

**American International University-Bangladesh (AIUB)  
Department of Computer Science  
Faculty of Science & Technology (FST)**

**Spring 23-24**

**Introduction to Data Science**

**(Mid Term Project)  
Section: B**

**Group no: 13**

**Submitted By:**

|  |  |  |
| --- | --- | --- |
| SN | Student Name | Student ID |
| 1 | Most. Lailun Nahar Aurthy | 20-43997-2 |
| 2 | Md. Iftear Hossain Ratul | 21-45678-3 |

**Data Set Description:**

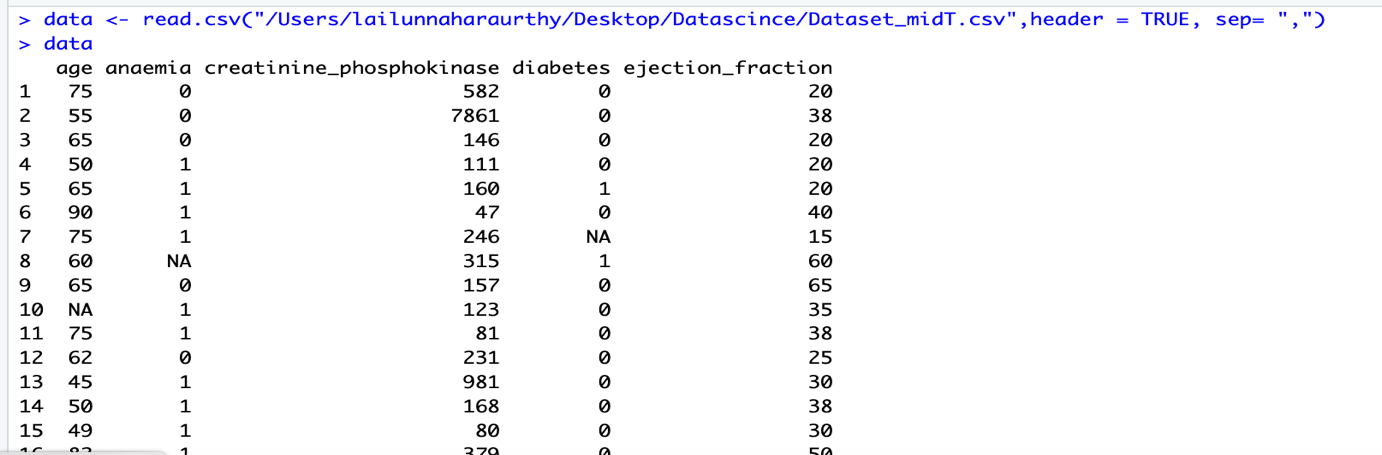
The Heart Failure Dataset comprises medical records from 299 patients who experienced heart failure, collected over their follow-up period. Each patient profile includes 13 clinical features, such as age, gender, and various health indicators like blood pressure and blood test results. These features help to characterize the patients' health status and potential risk factors for heart failure. Crucially, the dataset also indicates whether each patient passed away during the follow-up period, providing valuable insights into mortality rates and outcomes associated with heart failure. This dataset serves as a valuable resource for researchers aiming to understand the factors influencing heart failure and its consequences on patient health.

1. **Read data from csv file and show dataset table :**

**Code:** data <- read.csv("/Users/lailunnaharaurthy/Desktop/Datascince/Dataset\_midT.csv",header = TRUE, sep= ",")

data

**Output:**

****

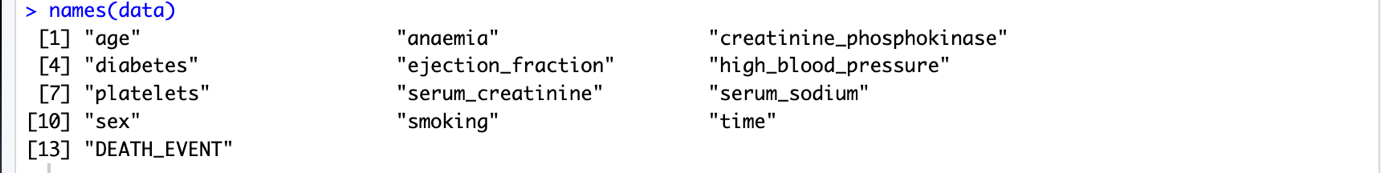
**Description:** The code reads a CSV file named "Dataset\_midT.csv", storing it as a dataframe named "data". It then displays the contents of the dataframe.

1. **Show all column names, descriptive summary and structure of the dataset:**

**Column names:**

**Code:** names(data)

**Output:**

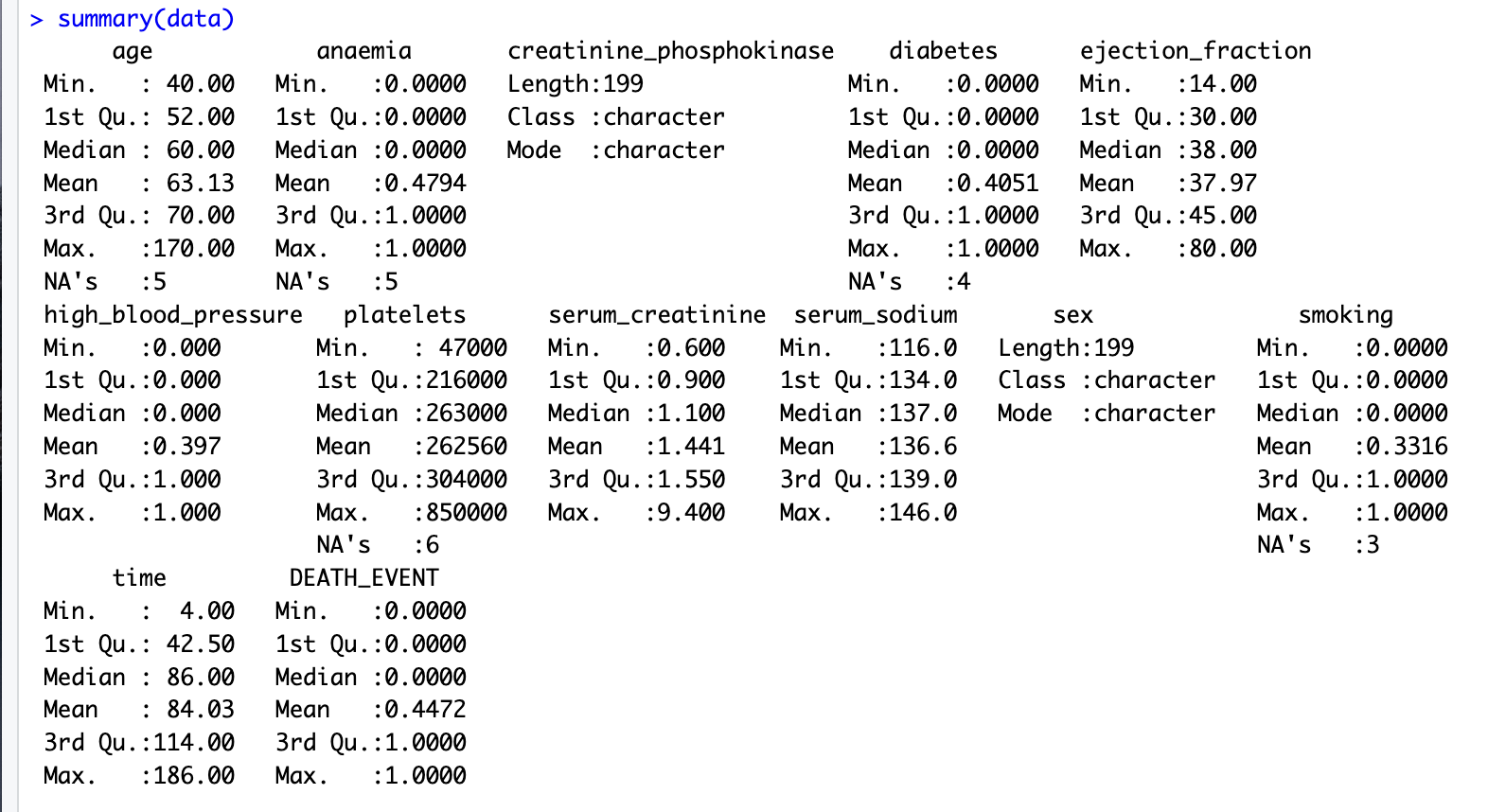


**Description:** The “names(data)” function returns the column names of the dataframe "data"

**Descriptive summary:**

**Code:** summary(data)

**Output:**

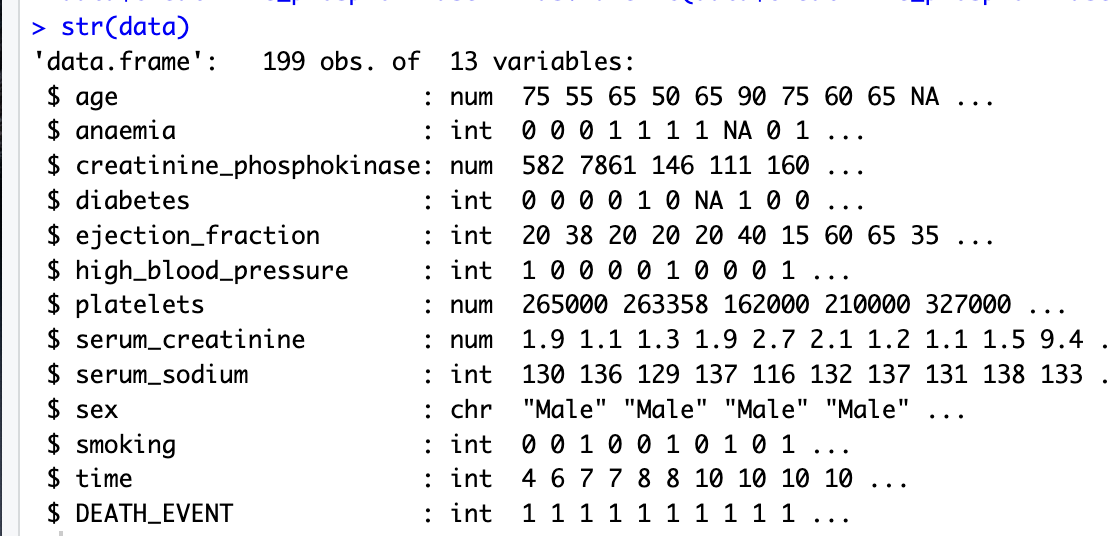


**Description:** summary(data) provides a summary of the descriptive statistics for each variable in the dataframe "data", including count, mean, median, minimum, maximum, and quartiles for numerical variables, and frequency counts for categorical variables.

**Structure:**

**Code:** str(data)

**Output:**



**Description:** str(data) displays the structure of the dataframe "data", providing information about the type and format of each variable/column, as well as the first few observations.

1. **Find invalid value and handle invalid value:**

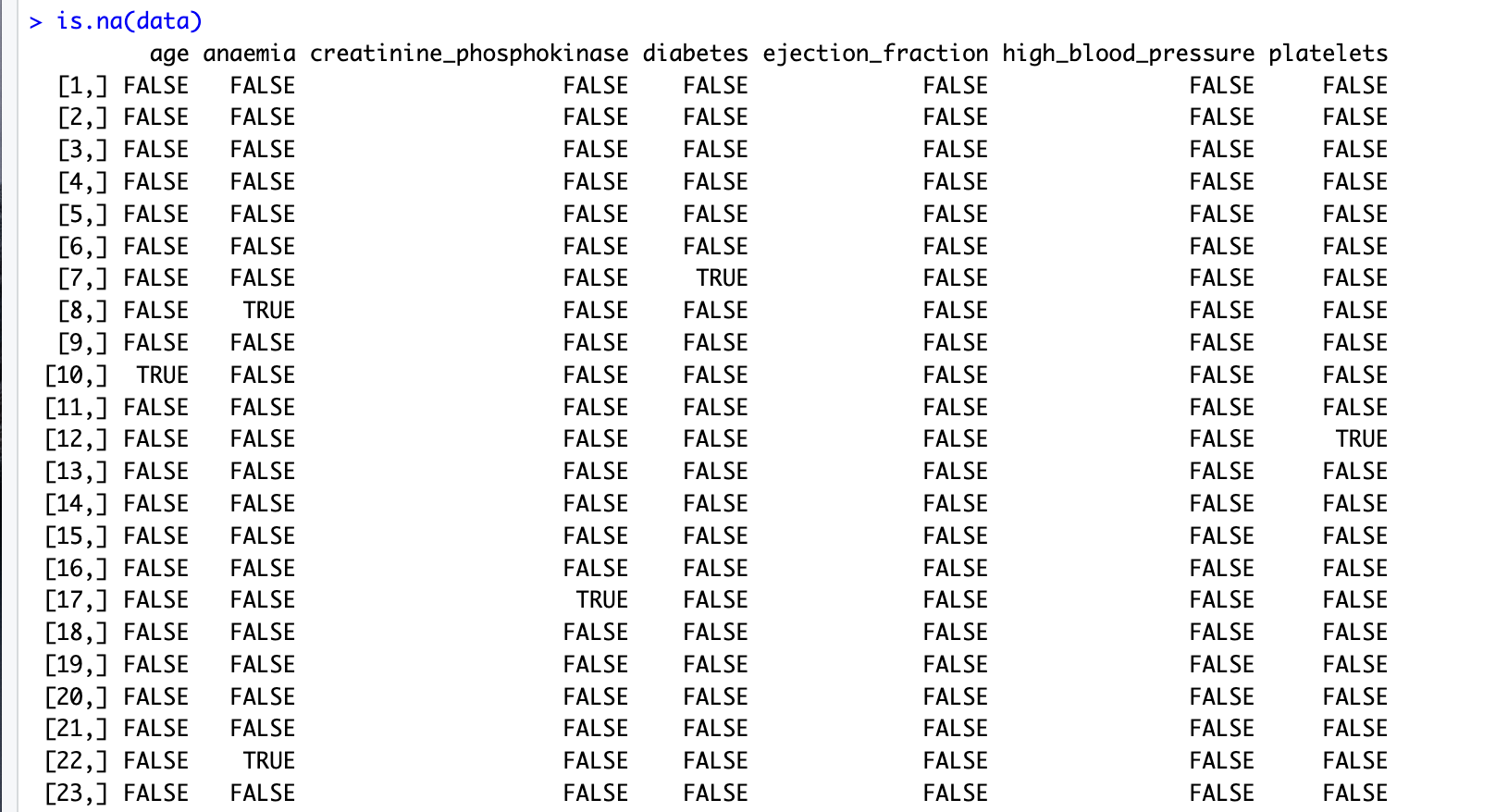
**Code:** data$creatinine\_phosphokinase[data$creatinine\_phosphokinase == "55X"] <- NA

**Description:** In the dataset "data" the attribute "creatinine\_phosphokinase" have an invalid value 55X. This code replaces any occurrences of "55X" in the "creatinine\_phosphokinase" column of the dataframe "data" with missing values (NA).

1. **Counting the number of missing/null values in each column and rows:**

**Code:** is.na(data)

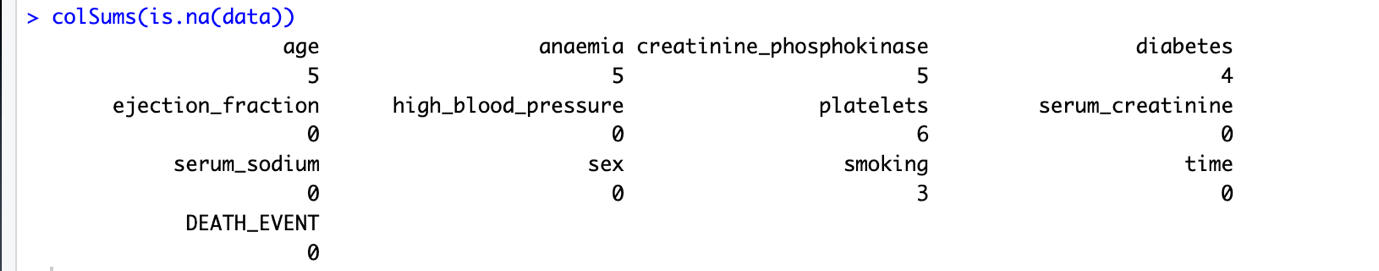
**Output:**



**Description:** is.na(data) checks for missing values in the dataframe "data" and returns a logical matrix of the same dimensions as "data", where TRUE indicates missing values and FALSE indicates non-missing values.

**Code:** colSums(is.na(data))

**Output**



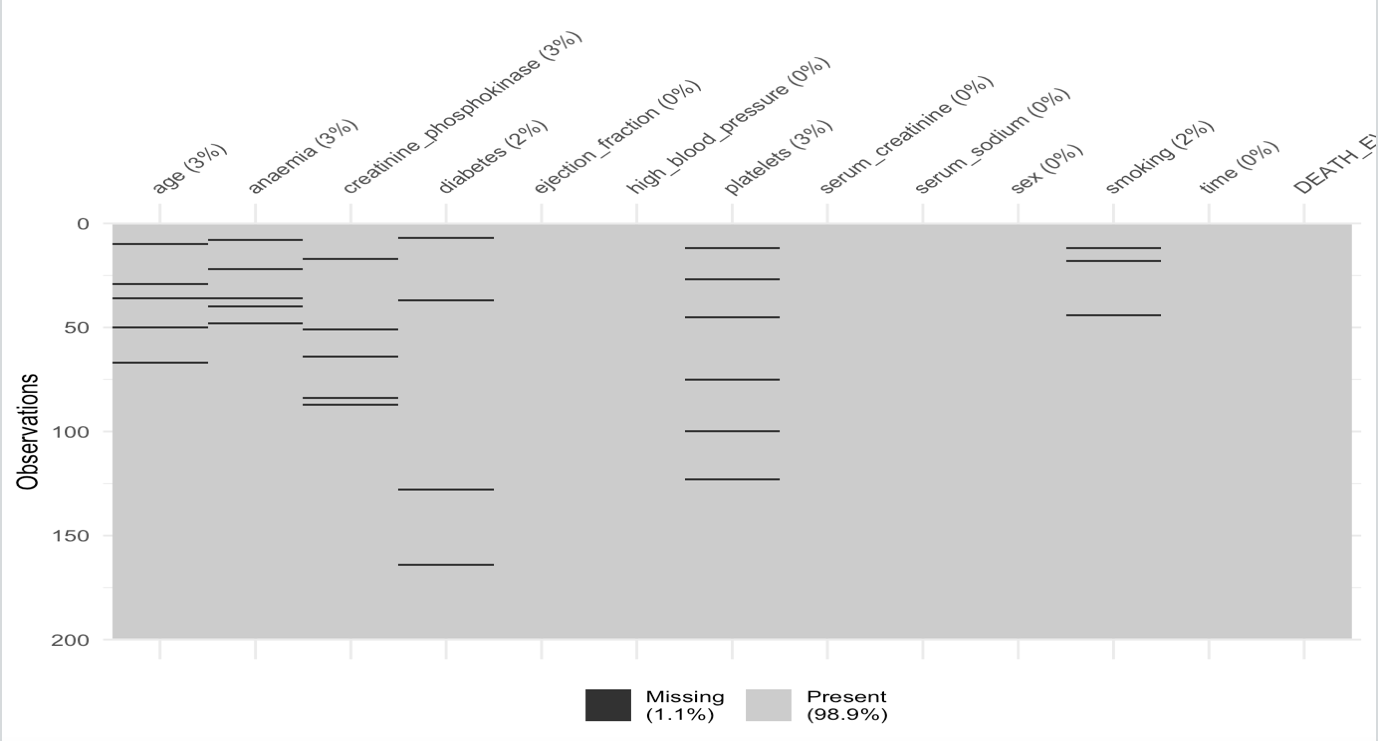
**Description:** colSums(is.na(data)) calculates the sum of missing values (NA) for each column in the dataframe "data". It returns a vector where each element represents the number of missing values in the corresponding column.

1. **Before handle missing value see missing values on a graph:**

**Code:** library(visdat)

vis\_miss(data)

**Output:**



**Description:** The vis\_miss() function from the visdat library visualizes missing values in the dataframe "data", providing a graphical representation of the presence and distribution of missing data across variables/columns.

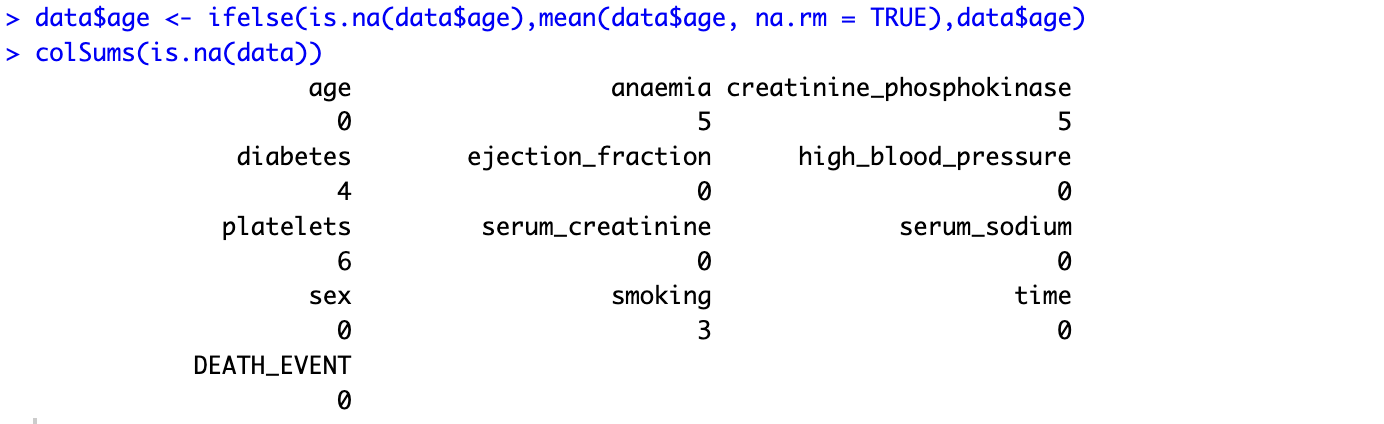
1. **Handling the dataset by discarding instances and Replace by Most Frequent/Average Value with missing/null values:**

**Using mean handle missing value of “age” attribute:**

**Code:** data$age <- ifelse(is.na(data$age),mean(data$age, na.rm = TRUE),data$age)

colSums(is.na(data))

**Output:**



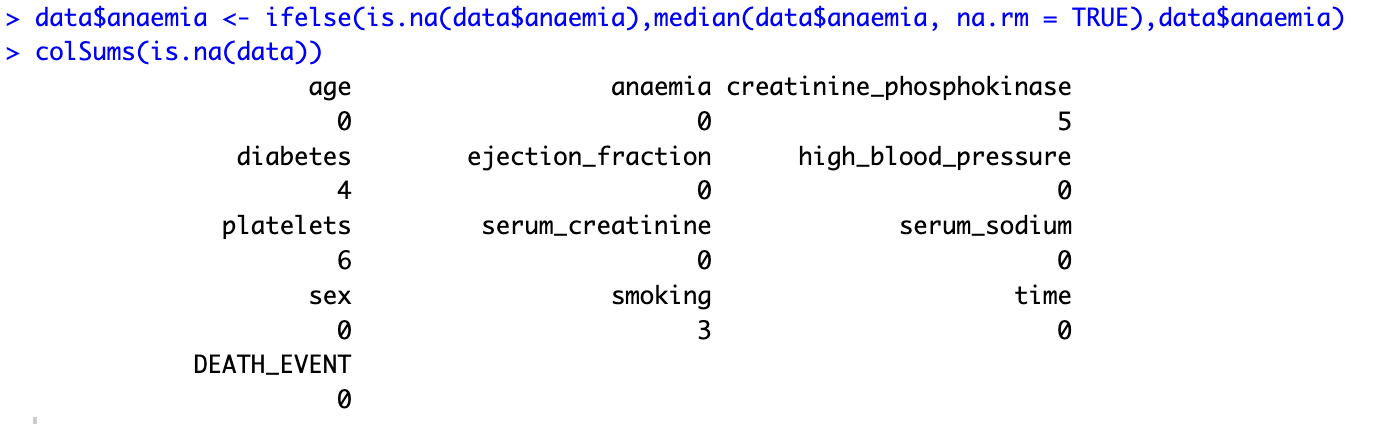
**Description:** The code replaces missing values in the "age" column of the dataframe "data" with the column's mean, and then checks for any remaining missing values in the dataframe.

**Using median handle missing value of “anaemia” attribute:**

**Code:** data$anaemia <- ifelse(is.na(data$anaemia),median(data$anaemia, na.rm = TRUE),data$anaemia)

colSums(is.na(data))

**Output:**



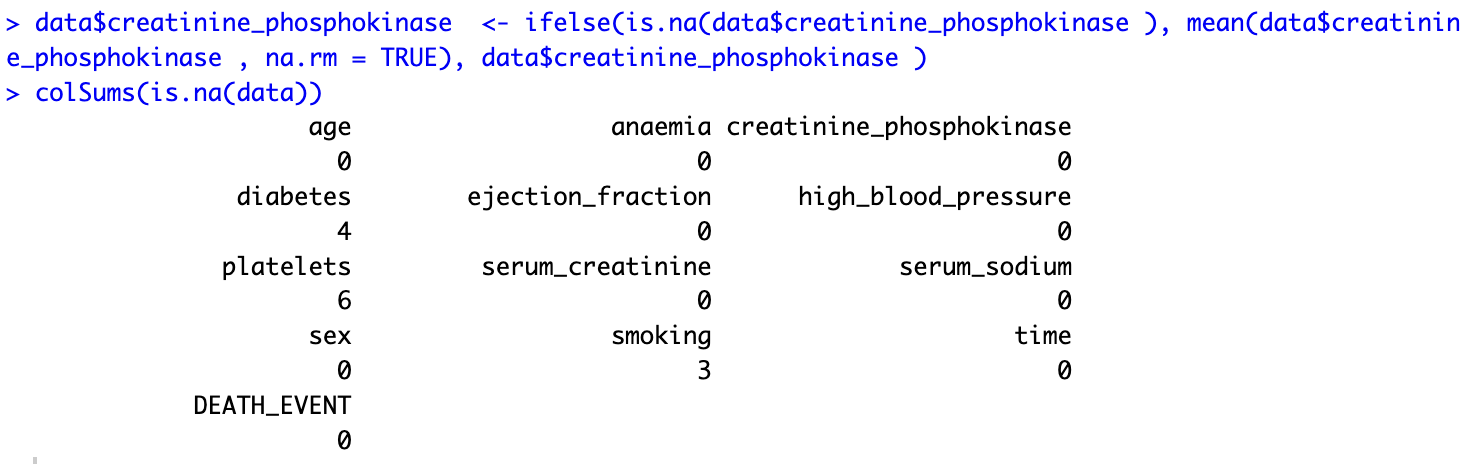
**Description:** This code replaces missing values in the "anaemia" column of the dataframe "data" with the column's median and then checks for any remaining missing values in the dataframe.

**Using mean handle missing value of “creatinine\_phosphokinase” attribute:**

**Code:** data$creatinine\_phosphokinase <- ifelse(is.na(data$creatinine\_phosphokinase ), mean(data$creatinine\_phosphokinase , na.rm = TRUE), data$creatinine\_phosphokinase )

colSums(is.na(data))

**Output:**



**Description:** This code replaces missing values in the "creatinine\_phosphokinase" column of the dataframe "data" with the column's mean and then checks for any remaining missing values in the dataframe.

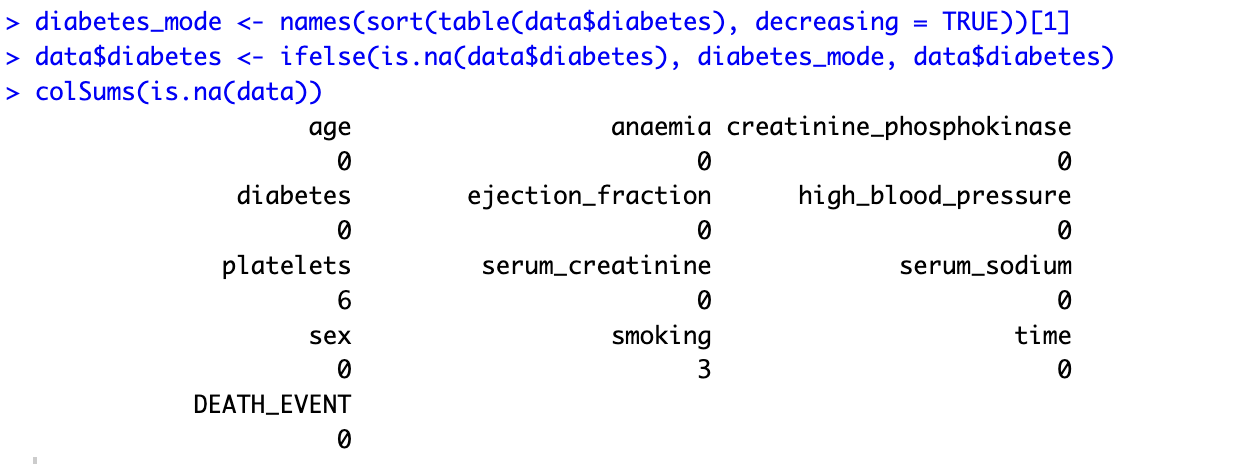
**Using mode handle missing value of “diabetes” attribute:**

**Code:** diabetes\_mode <- names(sort(table(data$diabetes), decreasing = TRUE))[1]

data$diabetes <- ifelse(is.na(data$diabetes), diabetes\_mode, data$diabetes)

colSums(is.na(data))

**Output:**



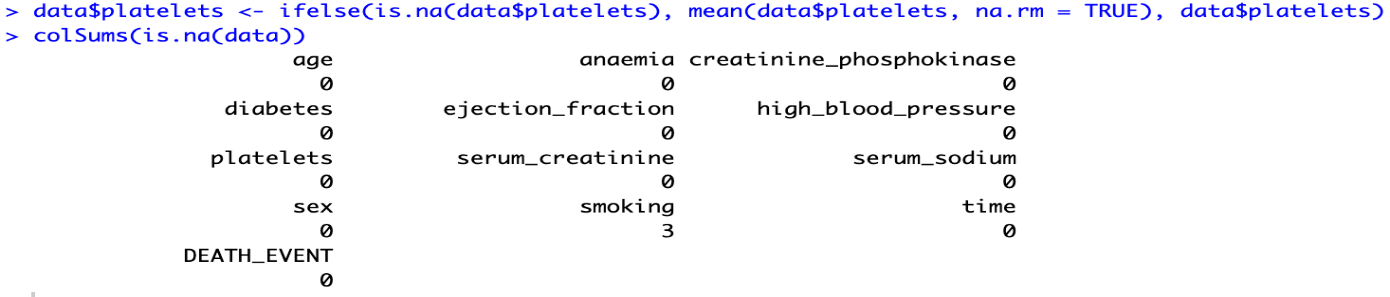
**Description:** This code finds the mode (most frequent value) of the "diabetes" column in the dataframe "data" and stores it in the variable "diabetes\_mode". Then, it replaces missing values in the "diabetes" column with the mode and checks for any remaining missing values in the dataframe.

**Using mean handle missing value of “platelets” attribute:**

**Code:** data$platelets <- ifelse(is.na(data$platelets), mean(data$platelets, na.rm = TRUE), data$platelets)

colSums(is.na(data))

**Output:**



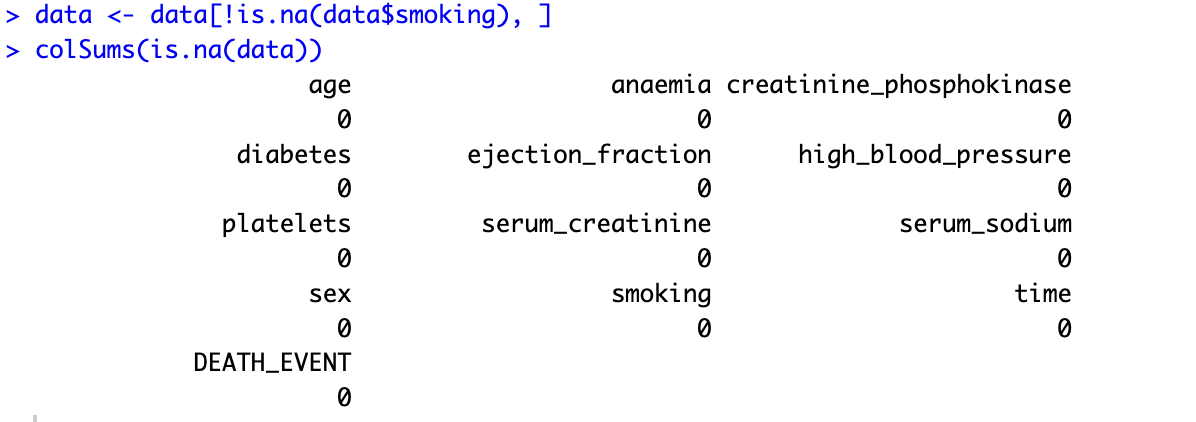
**Description:** This code replaces missing values in the "platelets" column of the dataframe "data" with the column's mean and then checks for any remaining missing values in the dataframe.

**Discarding all missing value of smoking attribute:**

**Code:** data <- data[!is.na(data$smoking), ]

colSums(is.na(data))

**Output:**



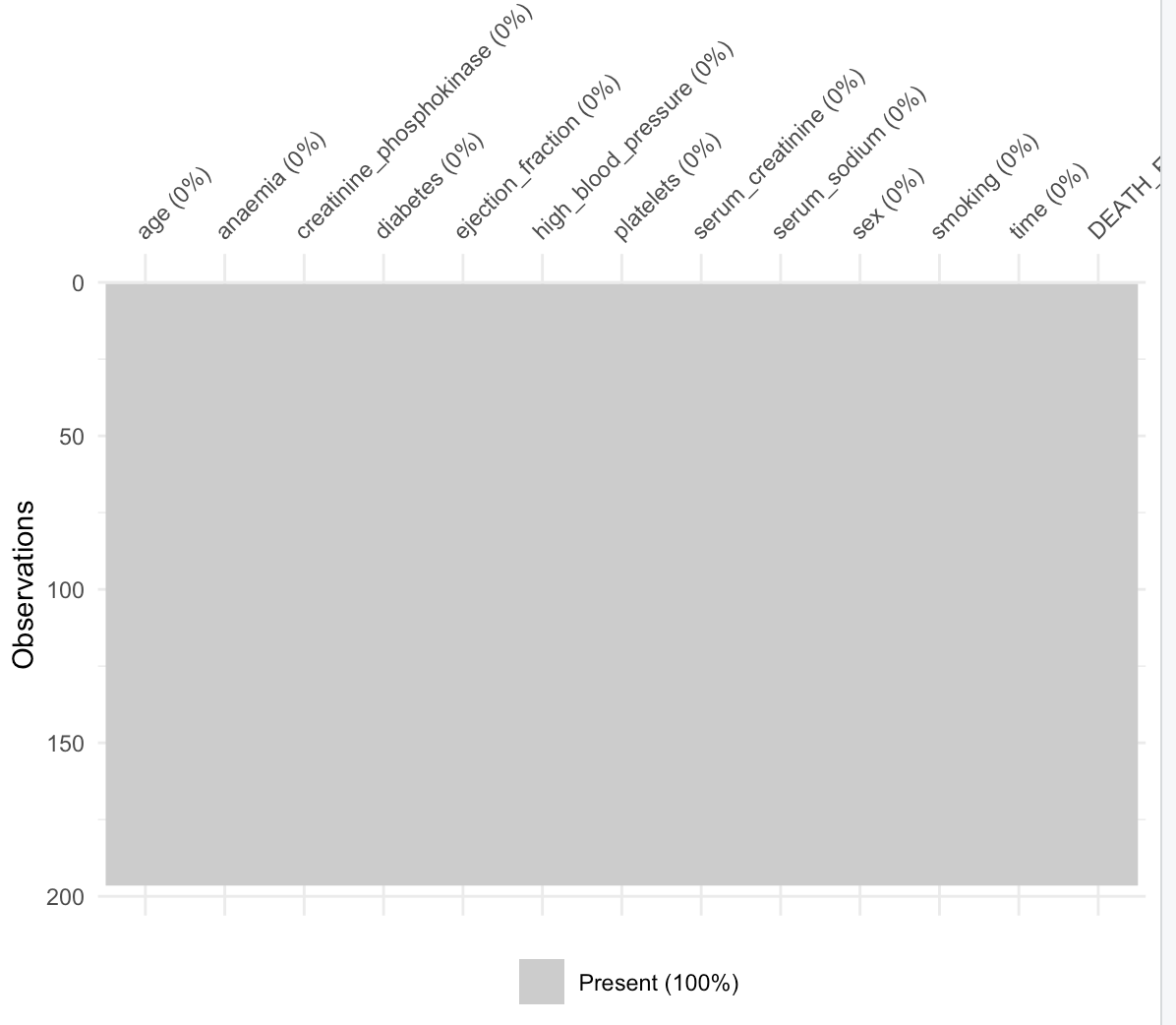
**Description:** This code removes rows from the dataframe "data" where the "smoking" column has missing values, ensuring that the dataframe contains complete information for the "smoking" variable. Then, it checks for any remaining missing values in the dataframe.

1. **After handle missing value see missing values on a graph:**

**Code:** library(visdat)

vis\_miss(data)

**Output:**



**Description:** The vis\_miss() function from the visdat library visualizes missing values in the dataframe "data", providing a graphical representation of the presence and distribution of missing data across variables/columns.

1. **howing mean, median and mode on a graph:**

**Code:** library(ggplot2)

library(dplyr)

numeric\_attributes <- data %>%

select\_if(is.numeric)

summary\_stats <- data.frame(Attribute = character(), Value = numeric(), Statistic = character(), stringsAsFactors = FALSE)

for (col in colnames(numeric\_attributes)) {

if (is.numeric(data[[col]])) {

mean\_val <- mean(data[[col]], na.rm = TRUE)

median\_val <- median(data[[col]], na.rm = TRUE)

mode\_val <- names(sort(table(data[[col]]), decreasing = TRUE))[1]

summary\_stats <- rbind(summary\_stats, data.frame(Attribute = col, Value = mean\_val, Statistic = "Mean"))

summary\_stats <- rbind(summary\_stats, data.frame(Attribute = col, Value = median\_val, Statistic = "Median"))

summary\_stats <- rbind(summary\_stats, data.frame(Attribute = col, Value = mode\_val, Statistic = "Mode"))

}}

summary\_stats$Attribute <- factor(summary\_stats$Attribute, levels = colnames(numeric\_attributes))

summary\_plot <- ggplot(summary\_stats, aes(x = Attribute, y = Value, fill = Statistic)) +

geom\_bar(stat = "identity", position = "dodge", width = 0.7) +

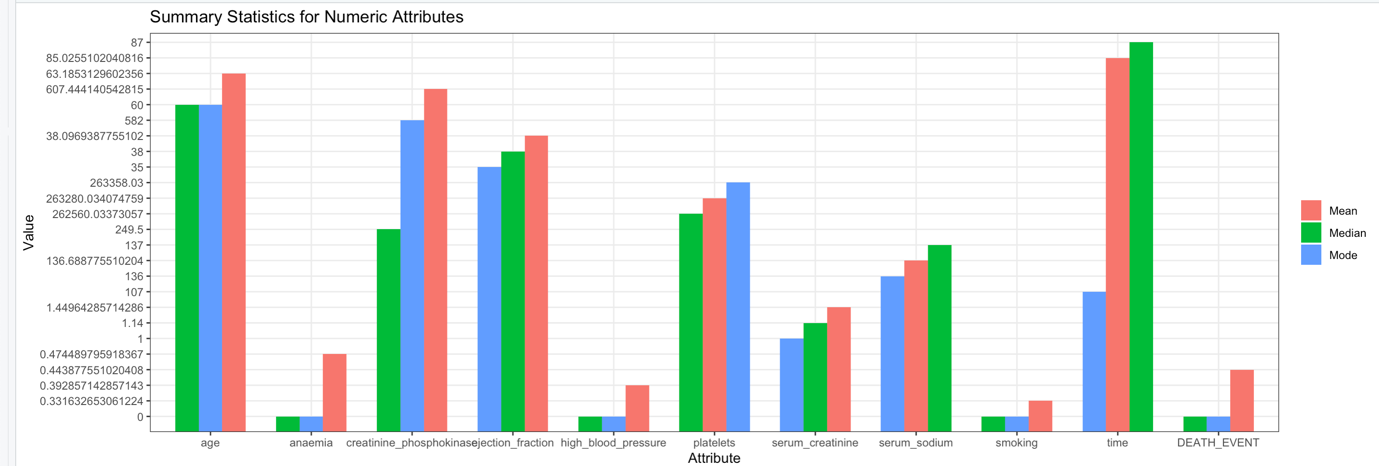
labs(title = "Summary Statistics for Numeric Attributes", x = "Attribute", y = "Value") +

theme\_bw() +

theme(legend.title = element\_blank())

print(summary\_plot)

**Output:**



**Description:** The provided code calculates summary statistics (mean, median, mode) for numeric attributes in the dataset using ggplot2 and dplyr libraries. It then creates a bar plot visualizing these statistics for each numeric attribute.

1. **Convert continuous or numeric attributes (age) into categorical attributes:**

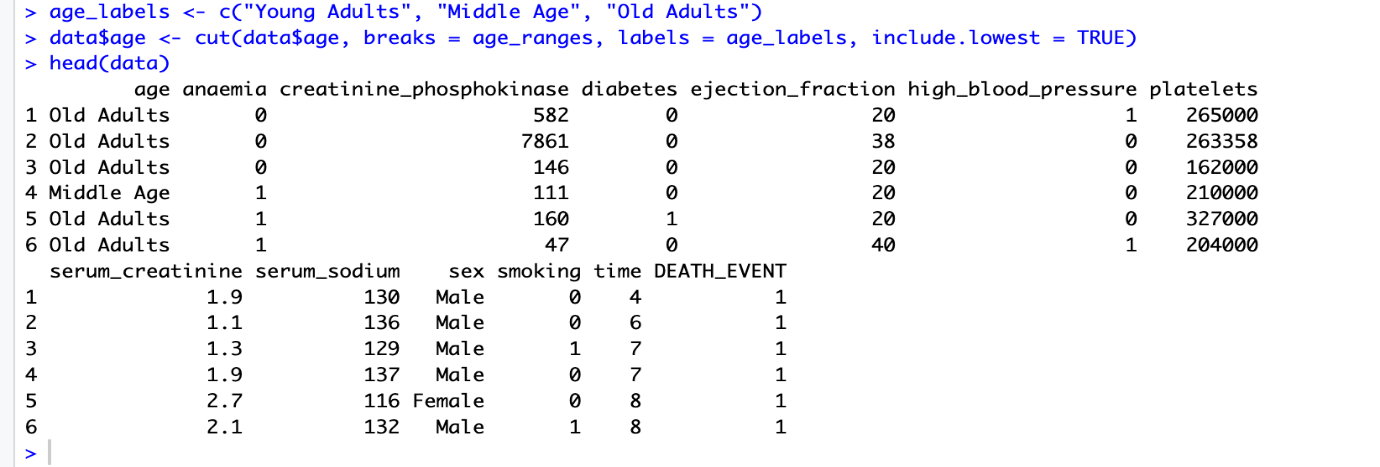
**Code:** age\_ranges <- c(0, 30, 50, Inf)

age\_labels <- c("Young Adults", "Middle Age", "Old Adults")

data$age <- cut(data$age, breaks = age\_ranges, labels = age\_labels, include.lowest = TRUE)

head(data)

**Output:**



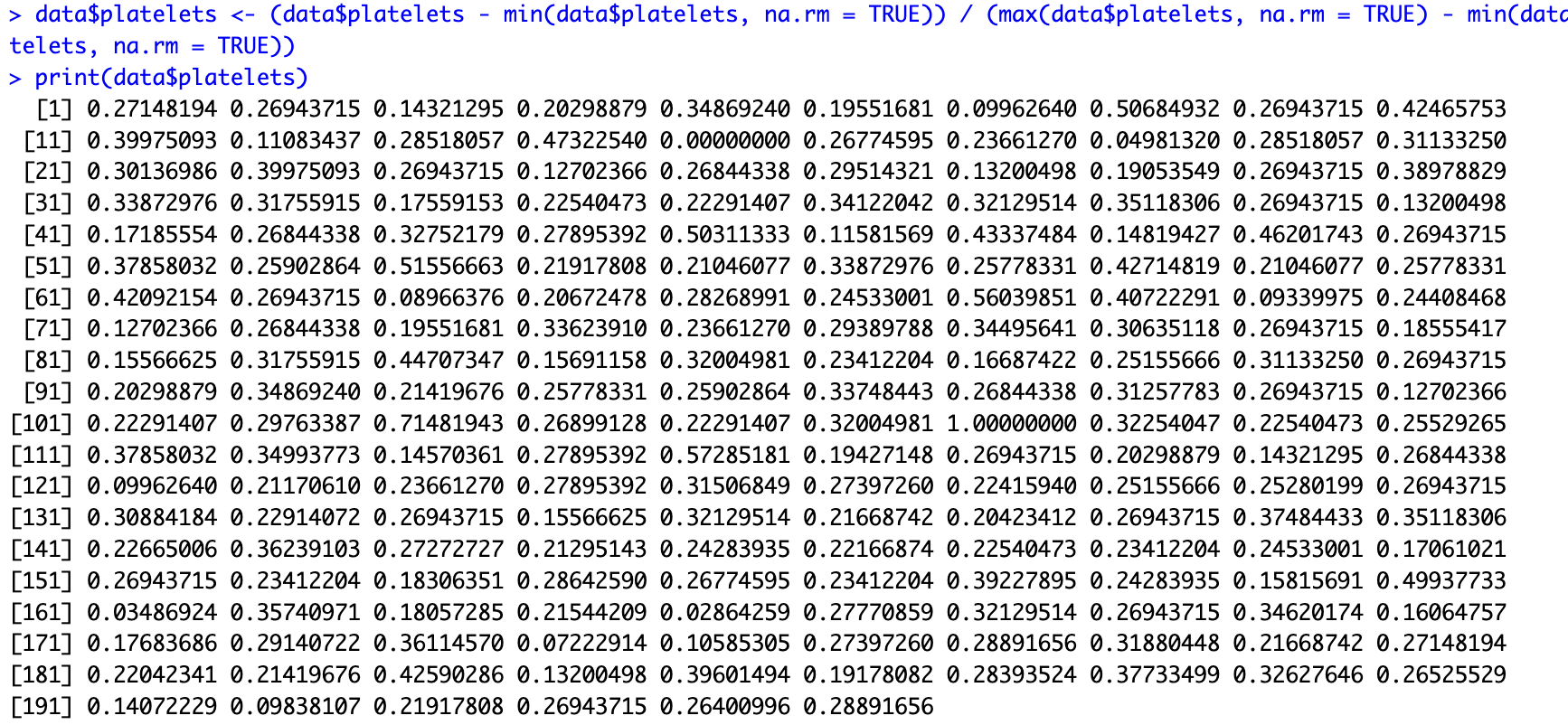
**Description:** This code converts the numerical attribute "age" in the dataset into a categorical attribute. It categorizes individuals' ages into groups: "Young Adults" (0-30), "Middle Age" (30-50), and "Old Adults" (50 and above). It then displays the first few rows of the dataset with the updated age categories.

1. **Apply the normalization method only for one attribute(platelets):**

**Code:** data$platelets <- (data$platelets - min(data$platelets, na.rm = TRUE)) / (max(data$platelets, na.rm = TRUE) - min(data$platelets, na.rm = TRUE))

print(data$platelets)

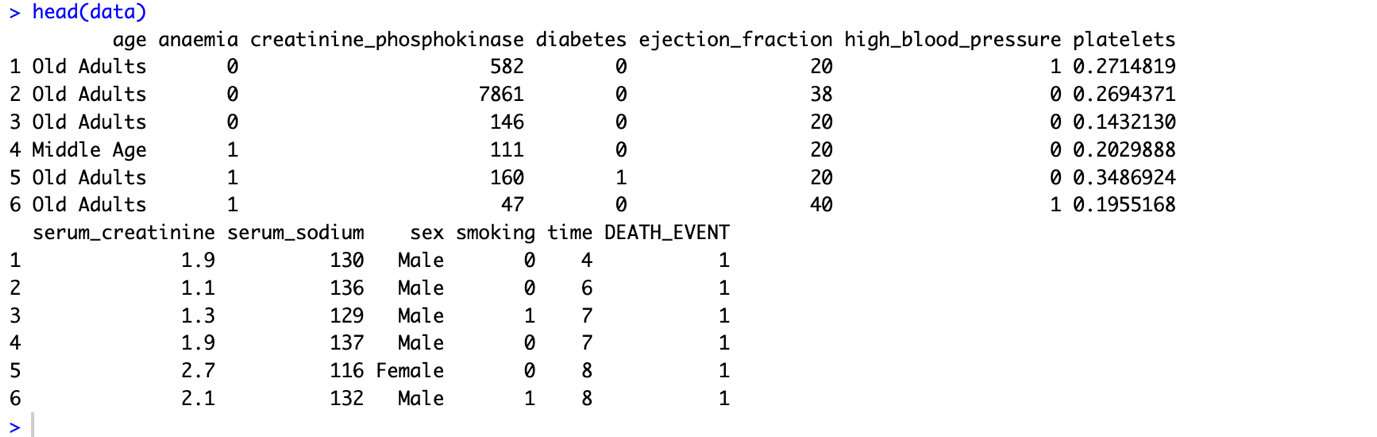
**Output:**



**Description:** This code normalizes the "platelets" column in the dataset using min-max scaling. It rescales the values to range between 0 and 1. Then, it prints the normalized values of the "platelets" column.

**Code:** head(data)

**Output:**



**Description:** The provided code normalizes the "platelets" column in the dataset using min-max scaling and then prints the first few rows of the dataframe "data" to display the changes.

1. **Show outliers in boxplot and the dataset for finding and removing outlier:**

**Boxplot before removing outliers of time attribute:**

**Code:** boxplot(data$time,

main="Boxplot before removing outliers of time",

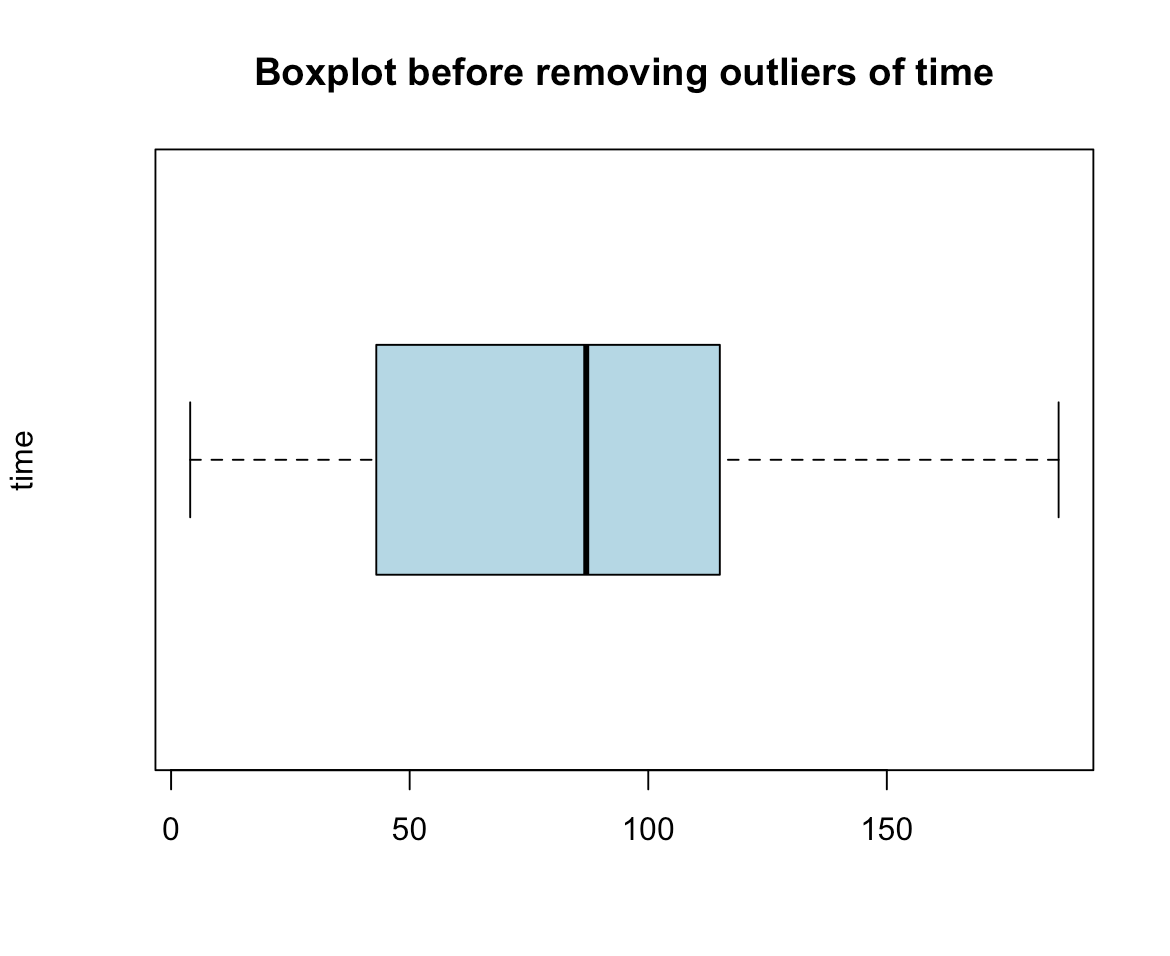
ylab="time",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**



**Description:** The code creates a boxplot of the "time" variable from the dataframe "data", displaying the distribution of values before removing outliers. We see the boxplot "time" variable have no outliers.

**Boxplot before removing outliers of serum\_creatinine attribute:**

**Code:** boxplot(data$serum\_creatinine,

main="Boxplot before removing outliers of serum\_creatinine",

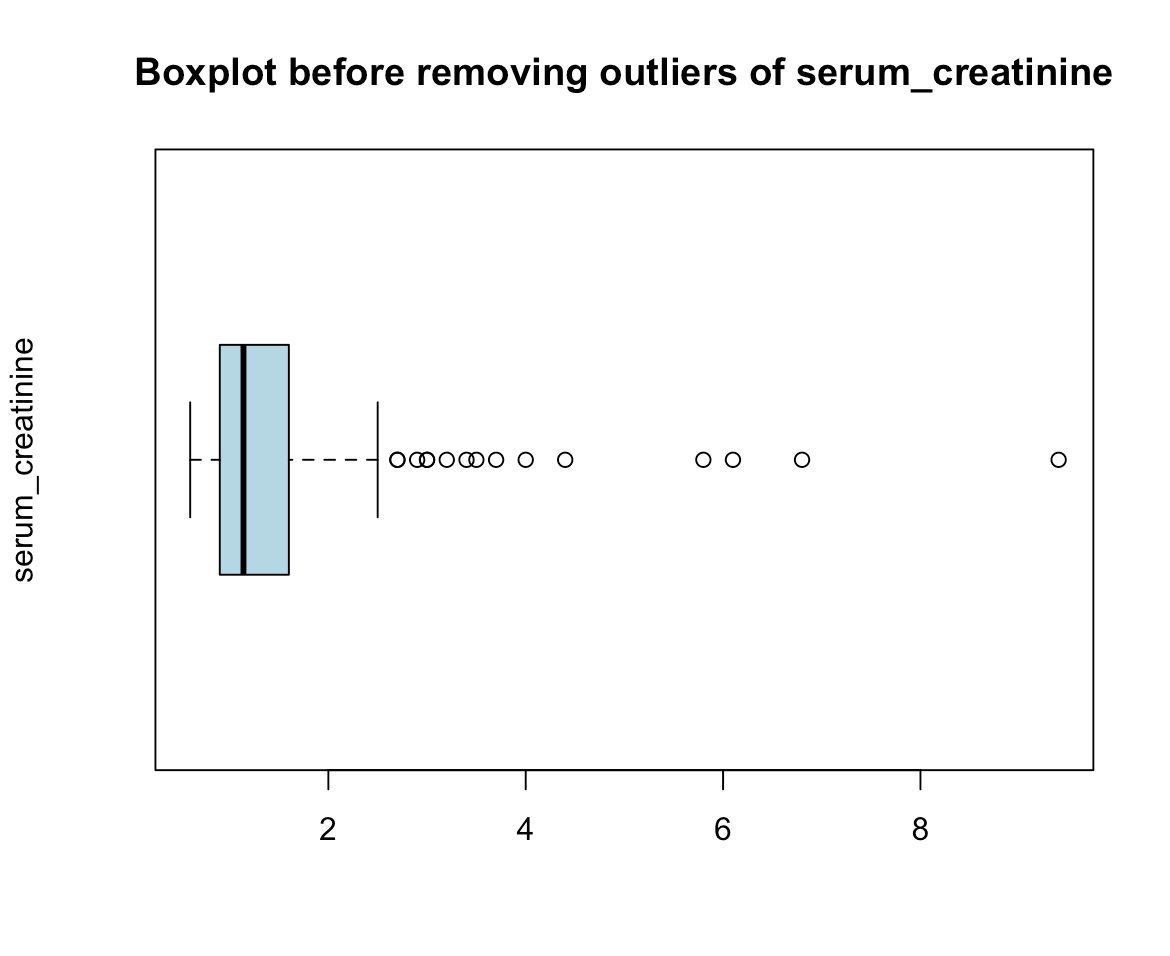
ylab="serum\_creatinine",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**

**Description:** This code generates a boxplot of the "serum\_creatinine" variable from the dataframe "data", illustrating the distribution of values before removing outliers. We see the boxplot "serum\_creatinine" variable have many outliers.

**Boxplot after removing outliers of serum\_creatinine attribute:**

**Code:** remove\_outliers <- function(x) {

q1 <- quantile(x, 0.25)

q3 <- quantile(x, 0.75)

iqr <- q3 - q1

lower\_bound <- q1 - 1.5 \* iqr

upper\_bound <- q3 + 1.5 \* iqr

x\_no\_outliers <- x[x >= lower\_bound & x <= upper\_bound]

return(x\_no\_outliers)}

data\_clean <- data[!data$serum\_creatinine %in% remove\_outliers(data$serum\_creatinine), ]

boxplot(data\_clean$serum\_creatinine,

main="Boxplot after removing outliers of serum\_creatinine",

ylab="serum\_creatinine",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**

A diagram of a box with a blue rectangle

Description automatically generated**Description:** This code defines a function “remove\_outliers” that removes outliers from a given numeric vector “x” using the Interquartile Range (IQR) method. It then applies this function to the "serum\_creatinine" column in the dataframe "data" to create a new dataframe "data\_clean" without outliers. Finally, it generates a boxplot of the "serum\_creatinine" variable from the cleaned dataframe, illustrating the distribution of values after removing outliers.

**Boxplot before removing outliers of creatinine\_phosphokinase attribut:**

**Code:** boxplot(data$creatinine\_phosphokinase,

main="Boxplot before removing outliers of creatinine\_phosphokinase",

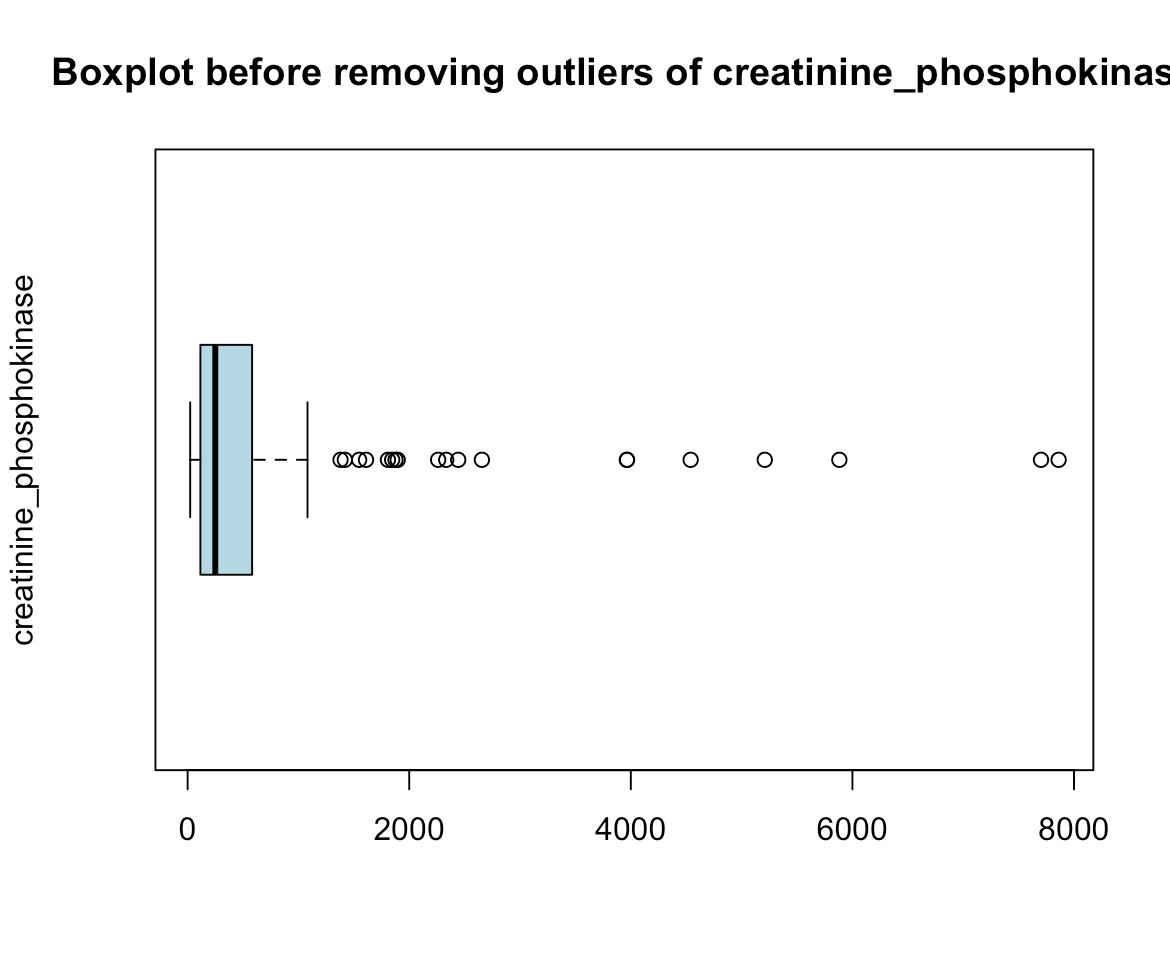
ylab="creatinine\_phosphokinase",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**

**Description:** This code creates a boxplot of the "creatinine\_phosphokinase" variable from the dataframe "data", illustrating the distribution of values before removing outliers. We see the boxplot "creatinine\_phosphokinase" variable have many outliers.

**Boxplot after removing outliers of creatinine\_phosphokinase attribute:**

**Code:** data\_clean\_creatinine <- data[!data$creatinine\_phosphokinase %in% remove\_outliers(data$creatinine\_phosphokinase), ]

boxplot(data\_clean\_creatinine$creatinine\_phosphokinase,

main="Boxplot after removing outliers of creatinine\_phosphokinase",

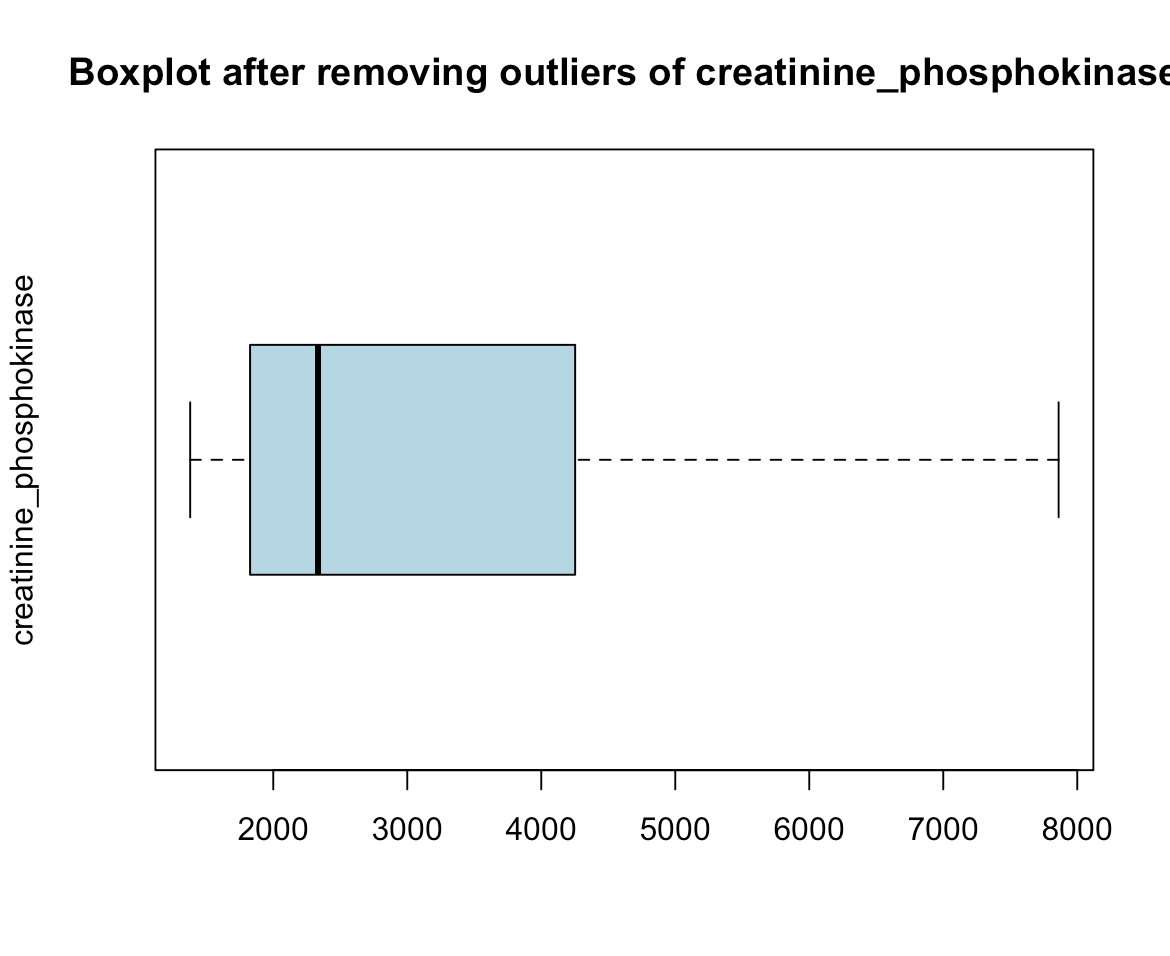
ylab="creatinine\_phosphokinase",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**



**Description:** This code creates a new dataframe "data\_clean\_creatinine" by removing outliers from the "creatinine\_phosphokinase" column in the original dataframe "data". Then, it generates a boxplot of the "creatinine\_phosphokinase" variable from the cleaned dataframe, illustrating the distribution of values after removing outliers.

**Boxplot before removing outliers of serum\_sodium attribute:**

**Code:** boxplot(data$serum\_sodium,

main="Boxplot before removing outliers of serum\_sodium",

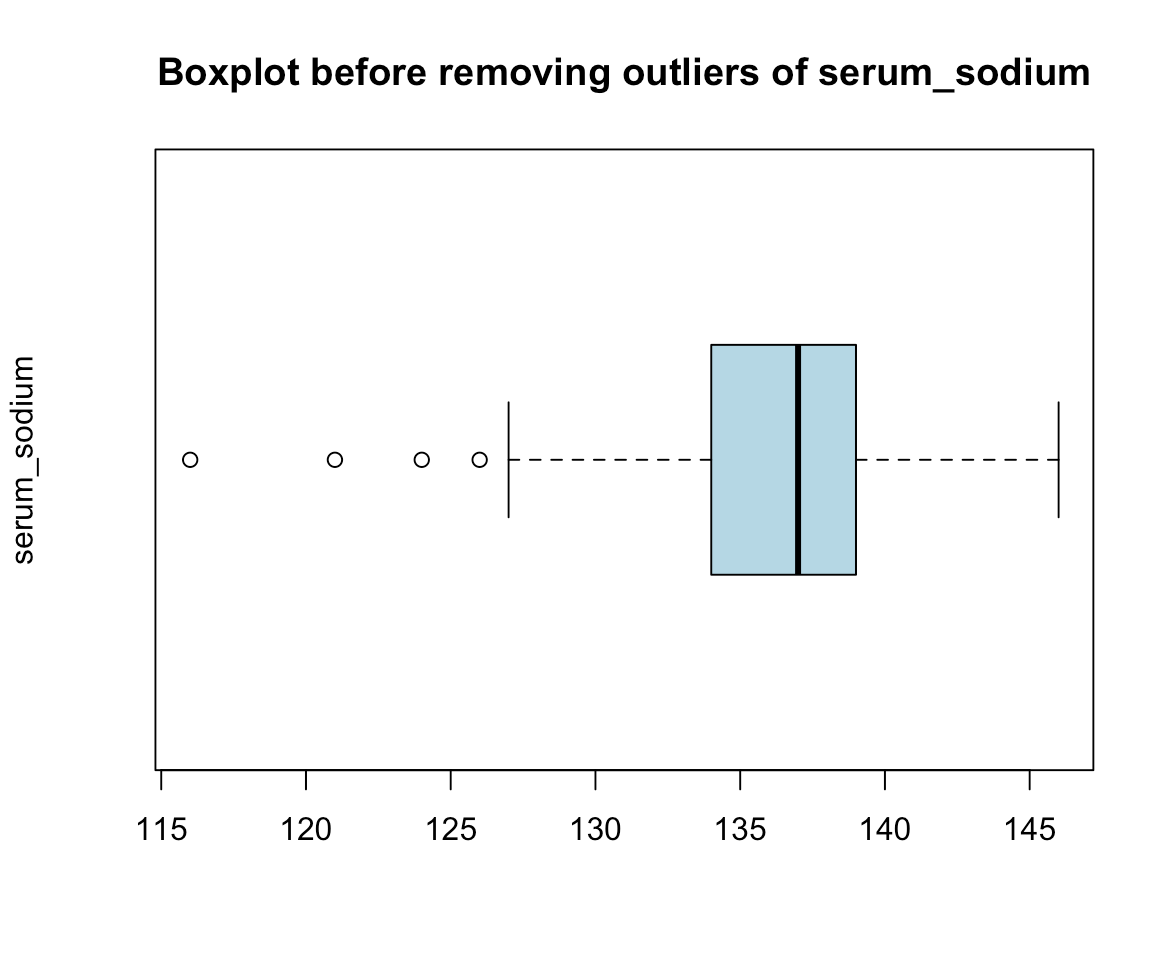
ylab="serum\_sodium",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**

**Description:** This code generates a boxplot of the "serum\_sodium" variable from the dataframe "data", displaying the distribution of values before removing outliers.

**Boxplot after removing outliers of serum\_sodium attribute:**

**Code:** data\_clean\_sodium <- data[!data$serum\_sodium %in% remove\_outliers(data$serum\_sodium), ]

boxplot(data\_clean\_sodium$serum\_sodium,

main="Boxplot after removing outliers of serum\_sodium",

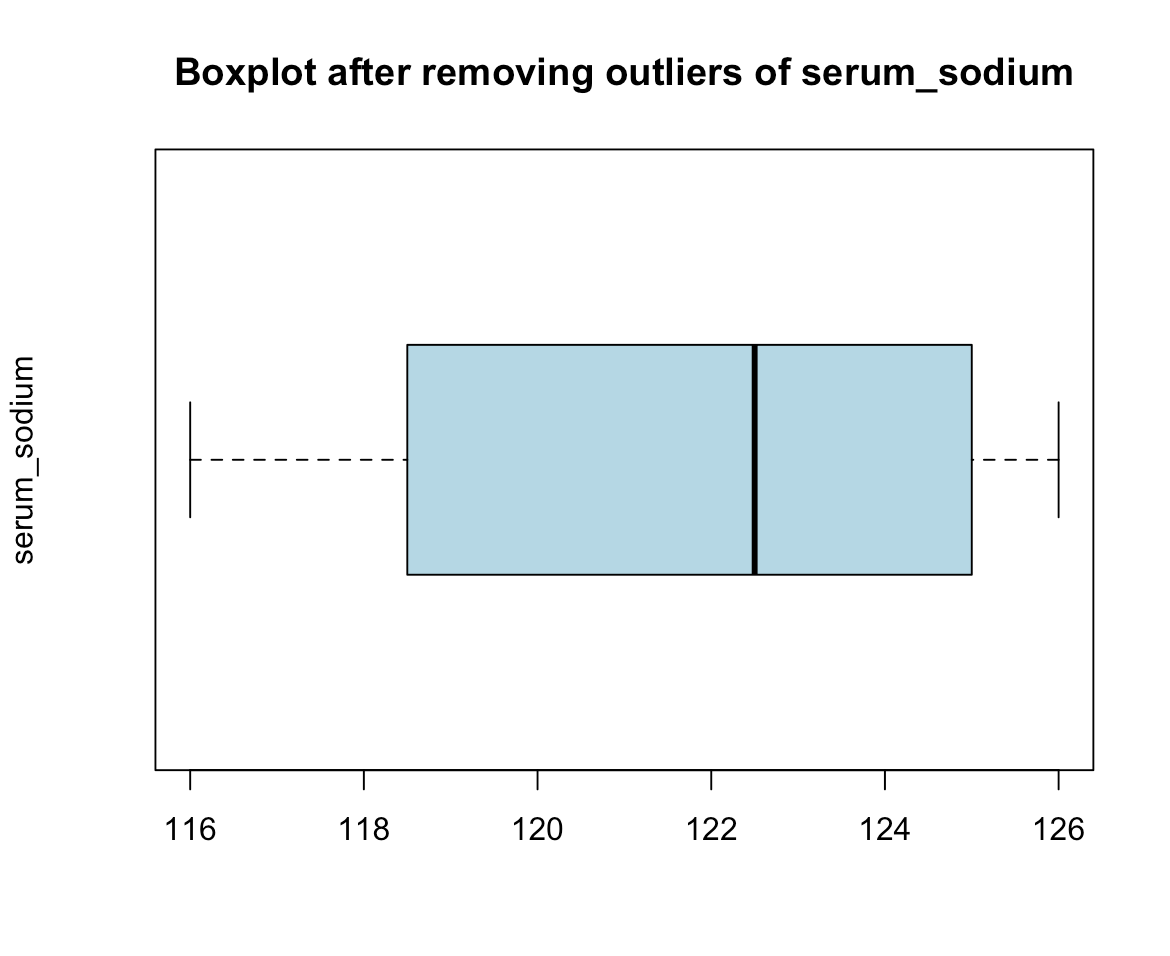
ylab="serum\_sodium",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**



**Description:** This code creates a new dataframe "data\_clean\_sodium" by removing outliers from the "serum\_sodium" column in the original dataframe "data". Then, it generates a boxplot of the "serum\_sodium" variable from the cleaned dataframe, illustrating the distribution of values after removing outliers.

**Boxplot before removing outliers of ejection\_fraction attribute:**

**Code:** boxplot(data$ejection\_fraction,

main="Boxplot before removing outliers of ejection\_fraction",

ylab="ejection\_fraction",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**

A graph with a blue box

Description automatically generated

**Description:** This code generates a boxplot of the "ejection\_fraction" variable from the dataframe "data", illustrating the distribution of values before removing outliers.

**Boxplot after removing outliers of ejection\_fraction attribute:**

**Code:** data\_clean\_ejection <- data[!data$ejection\_fraction %in% remove\_outliers(data$ejection\_fraction), ]

boxplot(data\_clean\_ejection$ejection\_fraction,

main="Boxplot after removing outliers of ejection\_fraction",

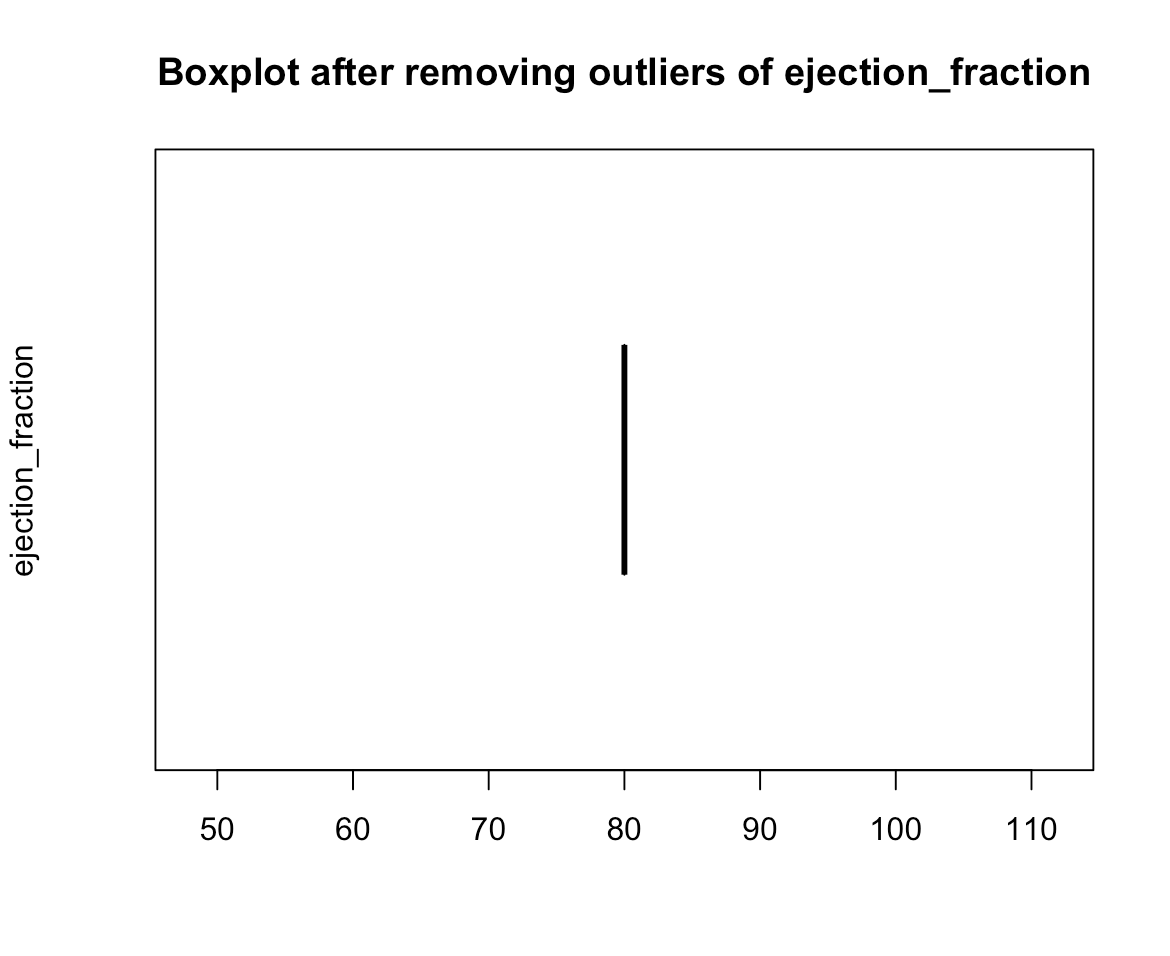
ylab="ejection\_fraction",

col="lightblue",

border="black",

horizontal=TRUE)

**Output:**



**Description:** This code creates a new dataframe "data\_clean\_ejection" by removing outliers from the "ejection\_fraction" column in the original dataframe "data". Then, it generates a boxplot of the "ejection\_fraction" variable from the cleaned dataframe, illustrating the distribution of values after removing outliers.