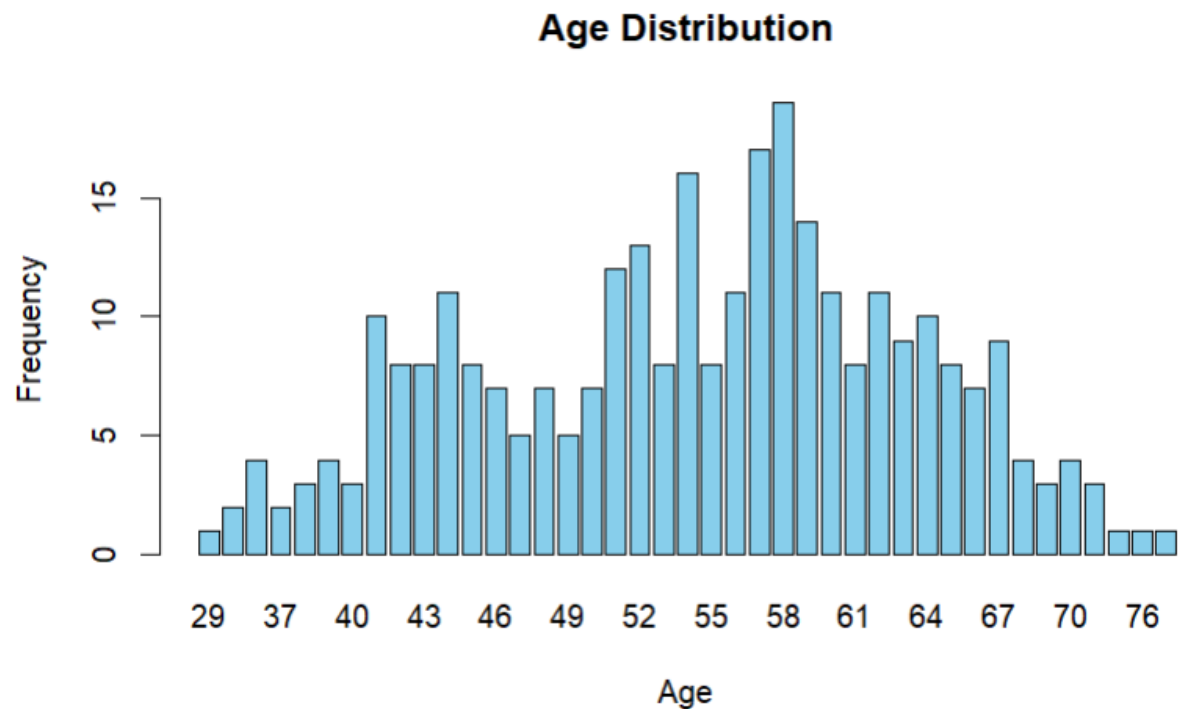


This chart shows how age relates to maximum heart rate achieved (thalachh). The square filled points (shape 22) help distinguish this plot visually.

17. barplot the 'age' variable of the Heart Attack dataset (10 marks):

a. Add labels, title, and color to the plot.

```
# Barplot of age
barplot(table(heart_data$age),
        main = "Age Distribution",
        xlab = "Age", ylab = "Frequency",
        col = "skyblue")
```



This shows the age distribution in our data. The highest frequency is of people aged between 58 to 61.

18. Create a histogram of the 'cp' attribute (10 marks):

a. Find the minimum and maximum of the attribute.

```
> cat("Minimum CP value:", min_cp, "\n")
Minimum CP value: 0
> cat("Maximum CP value:", max_cp, "\n")
Maximum CP value: 3
```

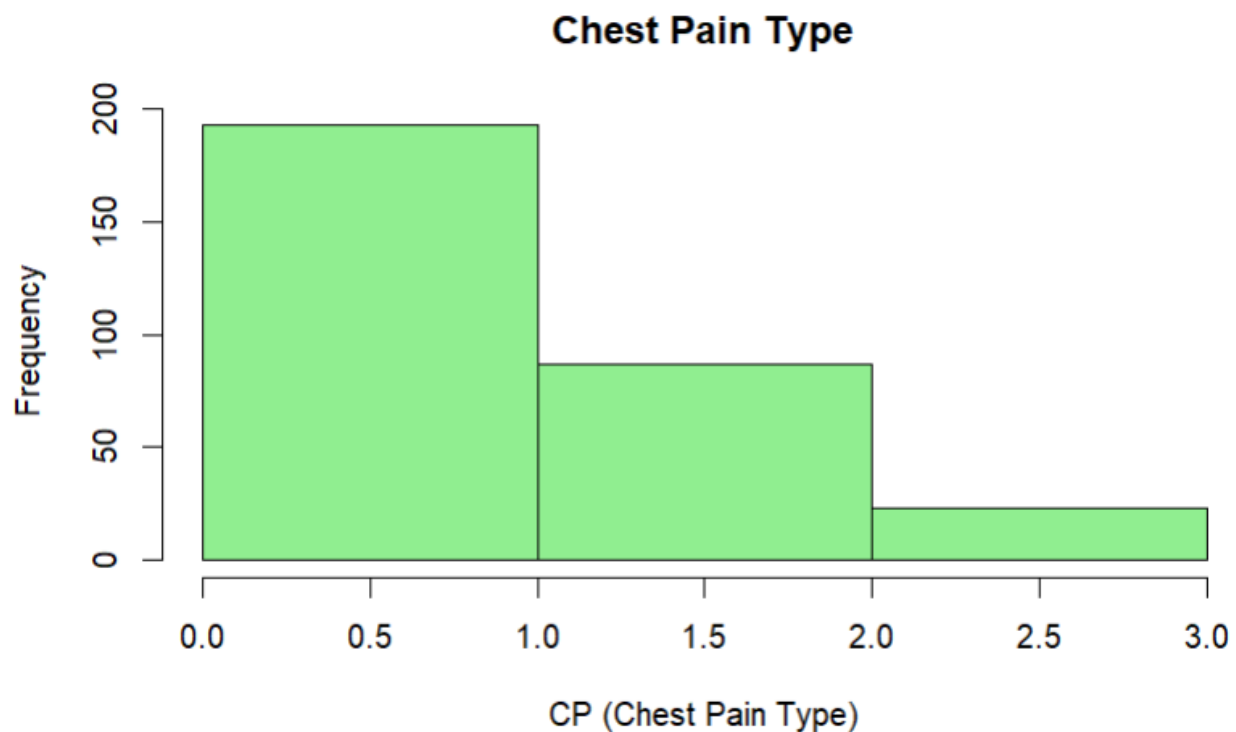
b. Add a break function and use the seq(x, y, z) function.

c. Add labels, title = (Chest Pain type), and color to the plot.

```
# Finding min and max of 'cp'
min_cp <- min(heart_data$cp)
max_cp <- max(heart_data$cp)

cat("Minimum CP value:", min_cp, "\n")
cat("Maximum CP value:", max_cp, "\n")

# Creating histogram with breaks|
hist(heart_data$cp,
     breaks = seq(min_cp, max_cp, 1),
     main = "Chest Pain Type",
     xlab = "CP (Chest Pain Type)", col = "lightgreen")
```

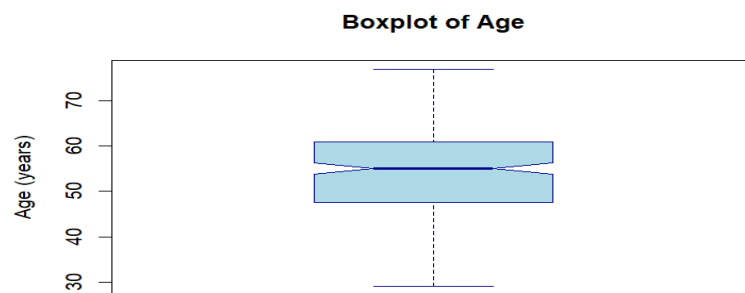


It controls how the histogram bins the data. Chest pain is usually **categorical (0, 1, 2, 3)** so `seq(min, max, 1)` ensures each type gets its own bin. The histogram of the `cp` (chest pain type) attribute reveals how chest pain categories are distributed among patients in the dataset. Each bar represents one of the four chest pain types: typical angina (0), atypical angina (1), non-anginal pain (2), and asymptomatic (3). From the plot, we can observe which chest pain type is most prevalent.

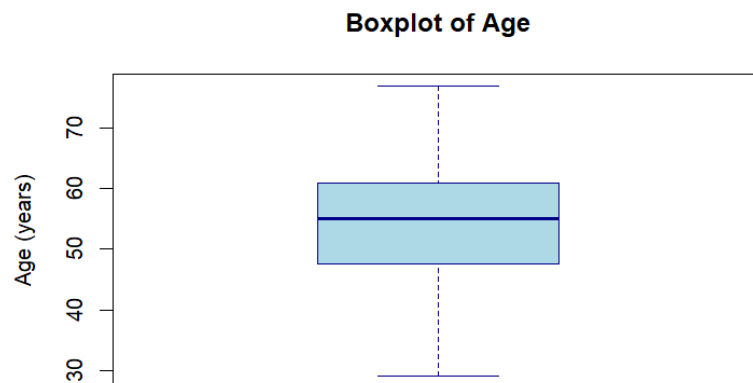
19. Boxplot the 'age' attribute and explain the output (10 marks).

```
boxplot(heart_data$age,  
        main = "Boxplot of Age",  
        ylab = "Age (years)",  
        col = "lightblue",  
        border = "darkblue",  
        notch = TRUE)
```

Output with Notch:



Output without Notch:



**Explanation:**



This boxplot visually summarizes the distribution of patient ages:

- The **box** captures the central 50% of the data (from Q1 to Q3).
- The **line inside the box** shows the **median age**.
- The **whiskers** stretch to the minimum and maximum values
- Any points beyond the whiskers are considered **outliers**, showing unusually young or old patients.

It gives a quick view of age spread and helps identify if the data is skewed or contains outliers.

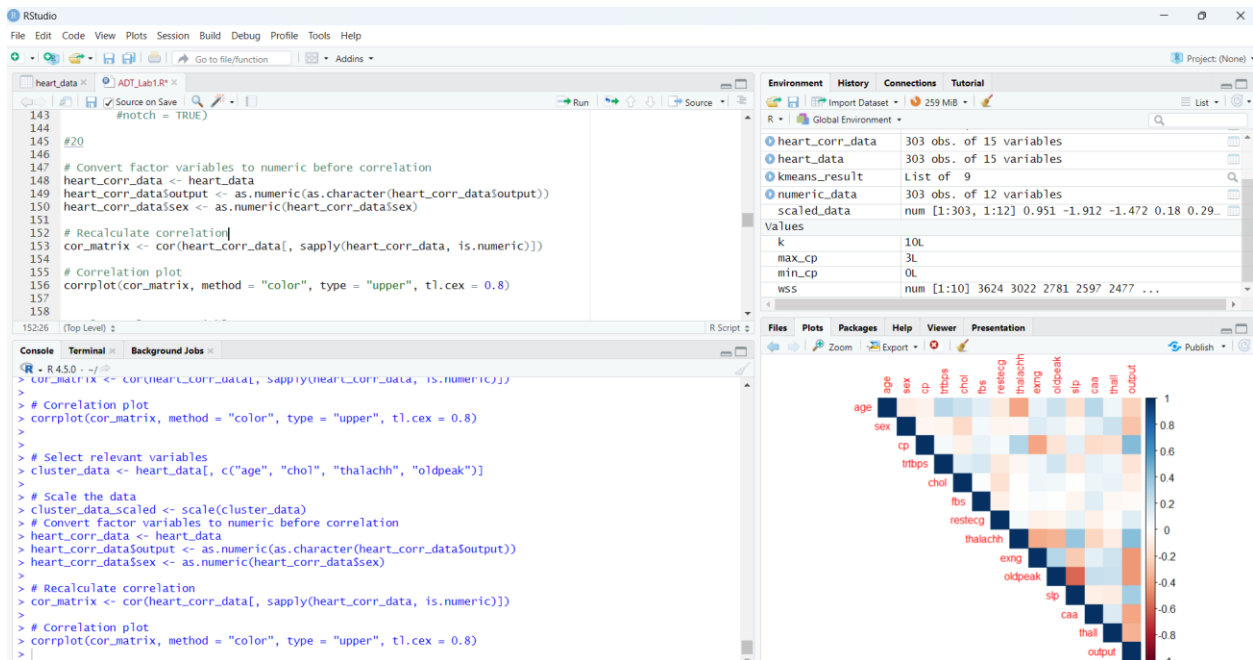
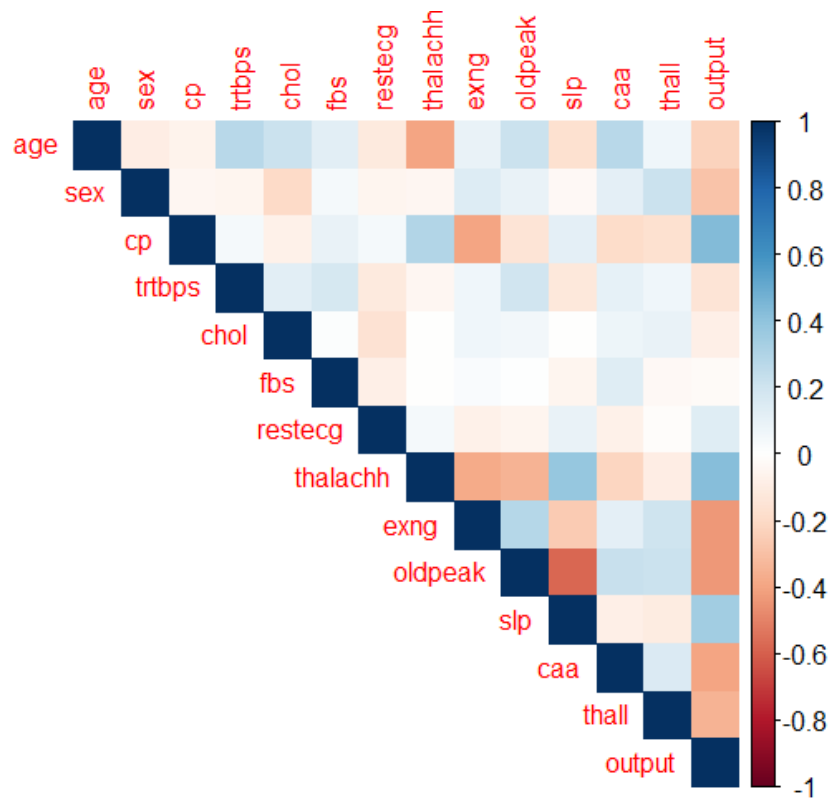
20. Create a correlation plot of the whole dataset variables and explain the output. Do not forget to convert some of the variable's datatype (10 marks).

```
# Convert factor variables to numeric before correlation
heart_corr_data <- heart_data
heart_corr_data$output <- as.numeric(as.character(heart_corr_data$output))
heart_corr_data$sex <- as.numeric(heart_corr_data$sex)

# Recalculate correlation
cor_matrix <- cor(heart_corr_data[, sapply(heart_corr_data, is.numeric)])

# Correlation plot
corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.8)
```

Output Image:



Explanation:

**Conversion:** Some categorical variables like sex or output might be stored as factors. These are converted to numeric to include them in the correlation analysis.

**Correlation Matrix:** `cor()` calculates Pearson correlation among numeric variables.

**`corrplot()`:**

`method = "color"` gives a color-based heatmap.


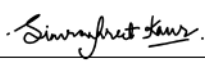
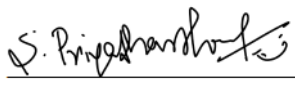
`type = "upper"` plots only the upper triangle of the matrix (avoids duplication).

`tl.cex = 0.8` adjusts the label size for better readability.

Place this confidentiality statement in your submission report.

**CONFIDENTIALITY AGREEMENT & STATEMENT OF HONESTY**

I, Group 14 verify that the submitted work is my own, original work, and that I did not use Generative AI tools (e.g., ChatGPT, Bard) to produce this lab report. I confirm knowing that a mark of 0 may be assigned for sharing or copying this work.

 Student Signature	<u>Aleena Ali Azeem</u> Student Name	<u>110190830</u> Student I.D.
 Student Signature	<u>Simranpreet Kaur</u> Student Name	<u>110189426</u> Student I.D.
 Student Signature	<u>Priyadharshan Reddy S</u> Student Name	<u>110191285</u> Student I.D.

---