**Report Writing**

**Data preparation:**

1. **Installing library packages and loading them:**

**Code:**

install.packages("ggplot2")

install.packages("dplyr")

install.packages("visdat")

install.packages("ROSE")

library(ggplot2)

library(dplyr)

library(visdat)

library(ROSE)

**Output:**

A screenshot of a computer

Description automatically generated

**Description:**

At first, all the necessary library packages were downloaded and loaded into R studio. libraries: ggplot2 for data visualization, dplyr for data manipulation, visdat for visualizing data structures, and ROSE for addressing imbalanced data.

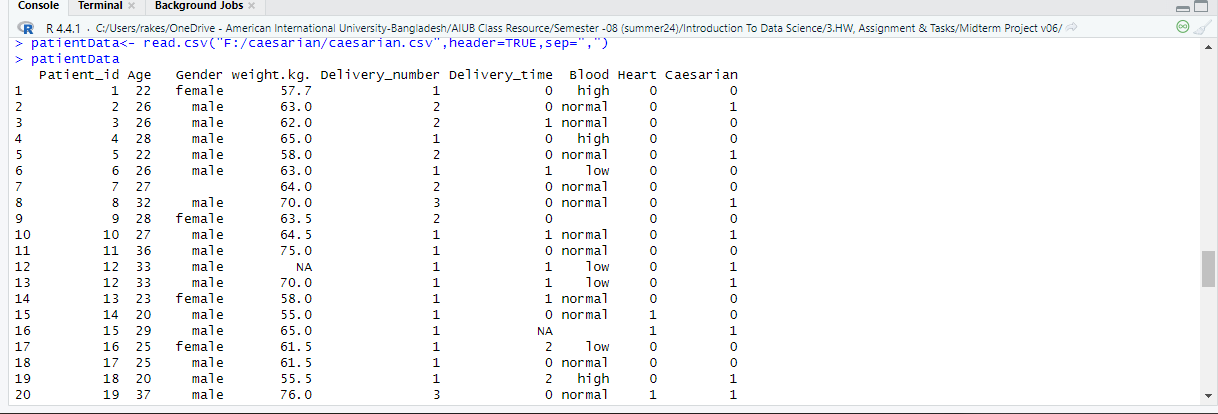
1. **Reading/Loading the dataset:**

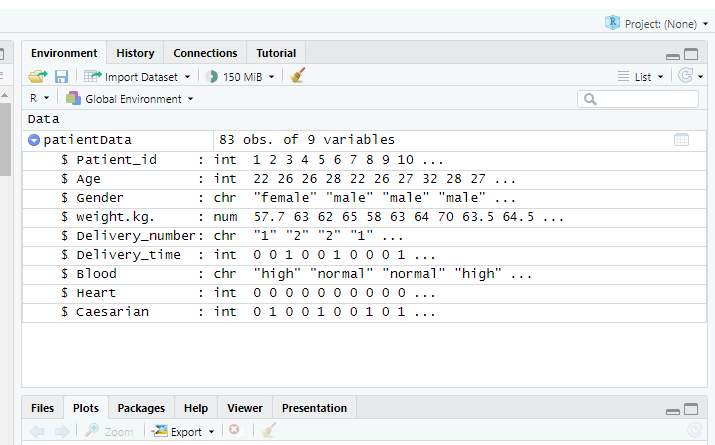
**Code:**

patientData<- read.csv("E:/caesarian/caesarian.csv",header=TRUE,sep=",")

patientData

**Output:**

****

****

**Description:**

After loading the library, the dataset stored in a CSV file was imported into R studio using read.csv() method. The dataset is stored in a dataframe called “patientData”. Using the str() method, it was checked whether the dataset was successfully loaded or not.

1. **Inspecting the structure and summery of the data:**

**Code:**

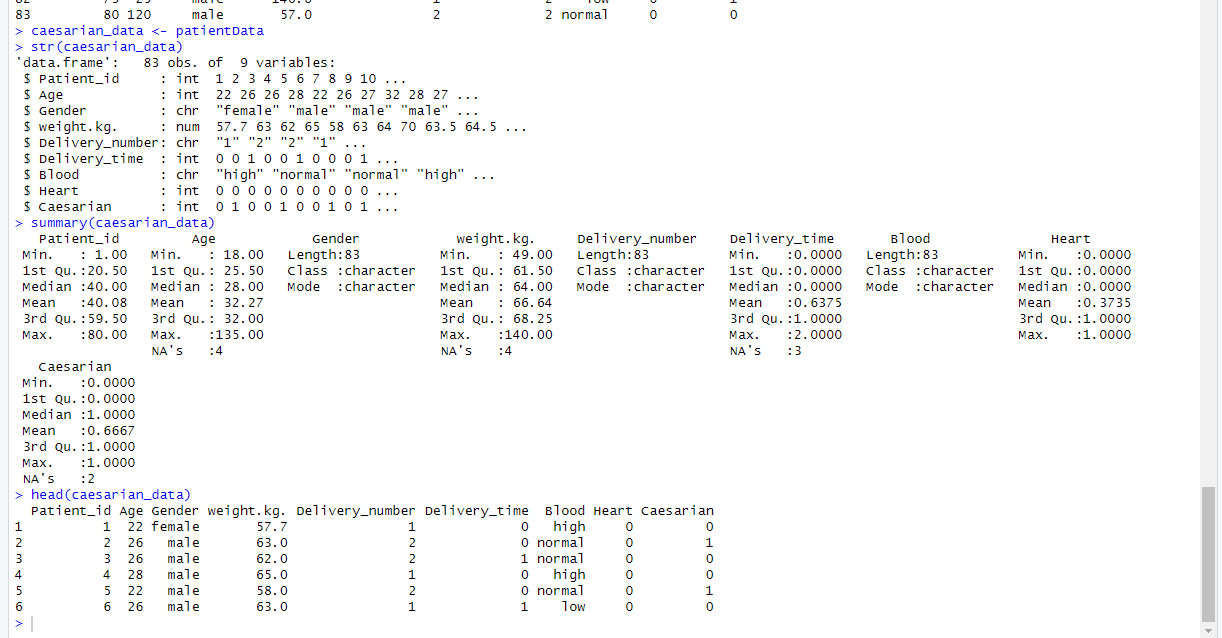
caesarian\_data <- patientData

str(caesarian\_data)

summary(caesarian\_data)

head(caesarian\_data)

**Output:**



**Description:**

First, a copy of patientData was made in caesarian\_data dataframe. The stl() displayed the data types of each column, the summary() slowed the min,max,mean,median,1st Quadrant, 3rd Quadrant, no of NA values, the head() printed the first 6 rows of the dataset.

1. **Checking unique values for the catagorical entities to find invalid entries:**

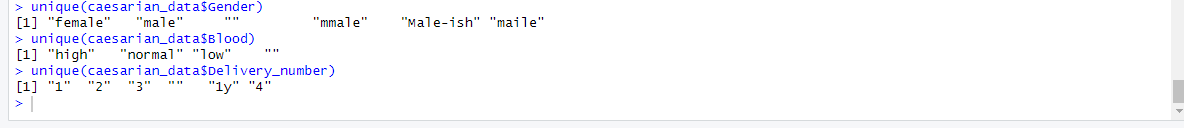
**Code:**

unique(caesarian\_data$Gender)

unique(caesarian\_data$Blood)

unique(caesarian\_data$Delivery\_number)

**Output:**

****

**Description:**

Checking for unique values to identify potential invalid entries in categorical type columns Gender, Blood, and Delivery\_number.

1. **Defining & replacing invalid values:**

**Code:**

valid\_genders <- c("male", "female")

valid\_delivery\_number <- c("1", "2", "3", "4")

replace\_invalid <- function(value, valid\_values, replacement = NA) {

if (value %in% valid\_values) {

return(value)

} else {

return(replacement)

}

}

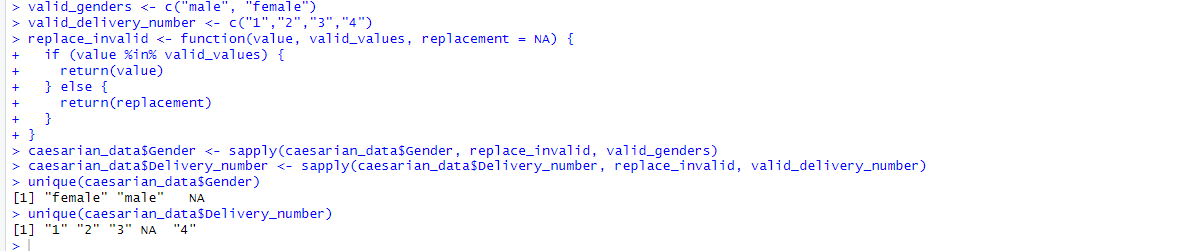
caesarian\_data$Gender<-sapply(caesarian\_data$Gender,replace\_invalid,valid\_genders)

caesarian\_data$Delivery\_number<-sapply(caesarian\_data$Delivery\_number, replace\_invalid, valid\_delivery\_number)

unique(caesarian\_data$Gender)

unique(caesarian\_data$Delivery\_number)

**Output:**

****

**Description:**

Now, valid values of gender and delivery\_number column was defined in vertor data type. A function named replace\_invalid was made. The function takes value, valid\_value and replacement = NA as its parameter and checks weather the value is in valid\_value or not. For valid, the value remains the same and if found invalid, it is replaced with the replacement value (NA). The function is applied to replace invalid values using sapply(). The sapply() function takes a list, vector, or data frame as input and gives output in the form of an array or matrix object. Optionally, the data was inspected to see weather the replacement was successful or not.

1. **Type conversion of data to numerical type:**

**Code:**

caesarian\_data$Delivery\_number <- as.integer(caesarian\_data$Delivery\_number)

caesarian\_data$Gender <- factor(caesarian\_data$Gender,

levels = c("male", "female"),

labels = c(0, 1))

caesarian\_data$Blood <- factor(caesarian\_data$Blood,

levels = c("low", "normal", "high"),

labels = c(0, 1, 2))

str(caesarian\_data)

**Output**:

A screenshot of a computer

Description automatically generated

**Description**: Then, by modifying the factor data structures, Delivery\_number and Blood column was converted to factor type to make the numerical. Factors are categorical variables that can only take a predefined set of values. In the factor() function, the level specifies the valid levels or categories for the factor variable, where the labels assigns numerical codes to each level. By using the <- as.integer() function, delivery\_number column was converted to a numerical type. Optionally, the structure of the dataset was checked using str() to ensure successful conversion.

1. **Calculating & displaying Q1, Q3, and IQR:**

**Code:**

Q1\_age <- quantile(caesarian\_data$Age, 0.25, na.rm = TRUE)

Q3\_age <- quantile(caesarian\_data$Age, 0.75, na.rm = TRUE)

IQR\_age <- Q3\_age - Q1\_age

Q1\_weight <- quantile(caesarian\_data$weight.kg., 0.25, na.rm = TRUE)

Q3\_weight <- quantile(caesarian\_data$weight.kg., 0.75, na.rm = TRUE)

IQR\_weight <- Q3\_weight - Q1\_weight

Q1\_age

Q3\_age

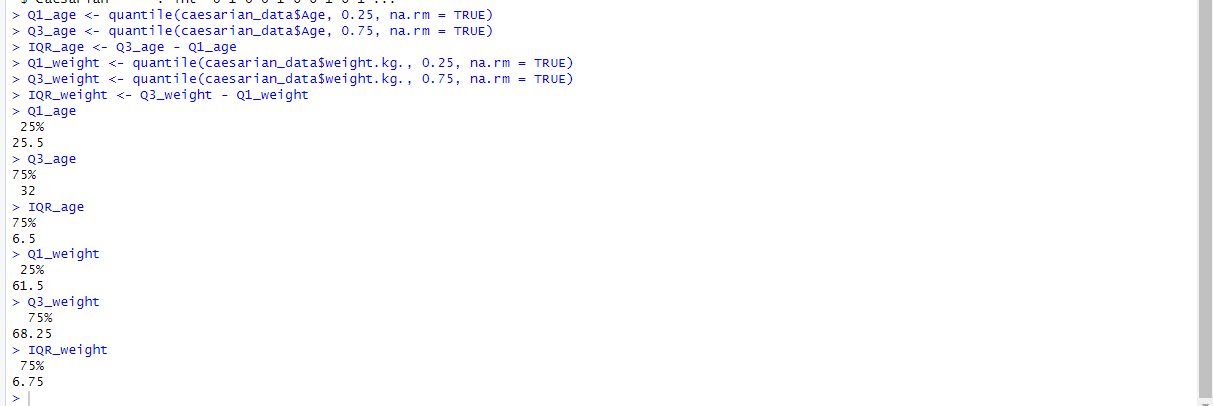
IQR\_age

Q1\_weight

Q3\_weight

IQR\_weight

**Output:**

****

**Description:**

Q1 (1st Quartile): The 25th percentile value, indicating that 25% of the data points are below this value.

Q3 (3rd Quartile): The 75th percentile value, indicating that 75% of the data points are below this value.

IQR (Interquartile Range): The difference between Q3 and Q1, representing the spread of the middle 50% of the data.

Here, quantile() function was used to calculate Q1 and Q3. Then using them IQR was measured and them all three are displayed.

1. **Identifying outliers and plotting boxplot:**

**Code:**

detect\_outliers <- function(x, Q1, Q3, IQR) {

lower\_bound <- Q1 - 1.5 \* IQR

upper\_bound <- Q3 + 1.5 \* IQR

return(x < lower\_bound | x > upper\_bound)

}

age\_outliers <- detect\_outliers(caesarian\_data$Age, Q1\_age, Q3\_age, IQR\_age)

weight\_outliers <- detect\_outliers(caesarian\_data$weight.kg., Q1\_weight, Q3\_weight, IQR\_weight)

ggplot(caesarian\_data, aes(y = "", x = Age)) +

geom\_boxplot(fill = "lightblue") +

ggtitle("Boxplot of Age with Outliers") +

theme\_minimal() +

labs(x = "Age", y = "")

ggplot(caesarian\_data, aes(y = "", x = weight.kg.)) +

geom\_boxplot(fill = "lightgreen") +

ggtitle("Boxplot of Weight (kg) with Outliers") +

theme\_minimal() +

labs(x = "Weight (kg)", y = "")

**Output:**

**A screenshot of a graph

Description automatically generated**

**A graph with green bars

Description automatically generated**

**Description:**

Here, detect\_outliers function identifies outliers in a numeric vector called x using the interquartile range (IQR) method. Using the function then outliers for age, weight column was identified and then using ggplot function the boxplot was formed. ggplot() function has different types of features to plot and visualize data like geom\_boxplot Adds a boxplot layer to the plot, ggtitle sets the plot’s title, theme\_minimal() removes background grid lines and clutter, labs() labels x and y axis.

1. **Handling outliers:**

**Code:**

cap\_outliers <- function(x, Q1, Q3, IQR) {

lower\_bound <- Q1 - 1.5 \* IQR

upper\_bound <- Q3 + 1.5 \* IQR

x[x < lower\_bound] <- lower\_bound

x[x > upper\_bound] <- upper\_bound

return(x)

}

caesarian\_data$Age <- cap\_outliers(caesarian\_data$Age, Q1\_age, Q3\_age, IQR\_age)

caesarian\_data$weight.kg. <- cap\_outliers(caesarian\_data$weight.kg., Q1\_weight, Q3\_weight, IQR\_weight)

ggplot(caesarian\_data, aes(y = "", x = Age)) +

geom\_boxplot(fill = "lightblue") +

ggtitle("Boxplot of Age After Handling Outliers") +

theme\_minimal() +

labs(x = "Age", y = "")

ggplot(caesarian\_data, aes(y = "", x = weight.kg.)) +

geom\_boxplot(fill = "lightgreen") +

ggtitle("Boxplot of Weight (kg) After Handling Outliers") +

theme\_minimal() +

labs(x = "Weight (kg)", y = "")

**Output:**

**A graph with a rectangle

Description automatically generated**

**A green rectangle with black lines

Description automatically generated**

**Description:**

A function called cap\_outliers was formed to handle outliers and using it outliers were handled for age and weight column. Then the boxplot was generated again to check whether outlier handling was successful or not.

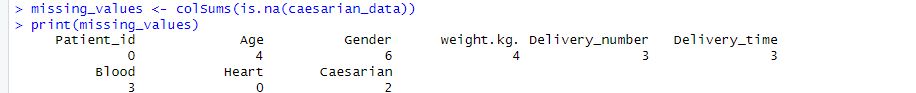
1. **Checking for missing values:**

**Code:**

missing\_values <- colSums(is.na(caesarian\_data))

print(missing\_values)

**Output:**

****

**Description:**

By using is.na() function the presence of missing values were detected and using colSums() function the sum was calculated and displayed.

1. **Visualizing missing values:**

**Code:**

vis\_miss(missing\_values)

**Output:**

**A grey rectangular object with black lines

Description automatically generated**

**Description:**

By using vis\_miss() function the Visualization of missing values was achieved.

1. **Removing missing values:**

**Code:**

caesarian\_data$Age[is.na(caesarian\_data$Age)] <- round(median(caesarian\_data$Age, na.rm = TRUE))

caesarian\_data$weight.kg.[is.na(caesarian\_data$weight.kg.)]<-mean(caesarian\_data$weight.kg., na.rm = TRUE)

mode\_impute <- function(x) {

uniq\_x <- na.omit(unique(x))

freq\_x <- table(x)

return(names(freq\_x)[which.max(freq\_x)])

}

caesarian\_data$Gender[is.na(caesarian\_data$Gender)]<-mode\_impute(caesarian\_data$Gender)

caesarian\_data$Delivery\_number[is.na(caesarian\_data$Delivery\_number)]<- mode\_impute(caesarian\_data$Delivery\_number)

caesarian\_data$Delivery\_time[is.na(caesarian\_data$Delivery\_time)]<- mode\_impute(caesarian\_data$Delivery\_time)

caesarian\_data$Blood[is.na(caesarian\_data$Blood)]<- mode\_impute(caesarian\_data$Blood)

caesarian\_data <- na.omit(caesarian\_data)

vis\_miss(caesarian\_data)

**Output:**

**A computer code with blue text

Description automatically generated**

**A graph with text on it

Description automatically generated**

**Description:**

For Numerical Columns like Age and weight column, missing values were imputed with median or mean. For Categorical Columns like Gender, Delivery\_number, Delivery\_time, and Blood, the mode imputation function was constructed, and missing values were removed using the function. Then, rows with any remaining missing values were removed using the na.omit() function. Finally, using the vis\_miss(), present of missing values were visualized.

1. **Removing duplicates in rows:**

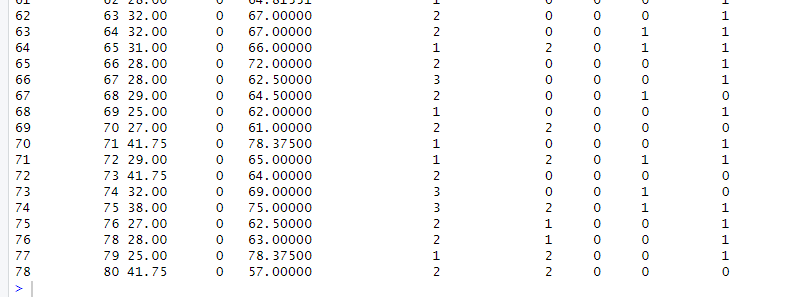
**Code:**

caesarian\_data<-distinct(caesarian\_data)

caesarian\_data<-distinct(caesarian\_data,Patient\_id, .keep\_all = TRUE)

caesarian\_data

**Output:**

****

**Description:**

Using distinct() function. Duplicates were removed. Then by calling the caesarian\_data it was checked whether duplicate values were removed or not.

1. **Perform & checking oversampling, undersampling, balanced sampling data:**

**Code:**

oversampled\_data <- ovun.sample(Caesarian ~ ., data = caesarian\_data, method = "over", p = 0.5, seed = 1)$data

table(oversampled\_data$Caesarian)

undersampled\_data <- ovun.sample(Caesarian ~ ., data = caesarian\_data, method = "under", p = 0.5, seed = 1)$data

table(undersampled\_data$Caesarian)

balanced\_data <- ovun.sample(Caesarian ~ ., data = caesarian\_data, method = "both", p = 0.5, seed = 1)$data

table(balanced\_data$Caesarian)

**Output:**

A screenshot of a computer code

Description automatically generated

**Description:**

The oversampling and under sampling on gender column was done and then the dataset was balanced and checked afterwards. Here, the ovun.sample function from the ROSE package performs oversampling and undersampling, with the method = "both" option combines oversampling and undersampling techniques to balance the data.

1. **Filter the dataset:**

**Code:**

filtered\_data <- caesarian\_data %>% filter(Gender == 1)

filtered\_data

**Output:**

**A screenshot of a computer

Description automatically generated**

**Description:**

The given dataset is about caesarian section results of pregnant women. Thus, the dataset is exclusively for female patients not male. So, the filter method was applied on the caesarian\_data and then stored in a new dataframe called filtered\_data. Then, the dataframe was displayed to see the outcome.

1. **Calculating mean, median, and mode:**

**Code:**

mean\_age <- mean(caesarian\_data$Age, na.rm = TRUE)

median\_age <- median(caesarian\_data$Age, na.rm = TRUE)

get\_mode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

mode\_age <- get\_mode(caesarian\_data$Age)

stats\_df <- data.frame(

Statistic = c("Mean", "Median", "Mode"),

Value = c(mean\_age, median\_age, mode\_age)

)

ggplot(stats\_df, aes(x = Statistic, y = Value, fill = Statistic)) +

geom\_bar(stat = "identity") +

geom\_text(aes(label = round(Value, 2)), vjust = -0.6) +

labs(title = "Mean, Median, and Mode of Age",

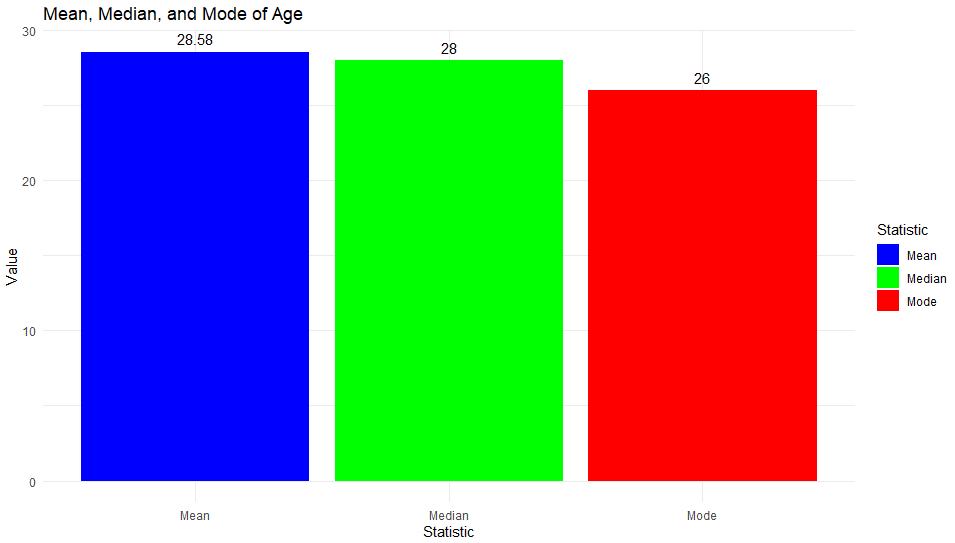
x = "Statistic",

y = "Value") +

theme\_minimal() +

scale\_fill\_manual(values = c("blue", "green", "red"))

**Output:**



**Description:**

Here, mean() and median() function was used to calculate the mean and median, and the na.rm = TRUE argument was used to remove any missing values before calculation. get\_mode() function identifies unique values in the data, then count occurrences using tabulate() then returns value with max count. stats\_df is a created dataframe and ggplot2 with geom\_bar() creates a bar graph.

1. **Age Distribution with mean, median and mode:**

**Code:**

plot\_mean <- ggplot(caesarian\_data, aes(x = Age)) +

geom\_histogram(binwidth = 1, fill = "blue", alpha = 0.7) +

geom\_vline(aes(xintercept = mean\_age), color = "red", linetype = "solid", size = 1) +

ggtitle("Age Distribution with Mean") + theme\_minimal() +

labs(x = "Age", y = "Frequency") + annotate("text", x = mean\_age, y = Inf, label = paste("Mean:", round(mean\_age, 2)), color = "red", vjust = 2)

plot\_median <- ggplot(caesarian\_data, aes(x = Age)) +

geom\_histogram(binwidth = 1, fill = "blue", alpha = 0.7) +

geom\_vline(aes(xintercept = median\_age), color = "green", linetype = "solid", size = 1) +

ggtitle("Age Distribution with Median") + theme\_minimal() + labs(x = "Age", y = "Frequency") + annotate("text", x = median\_age, y = Inf, label = paste("Median:", round(median\_age, 2)), color = "green", vjust = 2)

plot\_mode <- ggplot(caesarian\_data, aes(x = Age)) +

geom\_histogram(binwidth = 1, fill = "blue", alpha = 0.7) +

geom\_vline(aes(xintercept = mode\_age), color = "purple", linetype = "solid", size = 1) +

ggtitle("Age Distribution with Mode") + theme\_minimal() + labs(x = "Age", y = "Frequency") + annotate("text", x = mode\_age, y = Inf, label = paste("Mode:", round(mode\_age, 2)), color = "purple", vjust = 2)

plot\_mean

plot\_median

plot\_mode

**Output:**

A graph of a graph

Description automatically generated with medium confidence

**A graph with blue lines

Description automatically generated with medium confidence**

**A graph with blue lines

Description automatically generated**

**Description:**

Here, plot\_mode, plot\_median and plot\_mode was created and then called to display mean, median and mode. The geom\_histogram() creates a histogram, geom\_vline() adds vertical line, and annotates values of mean, median and mode on the top of histogram.

1. **Summary statistics for Age:**

**Code:**

summary(caesarian\_data$Age)

**Output:**

****

**Description:**

To summarize the Age variable in the caesarian\_data dataset,the summery() function is used here. Here,

Min: Minimum age in the dataset.

1st Qu.: First quartile (25th percentile).

Median: Middle value (50th percentile).

Mean: Average age.

3rd Qu.: Third quartile (75th percentile).

Max: Maximum age in the dataset.

1. **Min-Max Normalization:**

**Code:**

normalize <- function(x) {

return((x - min(x)) / (max(x) - min(x)))

}

caesarian\_data$Age\_normalized <- normalize(caesarian\_data$Age)

head(caesarian\_data)

**Output:**

**A screenshot of a computer code

Description automatically generated**

**Description:**

A function called normalize is made and using it the age column is normalized and then displayed. This function normalizes a numeric vector x to a range between 0 and 1 by using the following formula: Normailized value = x min(x) / max(x) – min(X). Then the normalized function is called to normalize the Age column of the caesarian\_data dataset and then the result is stored in Age\_normalized. The head() function is called to show the first six rows of the dataset of Age\_normalized column.

1. **Displaying the structure of the cleaned dataset:**

**Code:**

str(caesarian\_data)

A screenshot of a computer code

Description automatically generated**Output:**

**Description:**

Finally, the str() is used here to see the structure of the dataset at the ending.

1. **Showing the cleaned dataset:**

**Code**

caesarian\_data

**Output:**

A screenshot of a computer

Description automatically generated

**Description:**

Finally, by simply writing down caesarian\_data, the final version of the cleaned data dataset is viewed to see and observe the result.