Introduction to Data Science

Midterm Project

**Project Description**

Apply data preparation steps (which can be applied) for the given data set. In this project, we are going to use a modified version of the Caesarian Section Classification Dataset which can be downloaded from the Teams. The original dataset can be found in the following link where the dataset description is available as well (you may need to log in to download the dataset).

<https://archive.ics.uci.edu/ml/datasets/Caesarian+Section+Classification+Dataset> .

**Project Deliverables**

* Submit the implemented R program (R file or Text file) in the Teams. During VIVA session, you will bring this implemented program and we may ask you to execute the program.
* Submit the report in the Teams. See the instruction section below for the report details. **Please bring a printed copy of the submitted report during the VIVA session.**

**Instructions**

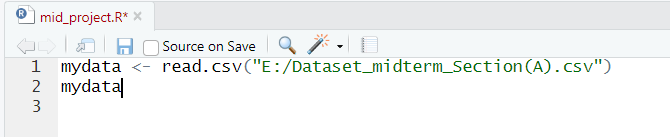
* The submission deadline for all deliverables is **March 16, 2024** (you must submit the assignment before **11:59 PM**).
* At the beginning of the report, write a short note about the dataset. You will get the dataset details from the above link provided for the dataset.
* For each implemented code segment in the R program, provide the code and its output along with their description in the report. In the description part, only write the content (do not write unnecessary content) that is sufficient to understand the code and its output.
* **Comments are not allowed in the R program**.
* The following topics can be focused to think about the project. **Note that the project is not limited to these topics which are mentioned to get an idea about how to proceed with the project.**
  + If there are any missing values in the dataset, then we will apply all applicable methods from the available options to handle the missing values.
  + We can see missing values on a graph.
  + We can show the mean, median, and mode on a graph.
  + We can convert continuous or numeric attributes into categorical attributes.
  + We can apply the normalization method only for one attribute.
  + If there exist any invalid data/outliers in the data set, then use the appropriate approach to handle those values.
  + We can see if there are any invalid values then we can recover those invalid values.

Project Overview:

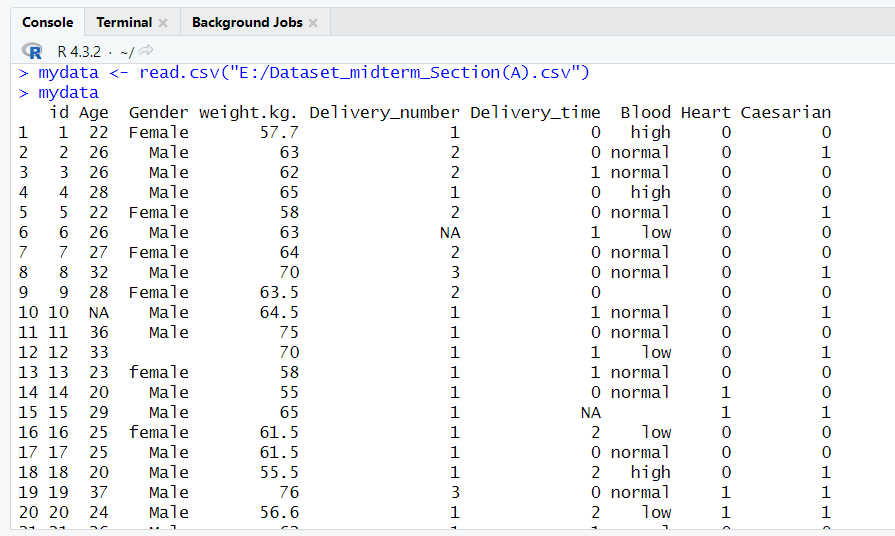
Our project revolves around analyzing factors influencing cesarean deliveries, encompassing various attributes such as age, gender, weight (in kg), delivery number, timing, blood pressure, and heart condition. Delivery types were categorized as timely (0), premature (1), and latecomer (2), while blood pressure and heart status were classified into high, low, normal, and apt, inept respectively. Initially, we imported the dataset into R Studio, and conducted exploratory data analysis through histograms, bar graphs, and box plots. We then scrutinized numeric attributes for alphabet presence and verified spelling accuracy in categorical ones. Missing values were handled by discarding null instances or replacing them with mode or average values. Categorical attributes were transformed into numerical ones using appropriate functions. Lastly, outliers were detected via box plots and handled using relevant techniques.

1. Load Dataset

Input:



Output:

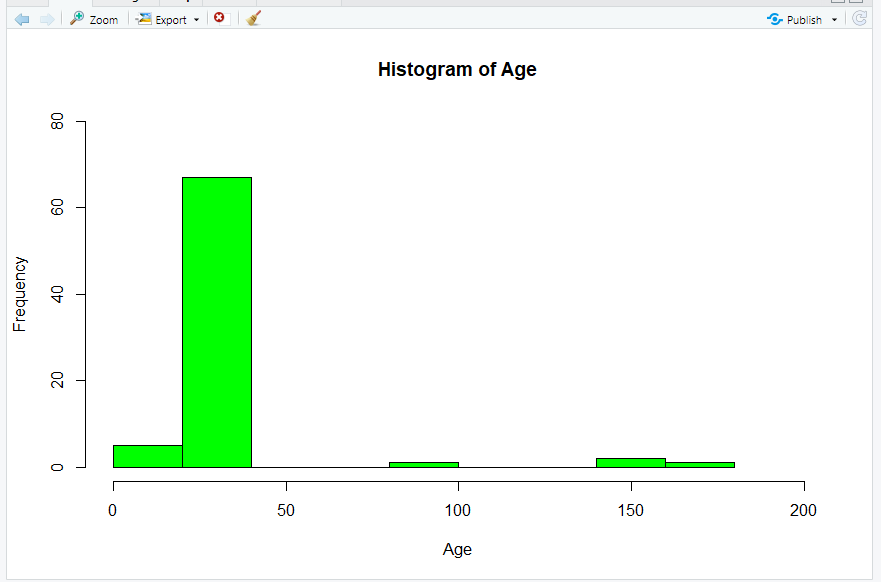


Description:  
The dataset was loaded into an R file in R Studio from Microsoft Excel with the help of the read.csv() function. Syntax read.csv (file, header = TRUE, sep = ",")

1. Data Exploration
2. Histogram of Feature: Age

Input: **hist(mydata$Age,main="Histogram of Age", xlab = "Age",xlim=c(0,200), ylim=c(0,80),col = "green", border = "black")**

Output:



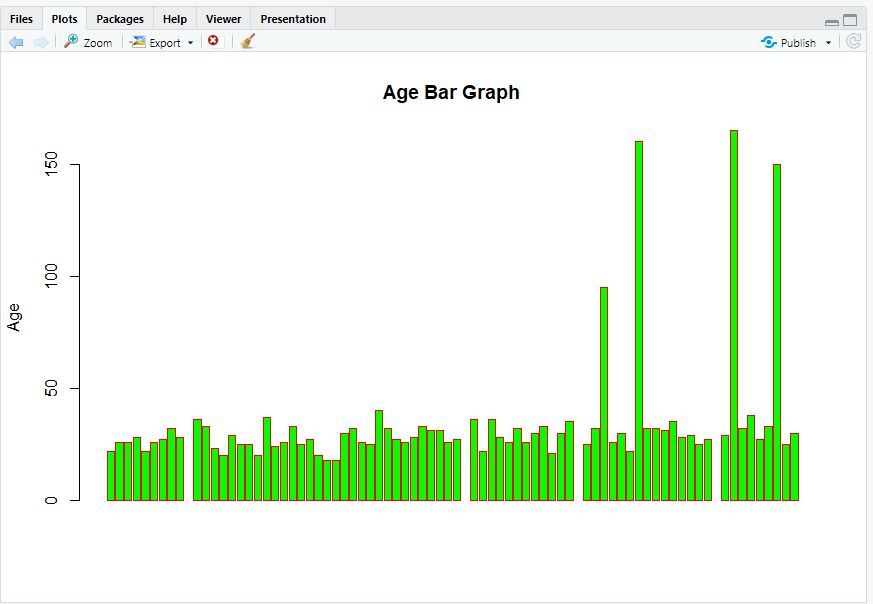
Description:  
Here the histogram of the age attribute was displayed using the hist() function. In the histogram the

The horizontal axis represented age (range 0 to 200) while the vertical axis represented the frequency or the number of people of a particular age range.

1. Bar graph of Feature: Age

Input: barplot(mydata$Age,ylab="Age",col="green", main="Age Bar Graph",border="red")

Output:

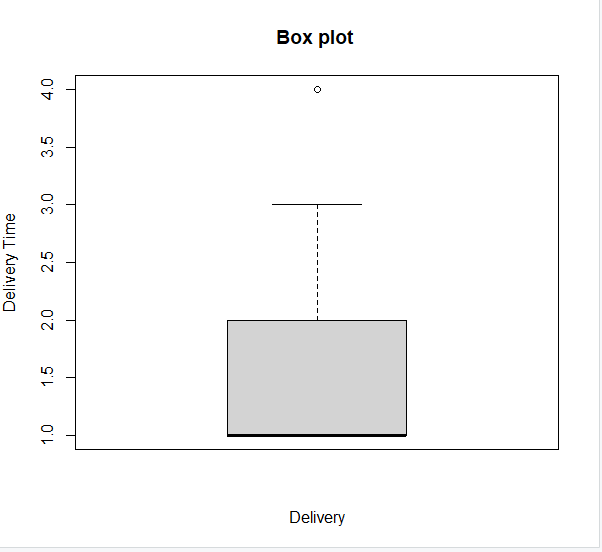


Description:  
Here the bar graph is displayed using the barplot() function. Here we can visualize that the majority of the people participating are aged below 50.

1. Box plot of Feature: Delivery Time

Input: boxplot(mydata$Delivery\_number,data = mydata, xlab = "Delivery",ylab = "Delivery Time", main = "Box plot ")

Output:



Description:  
The boxplot was displayed using the boxplot () function. The box plot suggests the delivery time types are somewhat symmetrically distributed.

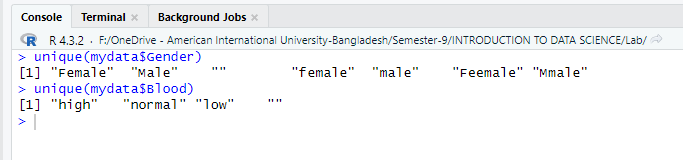
1. Processing Invalid Values
2. Checking spelling mistakes for categorical feature

Input:

unique(mydata$Gender)

unique(mydata$Blood)

Output:

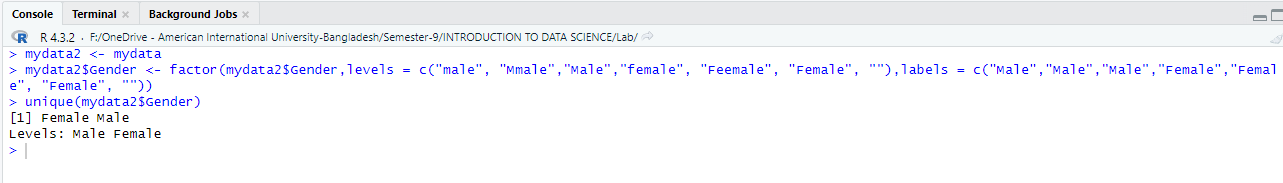


Description:  
Here the unique() function is used to find out the unique names in Gender attribute and Blood attribute(to find out spelling errors).

1. Replacing misspelled values with correctly spelled values:

Input: mydata2$Gender <- factor(mydata2$Gender,levels = c("male", "Mmale","Male","female", "Feemale", "Female", ""),labels = c("Male","Male","Male","Female","Female", "Female", ""))

Output:



Description:  
To correct the previously identified spelling mistakes in Gender attributes, the factor() function is used. It achieves this by mapping specific levels ("male", "Mmale", "Male", "female", "Feemale", "Female", and an empty string) to corresponding labels ("Male", "Male", "Male", "Female", "Female", "Female", and an empty string).

1. Checking for if there is any alphabet in numerical feature.

Input:

is.numeric(mydata$Age)

is.numeric(mydata$weight.kg.)

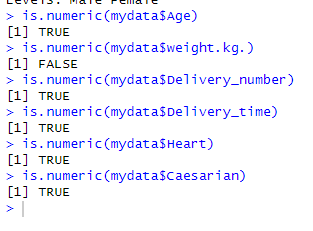
is.numeric(mydata$Delivery\_number)

is.numeric(mydata$Delivery\_time)

is.numeric(mydata$Heart)

is.numeric(mydata$Caesarian)

Output:



Description:  
Here the is.numeric() function is applied to check out whether any numerical attribute hosts any categorical value. If the output is TRUE it means all the values are numerical whereas it the output is false it indicates there are some categorical values in that numerical attribute.

1. Removing alphabets from numerical feature and converting into integer:

Input:

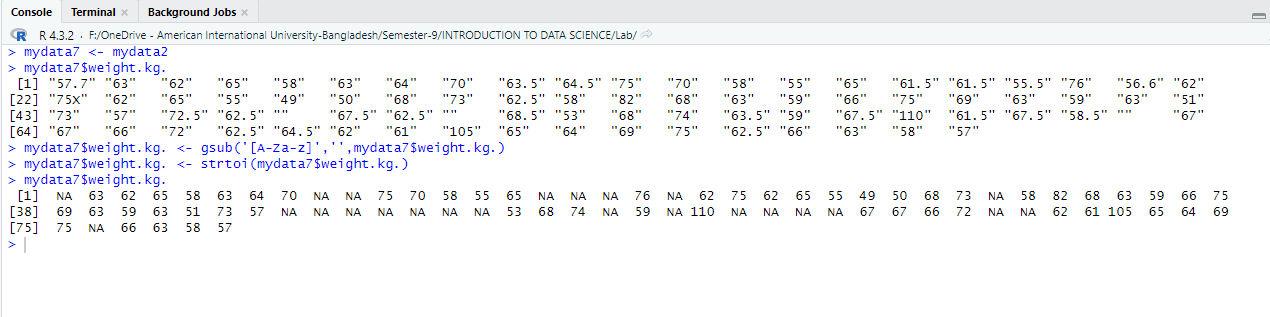
mydata7$weight.kg.

mydata7$weight.kg. <- gsub('[A-Za-z]','',mydata7$weight.kg.)

mydata7$weight.kg. <- strtoi(mydata7$weight.kg.)

mydata7$weight.kg.

Output:



Description:   
Here any alphabet in weight attribute is discarded using the gsub() function and the strtoi() converts the cleaned-up values (after removal of alphabetic characters) in the "weight.kg.” column to integers.

1. Missing Values

Input:

colSums(is.na(mydata))

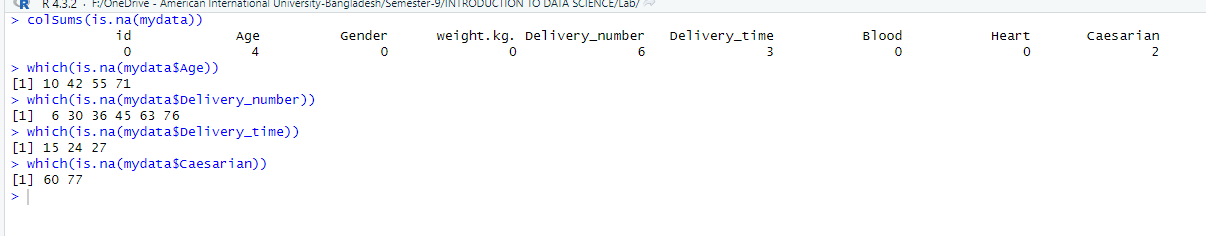
which(is.na(mydata$Age))

which(is.na(mydata$Delivery\_number))

which(is.na(mydata$Delivery\_time))

which(is.na(mydata$Caesarian))

Output:



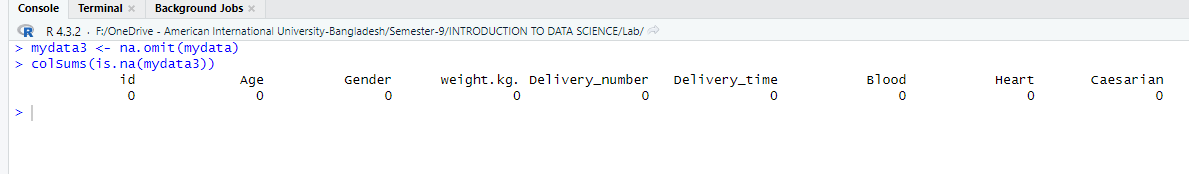
**Description**: This code snippet serves the purpose of detecting and locating missing values within specific columns of the dataset, which can be crucial for data preprocessing and quality assurance before analysis or modeling tasks.

colSums(is.na(mydata)): This line calculates the sum of missing values for each column in the dataset mydata. which(is.na(mydata$Age)): This line identifies the indices of missing values in the 'Age' column of the dataset. ‘Which(is.na(mydata$Delivery\_number)): Like the previous line, this line identifies the indices of missing values in the 'Delivery\_number' column of the dataset.which(is.na(mydata$Delivery\_time)): This line identifies the indices of missing values in the 'Delivery\_time' column of the dataset. ‘Which(is.na(mydata$Caesarian)): Finally, this line identifies the indices of missing values in the 'Caesarian' column of the dataset.

**Discard Null Instances**

mydata3 <- na.omit(mydata)

colSums(is.na(mydata3))

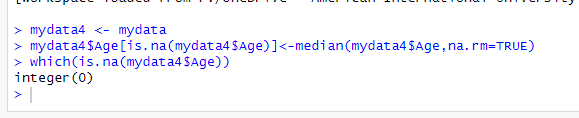


**Description:** The code removes rows with missing values from the dataset mydata, stored in mydata3. Then, it calculates and displays the sum of missing values for each column in the modified dataset to ensure data completeness.

**Fill Null values with Median.**

mydata4$Age[is.na(mydata4$Age)]<-median(mydata4$Age,na.rm=TRUE)

which(is.na(mydata4$Age))



**Description:** The code replaces missing values in the 'Age' column of dataset mydata4 with the median value of the 'Age' column, excluding NA values. It then checks for any remaining NA values in the 'Age' column to ensure successful imputation. This process ensures data completeness by filling null values with a representative measure.

Fill Null values with Mode

mydata4$Delivery\_number[is.na(mydata4$Delivery\_number)]<-mode(mydata4$Delivery\_number)

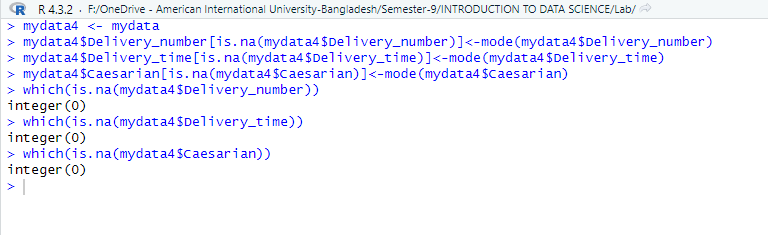
mydata4$Delivery\_time[is.na(mydata4$Delivery\_time)]<-mode(mydata4$Delivery\_time)

mydata4$Caesarian[is.na(mydata4$Caesarian)]<-mode(mydata4$Caesarian)

which(is.na(mydata4$Delivery\_number))

which(is.na(mydata4$Delivery\_time))

which(is.na(mydata4$Caesarian))



**Description:** These codes perform mode imputation for missing values in the specified columns, ensuring that the dataset is devoid of NA values in these columns afterward.

mydata4$Delivery\_number[is.na(mydata4$Delivery\_number)] <- mode(mydata4$Delivery\_number): This line fills missing values in the 'Delivery\_number' column of mydata4 with the mode (most frequent value) of the 'Delivery\_number' column.

mydata4$Delivery\_time[is.na(mydata4$Delivery\_time)] <- mode(mydata4$Delivery\_time): This line replaces NA values in the 'Delivery\_time' column of mydata4 with the mode of the 'Delivery\_time' column.

mydata4$Caesarian[is.na(mydata4$Caesarian)] <- mode(mydata4$Caesarian): This line fills any missing values in the 'Caesarian' column of mydata4 with the mode of the 'Caesarian' column.

1. Data transformation

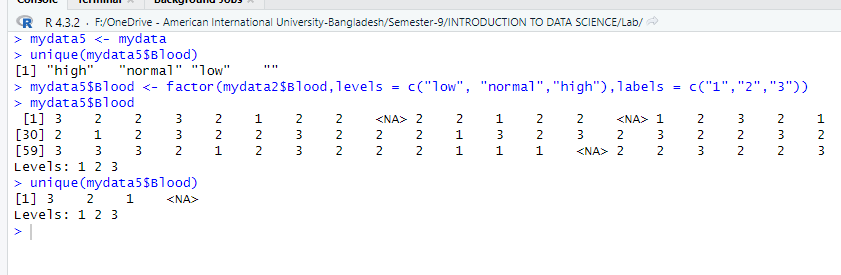
Representing categorical value with numbers

unique(mydata5$Blood)

mydata5$Blood <- factor(mydata2$Blood,levels = c("low", "normal","high"),labels = c("1","2","3"))

mydata5$Blood

unique(mydata5$Blood)



**Description:** unique(mydata5$Blood): This line displays the unique values present in the 'Blood' column of the dataset mydata5, showing the distinct categories of blood levels.

mydata5$Blood <- factor(mydata2$Blood, levels = c("low", "normal", "high"), labels = c("1", "2", "3")): This line converts the categorical variable 'Blood' in the dataset mydata5 into a factor with specified levels ("low", "normal", "high") and corresponding labels ("1", "2", "3"), representing each category with a numerical value.

mydata5$Blood: This line prints out the 'Blood' column of the dataset mydata5 after the transformation, displaying the updated representation of categorical values with numbers.

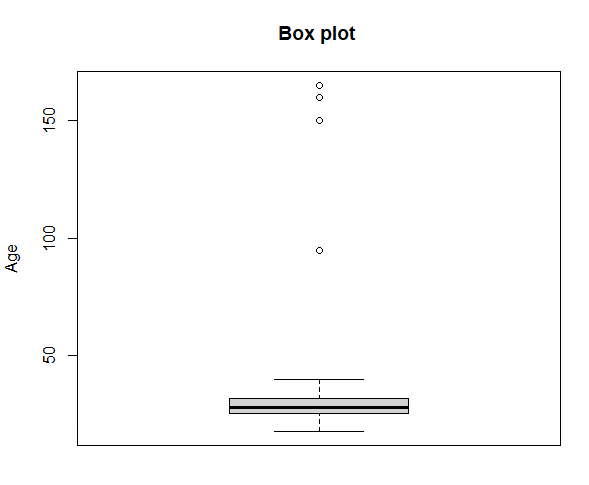
unique(mydata5$Blood): This line shows the unique values present in the 'Blood' column of the dataset mydata5 after the transformation, confirming the successful conversion of categorical values to numerical representations.

1. Outliers

Checking Feature: Age for outliers using Boxplot

boxplot(mydata$Age,data = mydata, ylab = "Age", main = "Box plot ")

**Description:** The provided code creates a box plot to visualize the distribution of ages in the dataset mydata. The boxplot() function is used with the 'Age' column as the data, specifying the y-axis label as "Age" and setting the main title of the plot to "Box plot". This plot provides insights into the central tendency, spread, and presence of outliers within the age data.



Removing outliers using Boxplot and replacing with NA

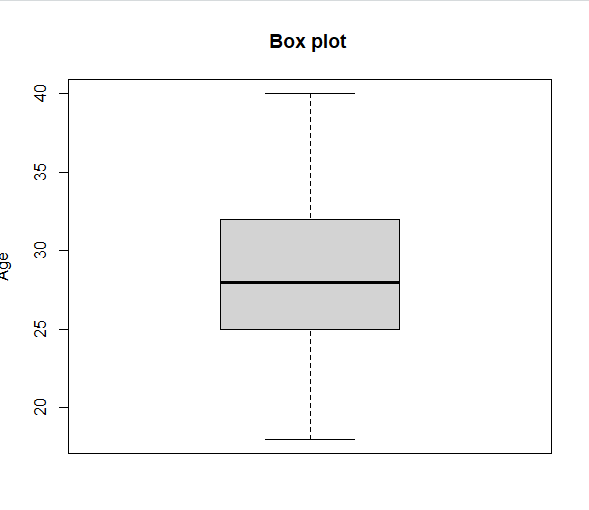
for(x in boxplot.stats(mydata6$Age)$out){

index = which(mydata6$Age==x)

mydata6[,2][index] <- NA

}

boxplot(mydata6$Age,data = mydata, ylab = "Age", main = "Box plot ")



**Description**: for(x in boxplot.stats(mydata6$Age)$out): This line initiates a loop where the variable x iterates over each outlier detected in the 'Age' column of the dataset mydata6, as identified by boxplot.stats().

index = which(mydata6$Age==x): For each outlier x, this line finds the indices of rows in the 'Age' column of mydata6 where the value equals x, using the which() function.

mydata6[,2][index] <- NA: This line assigns NA (missing) values to the identified outliers' indices in the 'Age' column of mydata6, effectively removing outliers by replacing them with missing values.

boxplot(mydata6$Age,data = mydata, ylab = "Age", main = "Box plot "): Finally, this line generates a box plot of the 'Age' column from the modified dataset mydata6, displaying the distribution of ages after outlier removal.