# Heart disease Project

Moimadnan Djimtone

2024-11-17

## 1. Data Preparation

```r
# Load the necessary libraries
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(readr)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following objects are masked from 'package:Matrix':
##
##     expand, pack, unpack
```

## a. Download and Load the dataset

```
data_heart <- read_csv('heart.csv') #Here we load our dataset which is heart.csv
```

```
## Rows: 1025 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpea...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_heart # Here we display the first six of our dataset
```

```
## # A tibble: 1,025 x 14
##      age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##    <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1    52     1     0      125   212     0       1     168     0     1       2
##  2    53     1     0      140   203     1       0     155     1     3.1     0
##  3    70     1     0      145   174     0       1     125     1     2.6     0
##  4    61     1     0      148   203     0       1     161     0     0       2
##  5    62     0     0      138   294     1       1     106     0     1.9     1
##  6    58     0     0      100   248     0       0     122     0     1       1
##  7    58     1     0      114   318     0       2     140     0     4.4     0
##  8    55     1     0      160   289     0       0     145     1     0.8     1
##  9    46     1     0      120   249     0       0     144     0     0.8     2
## 10    54     1     0      122   286     0       0     116     1     3.2     1
## # i 1,015 more rows
## # i 3 more variables: ca <dbl>, thal <dbl>, target <dbl>
```

### The variables in the dataset are and i describe the brief description for each:

age: Give the age of patients which is the numeric type

sex: Give the sex of patients 1 for male and 0 for female which binary

cp: Means chest pain type which is categorical

trestbps: Means resting blood pressure (in mm Hg on admission to the hospital)

chol: Means serum cholesterol in mg/dL

fbs: Means fasting blood sugar > 120 mg/dL (1 = True, 0 = False).

restecg: Means resting electrocardiographic results (categorical variable)

thalach: Maximum heart rate achieved

exang: Exercise-induced angina (1 = Yes, 0 = No)

oldpeak: ST depression induced by exercise relative to rest.

slope: The slope of the peak exercise ST segment (categorical variable)

ca: Number of major vessels (0-3) colored by fluoroscop

thal: Thalassemia (categorical variable):

target: Presence of heart disease (1 = Presence, 0 = No heart disease)

## b. Select variables for analysis

age: Give the age of patients which is the numeric type

sex: Give the sex of patients 1 for male and 0 for female which binary

cp: Means chest pain type which is categorical

chol: Means serum cholesterol in mg/dL

target: Presence of heart disease (1 = Presence, 0 = No heart disease)

```
# Select the required variables for analysis
selected_data <- data_heart[, c("target", "age", "sex", "cp", "chol")]
selected_data
```

```
## # A tibble: 1,025 x 5
##    target   age   sex    cp  chol
##     <dbl> <dbl> <dbl> <dbl> <dbl>
##  1      0    52     1     0   212
##  2      0    53     1     0   203
##  3      0    70     1     0   174
##  4      0    61     1     0   203
##  5      0    62     0     0   294
##  6      1    58     0     0   248
##  7      0    58     1     0   318
##  8      0    55     1     0   289
##  9      0    46     1     0   249
## 10      0    54     1     0   286
## # i 1,015 more rows
```

## 2. Data Cleaning

**Remove rows with missing values, if there are any such**

```
#Firstly we are going to check the missing values if there are
sum(is.na(selected_data)) #After verification, there are no missing values so we procede so we don't ha
```

```
## [1] 0
```

We see that there is no missing values in our dataset, now we are going to check if there are duplicates data in our dataset, if there are we're going to remove theses duplicates data.

```
#Check the duplicate data
duplicate_data <- data_heart[duplicated(selected_data),]
duplicate_data
```

```
## # A tibble: 723 x 14
##      age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##    <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1    34     0     1      118   210     0       1     192     0     0.7     2
##  2    50     0     1      120   244     0       1     162     0     1.1     2
##  3    46     1     0      120   249     0       0     144     0     0.8     2
##  4    55     1     0      140   217     0       1     111     1     5.6     0
##  5    66     0     2      146   278     0       0     152     0     0       1
##  6    29     1     1      130   204     0       0     202     0     0       2
##  7    52     1     1      134   201     0       1     158     0     0.8     2
##  8    46     1     2      150   231     0       1     147     0     3.6     1
```

```
## 9     38     1     2      138   175     0       1     173     0     0        2
## 10    37     0     2      120   215     0       1     170     0     0        2
## # i 713 more rows
## # i 3 more variables: ca <dbl>, thal <dbl>, target <dbl>
```

We see now that there are more than 700 hundred data duplicate so we're going to remove all duplicate data in our dataset.

```r
#From here, we want to clean ou remove the duplicate data in our dataset
data_clean <- data_heart[!duplicated(selected_data), ]
data_clean
```

```
## # A tibble: 302 x 14
##      age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##    <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1    52     1     0      125   212     0       1     168     0     1       2
##  2    53     1     0      140   203     1       0     155     1     3.1     0
##  3    70     1     0      145   174     0       1     125     1     2.6     0
##  4    61     1     0      148   203     0       1     161     0     0       2
##  5    62     0     0      138   294     1       1     106     0     1.9     1
##  6    58     0     0      100   248     0       0     122     0     1       1
##  7    58     1     0      114   318     0       2     140     0     4.4     0
##  8    55     1     0      160   289     0       0     145     1     0.8     1
##  9    46     1     0      120   249     0       0     144     0     0.8     2
## 10    54     1     0      122   286     0       0     116     1     3.2     1
## # i 292 more rows
## # i 3 more variables: ca <dbl>, thal <dbl>, target <dbl>
```

Once remove our duplicate dataset, we have now 302 clear rows and 14 clear columns.

### Convert categorical variables (cp and sex) info facotrs

```r
# Convert the 'sex' variable into a factor
# 0 = Female, 1 = Male
data_clean$sex <- factor(data_clean$sex, levels = c(0, 1), labels = c("Female", "Male"))

data_clean$cp <- factor(data_clean$cp, levels = c(0, 1, 2, 3),
                        labels = c("Type 1", "Type 2", "Type 3", "Type 4"))

# Verify the changes
str(data_clean)
```

```
## tibble [302 x 14] (S3: tbl_df/tbl/data.frame)
##  $ age     : num [1:302] 52 53 70 61 62 58 58 55 46 54 ...
##  $ sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 2 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "Type 1","Type 2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ trestbps: num [1:302] 125 140 145 148 138 100 114 160 120 122 ...
##  $ chol    : num [1:302] 212 203 174 203 294 248 318 289 249 286 ...
##  $ fbs     : num [1:302] 0 1 0 0 1 0 0 0 0 0 ...
##  $ restecg : num [1:302] 1 0 1 1 1 0 2 0 0 0 ...
##  $ thalach : num [1:302] 168 155 125 161 106 122 140 145 144 116 ...
##  $ exang   : num [1:302] 0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num [1:302] 1 3.1 2.6 0 1.9 1 4.4 0.8 0.8 3.2 ...
##  $ slope   : num [1:302] 2 0 0 2 1 1 0 1 2 1 ...
##  $ ca      : num [1:302] 2 0 0 1 3 0 3 1 0 2 ...
##  $ thal    : num [1:302] 3 3 3 3 2 2 1 3 3 2 ...
```

```
##  $ target  : num [1:302] 0 0 0 0 0 1 0 0 0 0 ...
```

**Convert the target variable (target) into a factor.**

```
data_clean$target <- as.factor(data_clean$target)
str(data_clean)
```

```
## tibble [302 x 14] (S3: tbl_df/tbl/data.frame)
##  $ age     : num [1:302] 52 53 70 61 62 58 58 55 46 54 ...
##  $ sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 2 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "Type 1","Type 2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ trestbps: num [1:302] 125 140 145 148 138 100 114 160 120 122 ...
##  $ chol    : num [1:302] 212 203 174 203 294 248 318 289 249 286 ...
##  $ fbs     : num [1:302] 0 1 0 0 1 0 0 0 0 0 ...
##  $ restecg : num [1:302] 1 0 1 1 1 0 2 0 0 0 ...
##  $ thalach : num [1:302] 168 155 125 161 106 122 140 145 144 116 ...
##  $ exang   : num [1:302] 0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num [1:302] 1 3.1 2.6 0 1.9 1 4.4 0.8 0.8 3.2 ...
##  $ slope   : num [1:302] 2 0 0 2 1 1 0 1 2 1 ...
##  $ ca      : num [1:302] 2 0 0 1 3 0 3 1 0 2 ...
##  $ thal    : num [1:302] 3 3 3 3 2 2 1 3 3 2 ...
##  $ target  : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
```

**Provide the R code you used to clean the dataset.**

**The R code we used to clean the dataset are:** read_csv(), read_xlsl(), read_sql(): permit us to read our dataset

head() : generally display the first six rows of our dataset

str() : to check the structure of our dataset

is.na() : To check the missing values if there are exists

factor() : To convert a categorical variable in factor

duplicate(): To check if there are duplicates data in our dataset

na.omit(): this function manage the missing data that means we're going to use this commande to clean our dataset if there are missing values

**How many rows and columns are in the cleaned dataset?**

```
data_clean
```

```
## # A tibble: 302 x 14
##      age sex    cp     trestbps  chol   fbs restecg thalach exang oldpeak slope
##    <dbl> <fct>  <fct>     <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
##  1    52 Male   Type 1      125   212     0       1     168     0     1       2
##  2    53 Male   Type 1      140   203     1       0     155     1     3.1     0
##  3    70 Male   Type 1      145   174     0       1     125     1     2.6     0
##  4    61 Male   Type 1      148   203     0       1     161     0     0       2
##  5    62 Female Type 1      138   294     1       1     106     0     1.9     1
##  6    58 Female Type 1      100   248     0       0     122     0     1       1
##  7    58 Male   Type 1      114   318     0       2     140     0     4.4     0
##  8    55 Male   Type 1      160   289     0       0     145     1     0.8     1
##  9    46 Male   Type 1      120   249     0       0     144     0     0.8     2
```

```
## 10    54 Male   Type 1      122   286     0       0     116      1      3.2      1
## # i 292 more rows
## # i 3 more variables: ca <dbl>, thal <dbl>, target <fct>
```

After display, we see that we have 302 rows and 14 columns.

# 3. Exploratory Data Analysis

**Summarize the variables using the summary() function.**

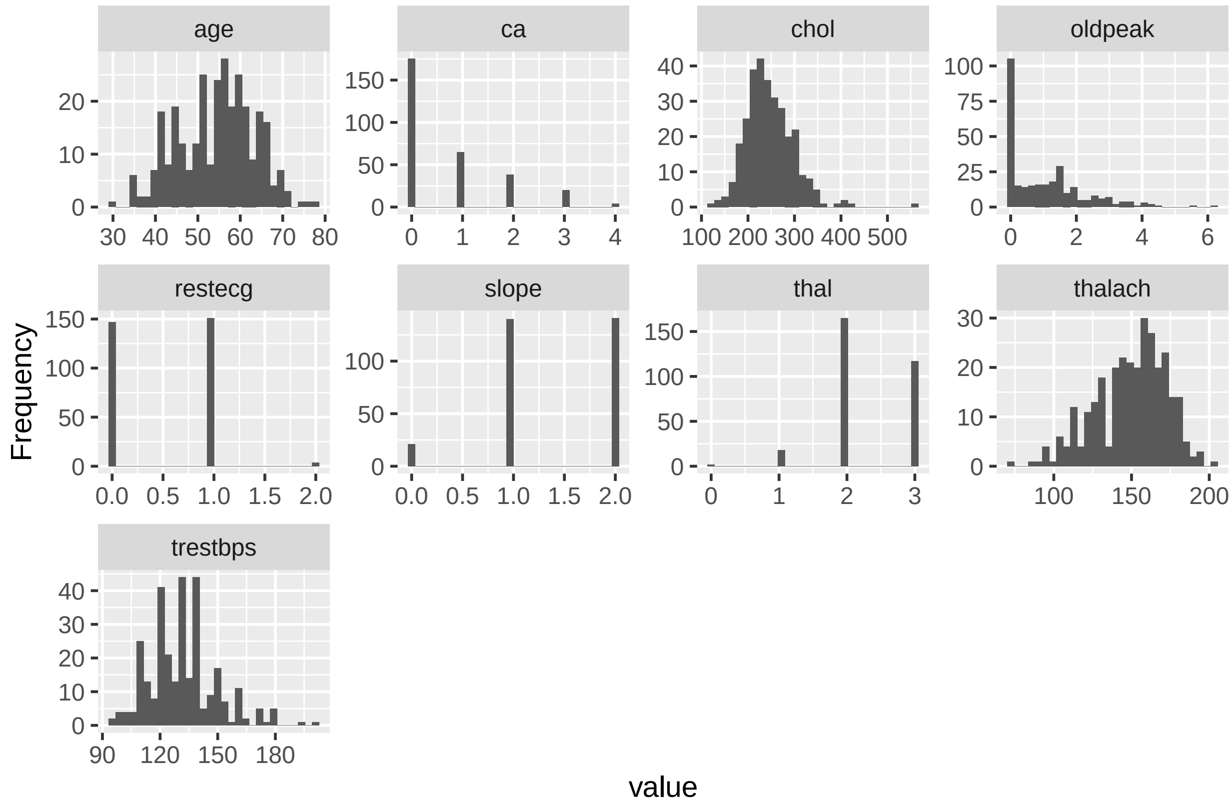
```
# Here we want to summarize our dataset in using summary()
summary(data_clean)
```

```
##       age             sex            cp         trestbps          chol      
##  Min.   :29.00   Female: 96   Type 1:143   Min.   : 94.0   Min.   :126.0  
##  1st Qu.:48.00   Male  :206   Type 2: 50   1st Qu.:120.0   1st Qu.:211.0  
##  Median :55.50                Type 3: 86   Median :130.0   Median :240.5  
##  Mean   :54.42                Type 4: 23   Mean   :131.6   Mean   :246.5  
##  3rd Qu.:61.00                             3rd Qu.:140.0   3rd Qu.:274.8  
##  Max.   :77.00                             Max.   :200.0   Max.   :564.0  
##       fbs           restecg          thalach          exang       
##  Min.   :0.000   Min.   :0.0000   Min.   : 71.0   Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:133.2   1st Qu.:0.0000  
##  Median :0.000   Median :1.0000   Median :152.5   Median :0.0000  
##  Mean   :0.149   Mean   :0.5265   Mean   :149.6   Mean   :0.3278  
##  3rd Qu.:0.000   3rd Qu.:1.0000   3rd Qu.:166.0   3rd Qu.:1.0000  
##  Max.   :1.000   Max.   :2.0000   Max.   :202.0   Max.   :1.0000  
##     oldpeak          slope             ca              thal       target 
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.000   0:138  
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000   1:164  
##  Median :0.800   Median :1.000   Median :0.0000   Median :2.000          
##  Mean   :1.043   Mean   :1.397   Mean   :0.7185   Mean   :2.315          
##  3rd Qu.:1.600   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.000          
##  Max.   :6.200   Max.   :2.000   Max.   :4.0000   Max.   :3.000          
```

With summary(), it's display some descriptive statistics like the mean, the median, the maximum, the third quartile, the minimum and the first quartile of each columns content the variables age, sex, cp, etc. We see that at each variable we describe the statistics. We apply this in our cleaned data.

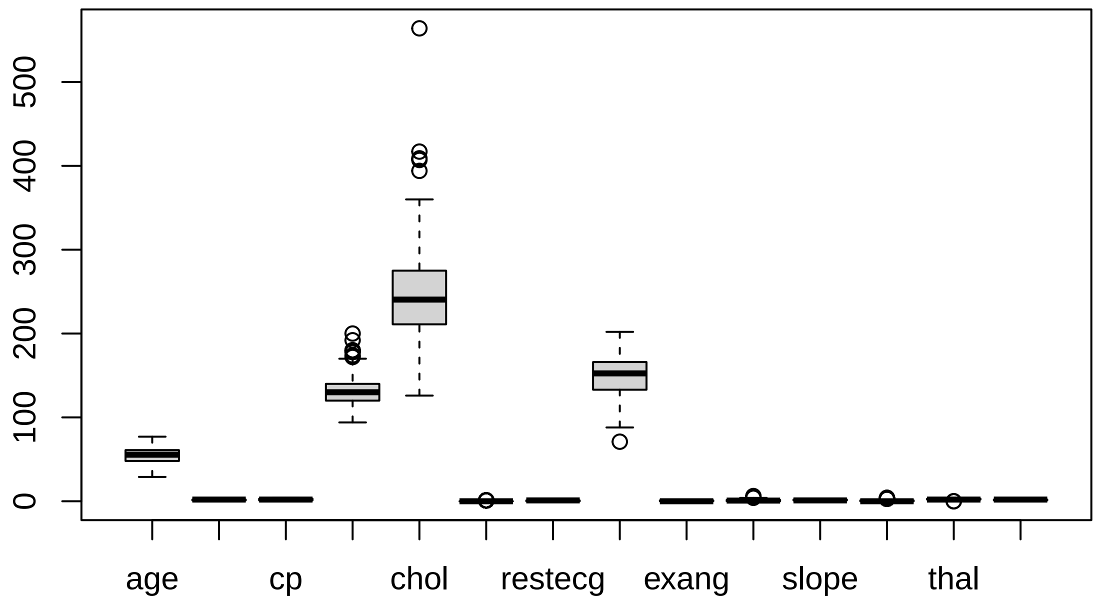
## Visualizing Distributions

```
#install.packages("DataExplorer")
library(DataExplorer)

# Visualize distributions
plot_histogram(data_clean)
```




Age : Displays a right-skewed distribution, indicating more younger individuals in the dataset. CA (Coronary Artery disease): Shows a bimodal distribution, suggesting two distinct groups based on artery blockage. Cholesterol (chol): Exhibits a peak around 200 mg/dL, suggesting this is a common cholesterol level among participants. Chest Pain Type(cp): The distribution is concentrated at specific values, indicating common types of chest pain experienced by individuals. Oldpeak : Shows a right-skewed distribution, indicating that most individuals have lower exercise-induced ST depression. Resting Electrocardiographic Results (restecg) : Displays a concentrated distribution, suggesting common findings in resting ECG. Slope: The distribution peaks at certain values, indicating prevalent slopes in exercise test responses. Thalassemia (thal): Shows a concentrated distribution with a few values, indicating common thalassemia levels. Maximum Heart Rate Achieved (thalach): Displays a normal distribution centered around 150 bpm. Resting Blood Pressure (trestbps): Also shows a normal distribution, suggesting a typical range of resting blood pressure among the population.

```
boxplot(data_clean, by = "target") # Replace 'target' with your dependent variable name
```



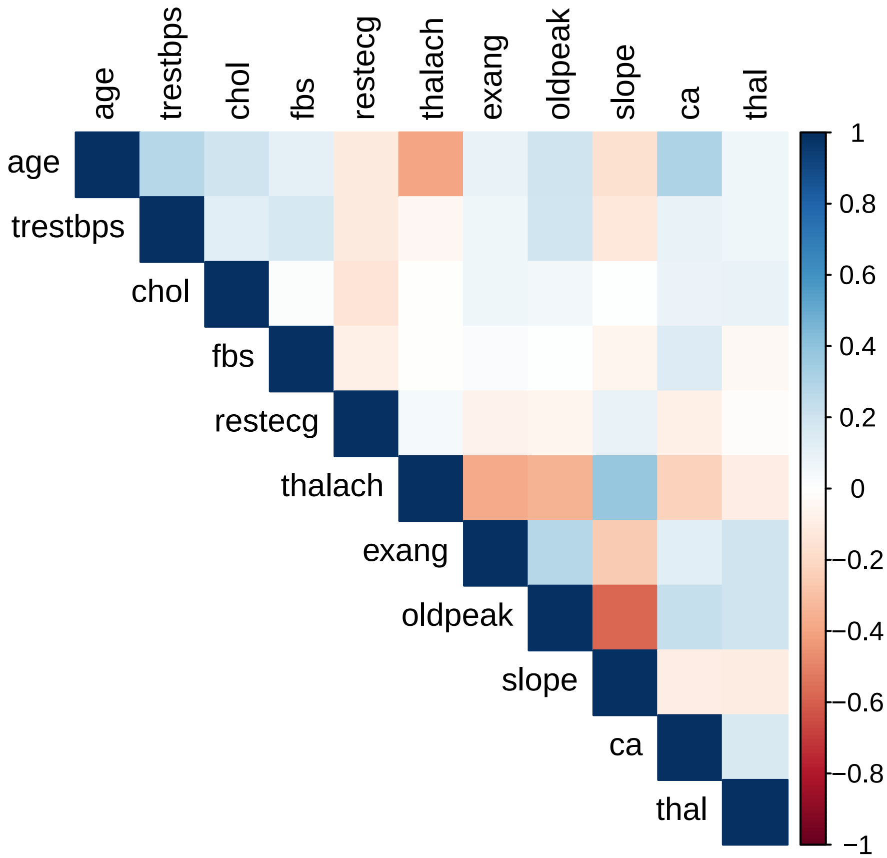

Age: The boxplot shows median age around 55, with a wider interquartile range, suggesting variability in age among individuals with heart disease. CA (coronary artery disease): The boxplot indicates a bimodal distribution, reflecting differing levels of artery blockage among the population. Cholesterol (chol): The median cholesterol level appears around 200 mg/dL, with some outliers, suggesting variability in cholesterol among individuals with heart disease. Chest Pain Type (cp): The boxplot indicates a range of chest pain types, with clear distinctions between categories. Oldpeak: Shows a higher median for individuals with heart disease, indicating more significant ST depression during exercise. Resting Electrocardiographic Results (restecg): The boxplot displays typical findings, with some variation in results. Slope: Reflects different slopes of exercise test responses, indicating variability in exercise tolerance. Thalassemia (thal): Shows a limited range, suggesting a commonality in thalassemia levels. Maximum Heart Rate Achieved (thalach): The boxplot indicates a median around 150 bpm, with variability among individuals. Resting Blood Pressure (trestbps): Displays a typical range of resting blood pressure, indicating overall cardiovascular health.

**Correlation for numerical variable**

```r
#install.packages("corrplot")  # Install if not already installed
library(corrplot)  # Load the package
```

```
## corrplot 0.95 loaded
```

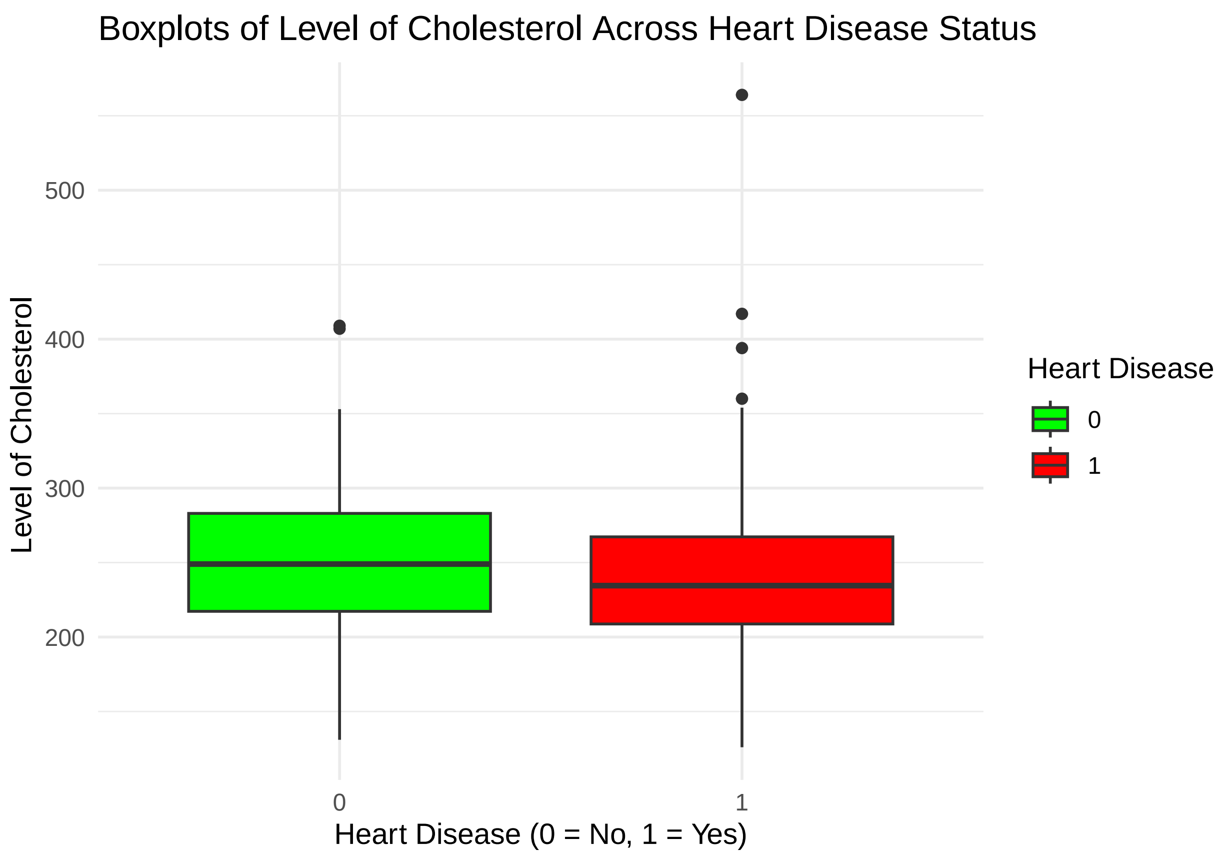
```r
# Compute the correlation matrix for numeric variables
library(dplyr)  # Ensure dplyr is loaded for select_if()
corr_matrix <- cor(select_if(data_clean, is.numeric))

# Plot the correlation heatmap
corrplot(corr_matrix, method = "color", type = "upper", tl.col = "black")
```




This explain the correlation between variables.

**Create boxplots of chol across levels of target.**

```r
# We have already load the ggplot2 in the beginning so don't need again here

# Assuming that our dataset is data_heart
# Convert target to a factor if not already done
#data_clean$target <- as.factor(data_clean$target)

# Here is our boxplot
ggplot(data_clean, aes(x = target, y = chol, fill = target)) +
  geom_boxplot() +
  labs(
    title = "Boxplots of Level of Cholesterol Across Heart Disease Status",
    x = "Heart Disease (0 = No, 1 = Yes)",
    y = "Level of Cholesterol "
  ) +
  theme_minimal() +
  scale_fill_manual(values = c("green", "red"), name = "Heart Disease")
```

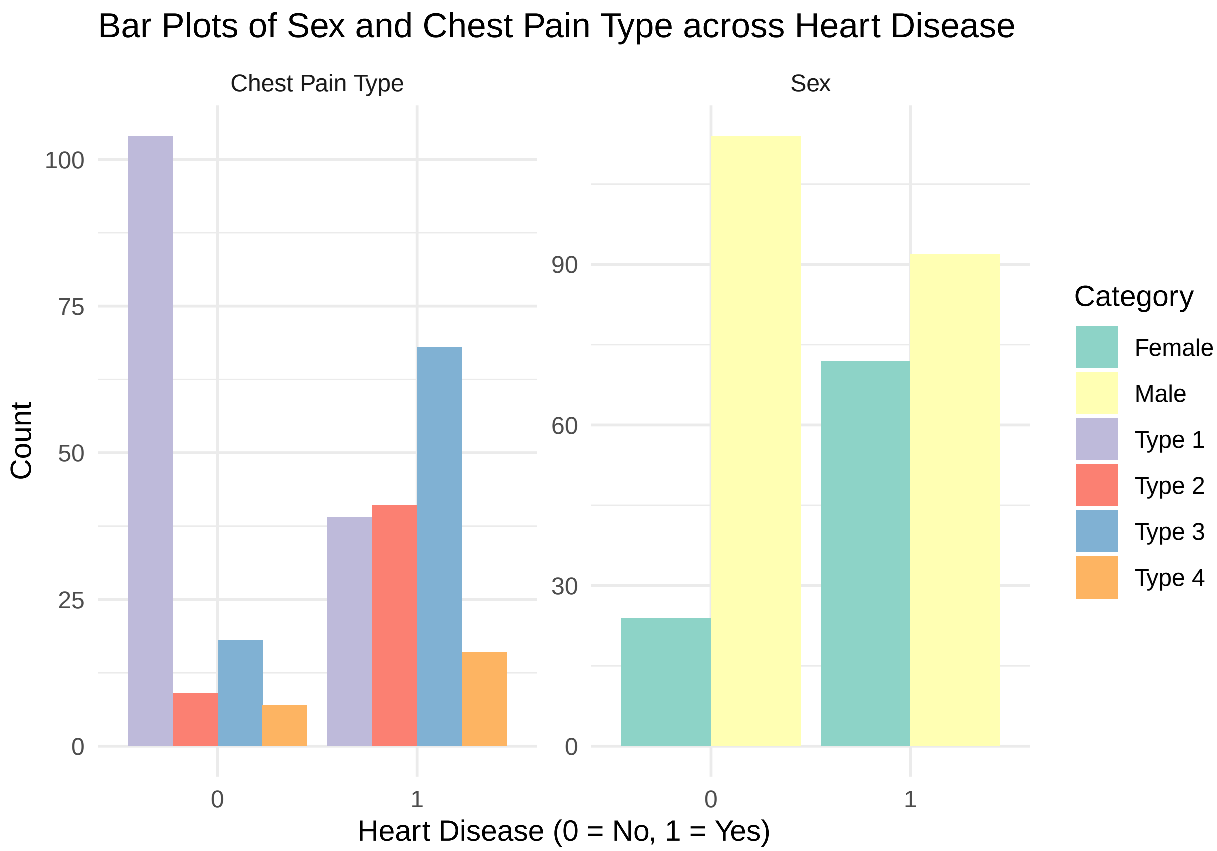



We see in our boxplots that the 1 represent the presence of heart disease and the 0 means lack of heart disease. The boxplots indicate the interquartile range means we have the IQR, 25th,75th in each representation but not the same like we see. We see in our second boxplot means the red boxplot contains the outliers more than the first(green) boxplot that means that the second boxplot has more variability cholesterol level, this represente that theses individuals who have the heart disease have extreme cholesterol values than those don't have heart disease. We remark also that in the first boxplot, the median is bigger than the second boxplot that means that the cholesterol is an important factor in this case.

**Create bar plots for sex and cp with respect to target.**

```
# Combine data for sex and cp with respect to target
data_sex_cp <- data_clean %>%
  pivot_longer(cols = c(sex, cp), names_to = "Variable", values_to = "Value")

# Create bar plots
library(ggplot2)
ggplot(data_sex_cp, aes(x = target, fill = as.factor(Value))) +  # Convert Value to factor
  geom_bar(position = "dodge") +
  facet_wrap(~ Variable, scales = "free", labeller = labeller(Variable = c(sex = "Sex", cp = "Chest Pai
  labs(
    title = "Bar Plots of Sex and Chest Pain Type across Heart Disease",
    x = "Heart Disease (0 = No, 1 = Yes)",
    y = "Count",
    fill = "Category"
  ) +
  theme_minimal() +
```

```r
  scale_fill_brewer(palette = "Set3")
```




We see here in our bar plot that the first one represente cp or Chest Pain types and the second one represente the sex. After our obervation, we remark that in cp for No disease heart, the type 1 indicate the count more than 100 so we can say that this type is less to find disease respect to type 2, type 3 and type 4. In cp again for Yes disease heart, we observe that the type 3 has more disease follows by type 2, type 1 and type 4.

In second bar plot, comparing sex respect to target, we observe that the percentage of Male doesn't have the disease spend largelly the percentage of female doesn't have disease and the percentage of male has disease is no too most comparing respect to female.

# 4. Fitting a Logistic Regression Model

**Fit a logistic regression model to predict target using age, sex, cp, and chol as predictors.**

```r
#We fit logistic regression model respect to target
logistic_model <- glm(target ~ age + sex + cp + chol, data = data_clean, family = "binomial")

# View model summary
summary(logistic_model)
```

```
##
## Call:
## glm(formula = target ~ age + sex + cp + chol, family = "binomial",
##     data = data_clean)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3720  -0.7044   0.2467   0.7322   2.2976
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.956253   1.275701   3.885 0.000102 ***
## age         -0.063368   0.017892  -3.542 0.000398 ***
## sexMale     -1.891184   0.361858  -5.226 1.73e-07 ***
## cpType 2     2.538803   0.448243   5.664 1.48e-08 ***
## cpType 3     2.360416   0.356639   6.619 3.63e-11 ***
## cpType 4     2.237305   0.535818   4.175 2.97e-05 ***
## chol        -0.004796   0.002862  -1.676 0.093777 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 416.42  on 301  degrees of freedom
## Residual deviance: 289.87  on 295  degrees of freedom
## AIC: 303.87
##
## Number of Fisher Scoring iterations: 5
```

In the course, we have seen that the coefficients represent the log-odds of survival for a unit change in the predictor. So if the coefficients are positive that means the predictor increases the likelihood of heart disease and if the the coefficients are negative that's increase in the predictor decreases the likelihood of heart disease. In this case, we are going to conclude that the cpType 2, cpType 3, cpType 4 have strong positive effect, this indicating that the three cpType are much more likely to have heart disease compare to others which are negative in the estimate.

# 5. Model Interpretation

**Convert the coefficients into odds ratios using exp(coef()).**

```r
# Conversion of coefficients to odds ratios
exp(coef(logistic_model))
```

```
## (Intercept)         age     sexMale    cpType 2    cpType 3    cpType 4
## 142.0605407   0.9385984   0.1508931  12.6645085  10.5953589   9.3680525
##        chol
##   0.9952157
```

```r
# from here, we have our Confidence intervals for odds ratios
exp(confint(logistic_model))
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %       97.5 %
## (Intercept) 12.39951817 1877.1163550
## age          0.90534242    0.9713586
## sexMale      0.07222334    0.2997431
## cpType 2     5.46958937   32.1088300
## cpType 3     5.38511840   21.9041165
## cpType 4     3.40021415   28.3545609
```

```
## chol          0.98957270    1.0008492
```

After our conversion, we observe that the value of cpType 2, cpType 3, cpType 4 are greater than one so the predictor increases the likelihood of heart disease. sexMale has lower odds (0.095) of heart disease than sexFemale. For chol, the predictor decreases the likelihood of heart disease.

# 6. Model Comparison

```r
# We fit our reduce model
reduced_model <- glm(target ~ age + sex + cp, data = data_clean, family = binomial)
summary(reduced_model)
```

```
##
## Call:
## glm(formula = target ~ age + sex + cp, family = binomial, data = data_clean)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2957  -0.6932   0.2717   0.7482   2.1879
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.82561    1.06000   3.609 0.000307 ***
## age         -0.06629    0.01767  -3.752 0.000176 ***
## sexMale     -1.74839    0.34554  -5.060 4.20e-07 ***
## cpType 2     2.51343    0.44510   5.647 1.63e-08 ***
## cpType 3     2.36261    0.35465   6.662 2.70e-11 ***
## cpType 4     2.26088    0.53246   4.246 2.18e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 416.42  on 301  degrees of freedom
## Residual deviance: 292.65  on 296  degrees of freedom
## AIC: 304.65
##
## Number of Fisher Scoring iterations: 5
```

**Perform a likelihood ratio test between the full and reduced models using anova().**

```r
# Here we want to perform the full model with more predictors
full_model <- glm(target ~ age + sex + cp + chol + thalach, data = data_clean, family = binomial)

# Perform the Likelihood Ratio Test
anova(reduced_model, full_model, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: target ~ age + sex + cp
## Model 2: target ~ age + sex + cp + chol + thalach
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       296     292.65
## 2       294     271.78  2   20.869 2.94e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Including chol significantly improve the model because refer to the course, you say that if Low p-value (< 0.05), that means that chol provides significant additional information.

# 7. Model Predictions and Performance

**Predict probabilities of heart disease for all individuals.**

```r
# Predict probabilities using the full model (or reduced model)
predicted_probs <- predict(full_model, type = "response")

# Display the first few predicted probabilities
head(predicted_probs)
```

```
##          1          2          3          4          5          6
## 0.37665520 0.28483951 0.08637026 0.26767533 0.17311653 0.36140900
```

```r
# Add the predicted probabilities to the dataset
data_clean$predicted_probabilities <- predicted_probs
```

```r
predicted_probs
```

```
##          1          2          3          4          5          6          7
## 0.37665520 0.28483951 0.08637026 0.26767533 0.17311653 0.36140900 0.08484723
##          8          9         10         11         12         13         14
## 0.12937600 0.20624540 0.05553807 0.43136562 0.45675290 0.99387929 0.06902326
##         15         16         17         18         19         20         21
## 0.29792610 0.88678348 0.05034135 0.96328017 0.78267933 0.73955570 0.48760433
##         22         23         24         25         26         27         28
## 0.28863532 0.95327500 0.98363660 0.43581678 0.89238476 0.89450721 0.56788364
##         29         30         31         32         33         34         35
## 0.23902096 0.32279842 0.12559726 0.20799187 0.83244388 0.72539649 0.49850823
##         36         37         38         39         40         41         42
## 0.33706968 0.03184585 0.66567276 0.81902925 0.62286211 0.59098064 0.92871696
##         43         44         45         46         47         48         49
## 0.52582375 0.71045062 0.67546261 0.88648503 0.23423222 0.92268619 0.38878509
##         50         51         52         53         54         55         56
## 0.92527993 0.70659815 0.07054930 0.76876277 0.87988310 0.60840659 0.77033705
##         57         58         59         60         61         62         63
## 0.97583153 0.04371527 0.72457316 0.58818909 0.75304459 0.09210681 0.98425284
##         64         65         66         67         68         69         70
## 0.65182892 0.07814330 0.14027582 0.15324998 0.32595784 0.22039554 0.74396594
##         71         72         73         74         75         76         77
## 0.84397284 0.17533450 0.80134011 0.79508738 0.66789630 0.87695775 0.52495956
##         78         79         80         81         82         83         84
## 0.60847327 0.13343808 0.95569548 0.71844605 0.59290765 0.73070397 0.58758462
##         85         86         87         88         89         90         91
## 0.02980907 0.17458995 0.73384535 0.97729225 0.89628398 0.63019624 0.91230510
##         92         93         94         95         96         97         98
## 0.08023854 0.29332628 0.20323647 0.57382415 0.07107116 0.54757831 0.07135354
##         99        100        101        102        103        104        105
## 0.08675408 0.53677046 0.13077089 0.17193748 0.77970923 0.10505725 0.79124212
##        106        107        108        109        110        111        112
```

```
## 0.10916906 0.95255172 0.10086137 0.82545929 0.22634146 0.93357979 0.59823909
##        113        114        115        116        117        118        119
## 0.44629559 0.80151905 0.02164228 0.51632361 0.54245441 0.92253306 0.24653978
##        120        121        122        123        124        125        126
## 0.16796789 0.93424052 0.45566294 0.76620864 0.85935951 0.88381584 0.23024746
##        127        128        129        130        131        132        133
## 0.05163873 0.90740463 0.59382115 0.94785431 0.09745047 0.67152235 0.30129856
##        134        135        136        137        138        139        140
## 0.06930455 0.53715775 0.73639205 0.96065585 0.42301563 0.03466971 0.79615957
##        141        142        143        144        145        146        147
## 0.98873735 0.10151477 0.40343159 0.02644039 0.34906915 0.54524725 0.21987723
##        148        149        150        151        152        153        154
## 0.82613288 0.79247069 0.06351656 0.32232453 0.89189588 0.78346227 0.35577412
##        155        156        157        158        159        160        161
## 0.02725441 0.21847337 0.50127518 0.90689768 0.14488261 0.61005878 0.87095179
##        162        163        164        165        166        167        168
## 0.93397903 0.11975358 0.81901960 0.06632273 0.07072307 0.96994470 0.45396807
##        169        170        171        172        173        174        175
## 0.78342005 0.95210624 0.90460876 0.84479341 0.11755112 0.96129102 0.73224666
##        176        177        178        179        180        181        182
## 0.55888809 0.59815842 0.93858485 0.79269648 0.92962303 0.96829902 0.05870048
##        183        184        185        186        187        188        189
## 0.25277054 0.37693863 0.93883480 0.90691375 0.86093078 0.54596429 0.01096446
##        190        191        192        193        194        195        196
## 0.81241017 0.77125980 0.39916058 0.85549743 0.10582587 0.34272351 0.98374921
##        197        198        199        200        201        202        203
## 0.98120922 0.93163453 0.98520804 0.77723577 0.89284531 0.93140997 0.08922418
##        204        205        206        207        208        209        210
## 0.50243159 0.04561470 0.73904026 0.98232820 0.74166452 0.97553652 0.02732323
##        211        212        213        214        215        216        217
## 0.91765277 0.69888150 0.84358242 0.55396153 0.90767128 0.16649716 0.33167373
##        218        219        220        221        222        223        224
## 0.04693801 0.02003594 0.66889836 0.92595487 0.95772113 0.12200364 0.63900191
##        225        226        227        228        229        230        231
## 0.13230757 0.88246618 0.81408494 0.22101452 0.60715714 0.11556309 0.71407752
##        232        233        234        235        236        237        238
## 0.04065807 0.73484877 0.58876944 0.81751330 0.76031720 0.84440084 0.67142829
##        239        240        241        242        243        244        245
## 0.58684790 0.66619896 0.97644457 0.77054224 0.23982144 0.05230778 0.13957440
##        246        247        248        249        250        251        252
## 0.31788539 0.56810181 0.52688469 0.49756617 0.48531286 0.47396885 0.49530233
##        253        254        255        256        257        258        259
## 0.96696658 0.87101069 0.73586560 0.28090309 0.97597439 0.07712264 0.86683233
##        260        261        262        263        264        265        266
## 0.29002471 0.32578992 0.95782414 0.48802469 0.51914143 0.87844549 0.11899552
##        267        268        269        270        271        272        273
## 0.24209651 0.26352751 0.22371748 0.64017090 0.93725992 0.50051362 0.61734942
##        274        275        276        277        278        279        280
## 0.72977896 0.96200327 0.89434190 0.50103218 0.18297494 0.71798201 0.94911156
##        281        282        283        284        285        286        287
## 0.82786605 0.68502770 0.10355579 0.97588834 0.09703158 0.52441596 0.41448416
##        288        289        290        291        292        293        294
## 0.88107053 0.66556092 0.12133236 0.15585145 0.71979264 0.95312336 0.26312762
##        295        296        297        298        299        300        301
```

```
## 0.73916662 0.20487832 0.49966232 0.76424402 0.98950151 0.26339999 0.33267749
##        302
## 0.09274473
```

**Convert these probabilities into binary predictions using a threshold of 0.5.**

In this case, we can apply a simple thresholding operation to predict probabilities into binary predictions using a threshold of 0.5.

```r
# Convert probabilities to binary predictions using a threshold of 0.5
predicted_class <- ifelse(predicted_probs >= 0.5, 1, 0)


# we add the binary predictions to the dataset
data_clean$predicted_class <- predicted_class

# we display the first few rows with the predicted probabilities and binary predictions
head(data_clean[, c("predicted_probabilities", "predicted_class")])
```

```
## # A tibble: 6 x 2
##   predicted_probabilities predicted_class
##                     <dbl>           <dbl>
## 1                  0.377                0
## 2                  0.285                0
## 3                  0.0864               0
## 4                  0.268                0
## 5                  0.173                0
## 6                  0.361                0
```

**Create a confusion matrix to evaluate the model's performance.**

```r
#We fit the full logistic regression model
# We're predicting the target variable (presence of heart disease)
full_model <- glm(target ~ age + sex + cp + chol + thalach, data = data_clean, family = binomial)

# Predict probabilities using the fitted model
predicted_probs <- predict(full_model, type = "response")

# Convert probabilities to binary predictions using a threshold of 0.5
predicted_class <- ifelse(predicted_probs >= 0.5, 1, 0)

# Add the binary predictions to the dataset
data_clean$predicted_class <- predicted_class

# Create the confusion matrix by comparing actual values with predicted values
confusion_matrix <- table(Predicted = data_clean$predicted_class, Actual = data_clean$target)

# Print the confusion matrix
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(confusion_matrix)
```

```
##          Actual
## Predicted   0   1
```

```
##          0  99  28
##          1  39 136
```

```r
# Calculate performance metrics

# Accuracy: (TP + TN) / Total
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

# Precision: TP / (TP + FP)
precision <- confusion_matrix[2, 2] / sum(confusion_matrix[2, ])

# Recall: TP / (TP + FN)
recall <- confusion_matrix[2, 2] / sum(confusion_matrix[, 2])

# F1-Score: 2 * (Precision * Recall) / (Precision + Recall)
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the metrics
cat("Accuracy: ", accuracy, "\n")
```

```
## Accuracy:  0.7781457
```

```r
cat("Precision: ", precision, "\n")
```

```
## Precision:  0.7771429
```

```r
cat("Recall: ", recall, "\n")
```

```
## Recall:  0.8292683
```

```r
cat("F1-Score: ", f1_score, "\n")
```

```
## F1-Score:  0.8023599
```

The accuracy of the model is 77% and we have 39 False Positive and 28 False Negative.

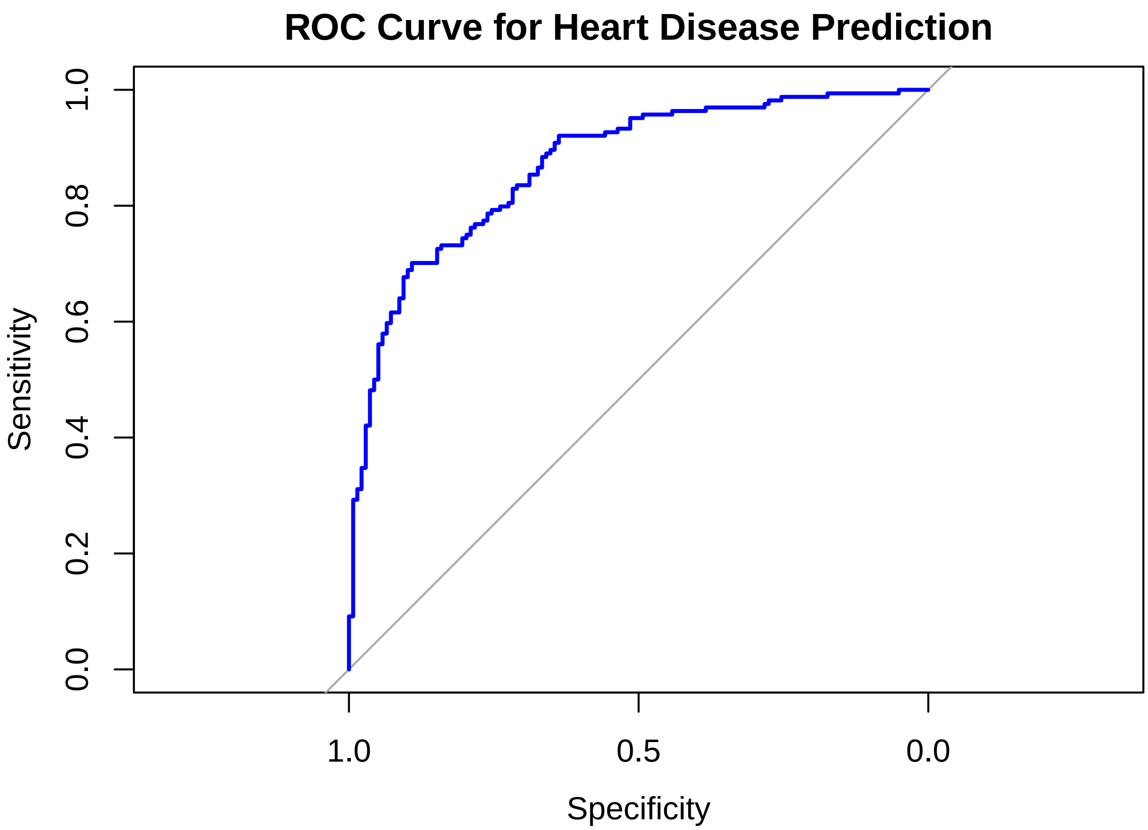
# 8. ROC Curve and AUC

```r
# Install the pROC package if you haven't already
#install.packages("pROC")

# Load the pROC package
library(pROC)



# Generate the ROC curve
roc_curve <- roc(data_clean$target, predicted_probs)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
# Plot the ROC curve
plot(roc_curve, main = "ROC Curve for Heart Disease Prediction", col = "blue", lwd = 2)
```




```r
# Calculate the AUC
auc_value <- auc(roc_curve)

# Display the AUC value
auc_value
```

```
## Area under the curve: 0.8701
```

The ROC curve summarizes the trade-off between sensitivity and specificity at different thresholds. The Area Under Curve (AUC) quantifies the model's discriminative ability. Our AUC = 87% which indicates that the model is quite effective at distinguishing between the two classes and has a strong ability to predict heart disease presence or absence.

Reference:


https://en.wikipedia.org/wiki/Classification

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

Lecture 3: Classification