AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

Assignment Cover Sheet

Assignment Title:	Mid Term Project			
Assignment No:	2		Date of Submission:	11 November 2023
Course Title:	INTRODUCTION TO DATA SCIENCE			
Course Code:	CSC4180		Section:	B
Semester:	Spring	2023-24	Course Teacher:	TOHEDUL ISLAM

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaborationhas been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
7. I/we understand thatPlagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.: 8

No	Name	ID	Program	Signature
1	Borshon Alfred Gomes	21-44561-1	CSE	
2	Rianul Amin Rian	21-44589-1	CSE	
3			Choose an item.	
4			Choose an item.	
5			Choose an item.	
6			Choose an item.	
7			Choose an item.	
8			Choose an item.	
9			Choose an item.	
10			Choose an item.	

Faculty use only		
FACULTYCOMMENTS	**Marks Obtained**	
	Total Marks	

- ## About the dataset

Patients' medical, demographic, and diabetes status (positive or negative) are all collected into the Diabetes Prediction Dataset. Information about age, gender, blood pressure, heart disease, smoking history, body mass index (BMI), HbA1c level, and blood sugar level are among the features included in the report. Using this dataset, machine learning models can be developed to predict a patient's likelihood of developing diabetes based on their medical history and demographic data. This can be helpful for medical professionals for identifying individuals who may be at risk of developing diabetes and creating individualized treatment strategies. Researchers can also utilize the dataset to investigate the associations between different demographic and medical characteristics and the risk of developing diabetes.

- **Importing the dataset**

Code

```
dataset <- read.csv('C:/Users/User/Desktop/Dataset_MIdterm_sectoin(B).csv', na.strings = c(""))
dataset
```

Output

```
> dataset
    gender age hypertension heart_disease smoking_history    bmi HbA1c_level blood_glucose_level
1   Female  80            0             1           never  25.19         6.6                 140
2   Female  54            0             0         No Info  27.32         6.6                  80
3     Male  28            0             0           never -27.32         5.7                 158
4   Female  NA            0             0         current  23.45         5.0                 155
5     Male  76            1             1         current  20.14         4.8                 155
6   Female  20            0             0           never  27.32         6.6                  85
7     <NA>  79            0             0         No Info  23.86         5.7                  85
8     Male  42            0             0           never  33.64         4.8                 145
9   Female  32            0             0           never  27.32         5.0                 100
10  Female  53            0             0           never  27.32         6.1                  85
11  Female  54            0             0          former  54.70         6.0                 100
12  Female  78           NA             0          former  36.05         5.0                 130
13  Female  67            0             0           never  25.69         5.8                 200
14  Female  76            0             0         No Info  27.32         5.0                 160
15    <NA>  78            0             0         No Info  27.32         6.6                 126
16    Male  15            0             0           never  30.36         6.1                 200
17  Female  42            0             0           never  24.48         5.7                 158
18  Female  42            0             0         No Info  27.32         5.7                  80
19    Male  NA            0             0            ever  25.72         3.5                 159
20    Male  40            0             0         current  36.38         6.0                  90
21    Male   5            0             0         No Info  18.80         6.2                  85
```

Explanation

First, the provided dataset was converted from XLSX to CSV format. Then the dataset was imported in Rstudio using the method read.csv. Two parameters were passed, first is the location of the csv file and then **na.strings = c("")** which replaces empty strings with NAs.

- **Detecting NA values in dataset**

Code

is.na(dataset)
sum(is.na(dataset))

Output

```
> is.na(dataset)
       gender   age hypertension heart_disease smoking_history   bmi HbA1c_level
  [1,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
  [2,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
  [3,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
  [4,]  FALSE  TRUE        FALSE         FALSE           FALSE FALSE       FALSE
  [5,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
  [6,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
  [7,]   TRUE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
  [8,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
  [9,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [10,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [11,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [12,]  FALSE FALSE         TRUE         FALSE           FALSE FALSE       FALSE
 [13,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [14,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [15,]   TRUE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [16,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [17,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
 [18,]  FALSE FALSE        FALSE         FALSE           FALSE FALSE       FALSE
```

```
> sum(is.na(dataset))
[1] 11
>
```

Explanation

is.na method returns **TRUE** if NA value exists and **FALSE** if it doesn't. **sum(is.na(dataset))** returns the number of NA values in the dataset.

- **Deleting instances with missing values**

Code

```
dataset1 <- na.omit(dataset)
dataset1
```

Output

```
> dataset1 <- na.omit(dataset)
> dataset1
     gender age hypertension heart_disease smoking_history    bmi HbA1c_level
1    Female  80            0             1           never  25.19         6.6
2    Female  54            0             0         No Info  27.32         6.6
3      Male  28            0             0           never -27.32         5.7
5      Male  76            1             1         current  20.14         4.8
6    Female  20            0             0           never  27.32         6.6
8      Male  42            0             0           never  33.64         4.8
9    Female  32            0             0           never  27.32         5.0
10   Female  53            0             0           never  27.32         6.1
11   Female  54            0             0          former  54.70         6.0
13   Female  67            0             0           never  25.69         5.8
14   Female  76            0             0         No Info  27.32         5.0
16     Male  15            0             0           never  30.36         6.1
17   Female  42            0             0           never  24.48         5.7
18   Female  42            0             0         No Info  27.32         5.7
20     Male  40            0             0         current  36.38         6.0
21     Male   5            0             0         No Info  18.80         6.2
22   Female  69            0             0           never  21.24         4.8
```

Explanation

na.omit method removes all the NA values from a data object. After removing the NA values, the result was stored in **dataset1**.

- **Replacing NA values with mode for Gender**

Code

```
missingValueLocations <- which(is.na(dataset$gender))
genderMode <- names(sort(table(dataset$gender[!is.na(dataset$gender)]), decreasing
= TRUE)[1])
dataset$gender[missingValueLocations] <- genderMode
cat("Gender Mode:", genderMode) dataset$gender
```

Output

```
> missingValueLocations <- which(is.na(dataset$gender))
> genderMode <- names(sort(table(dataset$gender[!is.na(dataset$gender)]), decreasing = TRUE)[1])
> dataset$gender[missingValueLocations] <- genderMode
> cat("Gender Mode:", genderMode)
Gender Mode: Female
> dataset$gender
  [1] "Female" "Female" "Male"   "Female" "Male"   "Female" "Female" "Male"   "Female" "Female" "Female" "Female"
 [13] "Female" "Female" "Female" "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Female" "Female" "Female"
 [25] "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Female" "Female" "Female" "Male"   "Female" "Female"
 [37] "Male"   "Female" "Female" "Female" "Female" "Female" "Female" "Female" "Female" "Male"   "Female" "Female"
 [49] "Female" "Female" "Female" "Female" "Female" "Female" "Female" "Male"   "Male"   "Male"   "Male"   "Female"
 [61] "Male"   "Female" "Female" "Female" "Female" "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Male"
 [73] "Female" "Male"   "Female" "Male"   "Female" "Female" "Female" "Female" "Female" "Male"   "Female" "Male"
 [85] "Female" "Male"   "Female" "Male"   "Male"   "Female" "Male"   "Male"   "Female" "Female" "Female" "Female"
 [97] "Male"   "Female" "Male"   "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Female" "Male"
[109] "Female" "Female" "Female" "Female" "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Female" "Female"
>
```

Explanation

First, locations of the NA values in the gender column were stored in **missingValueLocations** using the **which(is.na(dataset$gender))** method which returns the indexes of the NA values. Then **table** method was used which returns a categorical representation of data with variable name and the frequency in the form of a table. **dataset$gender[!is.na(dataset$gender)]** was passed as an argument to tabulate only the gender column where the values are not NA. After that the result was sorted using the **sort** method. **decreasing = TRUE** was used in order to sort the result in a decreasing order. Then the **names** method returned the name of index 1 and the result was stored in **genderMode. cat** method was used to print the mode value and lastly all the NA values in the gender column were replaced by **genderMode.**

- **Replacing NA values with mode for Hypertension**

Code

```
missingValueLocations <- which(is.na(dataset$hypertension))
hypertensionMode <- names(sort(table(dataset$hypertension[!is.na(dataset$hypertension)]),
decreasing = TRUE)[1])
dataset$hypertension[missingValueLocations] <-
hypertensionMode cat("Hypertension Mode:", hypertensionMode)
dataset$hypertension
```

Output

```
> missingValueLocations <- which(is.na(dataset$hypertension))
> hypertensionMode <- names(sort(table(dataset$hypertension[!is.na(dataset$hypertension)]), decreasing = TRUE)[1])
> dataset$hypertension[missingValueLocations] <- hypertensionMode
> cat("Hypertension Mode:", hypertensionMode)
Hypertension Mode: 0
> dataset$hypertension
  [1] "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
 [19] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
 [37] "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
 [55] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0"
 [73] "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0"
 [91] "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "0"
[109] "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0"
>
```

Explanation

First, locations of the NA values in the hypertension column were stored in **missingValueLocations** using the **which(is.na(dataset$hypertension))** method which returns the indexes of the NA values. Then **table** method was used which returns a categorical representation of data with variable name and the frequency in the form of a table. **dataset$hypertension[!is.na(dataset$hypertension)]** was passed as an argument to tabulate only the hypertension column where the values are not NA. After that the result was sorted using the **sort** method. **decreasing = TRUE** was used in order to sort the result in a decreasing order. Then the **names** method returned the name of index 1 and the result was stored in **hypertensionMode. cat** method was used to print the mode value and lastly all the NA values in the hypertension column were replaced by **hypertensionMode.**

- **Replacing NA values with mode for Smoking History**

Code

```
missingValueLocations <- which(is.na(dataset$smoking_history))
smokingHistoryMode <-
names(sort(table(dataset$smoking_history[!is.na(dataset$smoking_history)]), decreasing =
TRUE)[1])
dataset$smoking_history[missingValueLocations] <-
smokingHistoryMode cat("Smoking History Mode:",
smokingHistoryMode) dataset$smoking_history
```

Output

```
> missingValueLocations <- which(is.na(dataset$smoking_history))
> smokingHistoryMode <- names(sort(table(dataset$smoking_history[!is.na(dataset$smoking_history)]), decreasing = TRUE)[1])
> dataset$smoking_history[missingValueLocations] <- smokingHistoryMode
> cat("Smoking History Mode:", smokingHistoryMode)
Smoking History Mode: never
> dataset$smoking_history
  [1] "never"       "No Info"     "never"       "current"     "current"     "never"       "No Info"
  [8] "never"       "never"       "never"       "former"      "former"      "never"       "No Info"
 [15] "No Info"     "never"       "never"       "No Info"     "ever"        "current"     "No Info"
 [22] "never"       "former"      "No Info"     "never"       "former"      "never"       "never"
 [29] "No Info"     "No Info"     "current"     "never"       "never"       "No Info"     "No Info"
 [36] "never"       "No Info"     "never"       "No Info"     "No Info"     "never"       "never"
 [43] "never"       "never"       "No Info"     "never"       "No Info"     "No Info"     "never"
 [50] "current"     "former"      "not current" "ever"        "never"       "No Info"     "never"
 [57] "No Info"     "No Info"     "No Info"     "never"       "never"       "never"       "never"
 [64] "No Info"     "never"       "never"       "former"      "No Info"     "never"       "not current"
 [71] "current"     "former"      "former"      "former"      "never"       "No Info"     "current"
 [78] "never"       "never"       "never"       "never"       "never"       "never"       "current"
 [85] "not current" "never"       "never"       "No Info"     "not current" "never"       "ever"
 [92] "never"       "current"     "never"       "former"      "never"       "never"       "never"
 [99] "never"       "former"      "current"     "current"     "No Info"     "No Info"     "never"
[106] "current"     "former"      "never"       "never"       "former"      "never"       "ever"
[113] "never"       "never"       "not current" "ever"        "never"       "ever"        "No Info"
[120] "No Info"
>
```

Explanation

First, locations of the NA values in the smoking_history column were stored in **missingValueLocations** using the **which(is.na(dataset$smoking_history))** method which returns the indexes of the NA values. Then **table** method was used which returns a categorical representation of data with variable name and the frequency in the form of a table. **dataset$smoking_history[!is.na(dataset$smoking_history)]** was passed as an argument to tabulate only the smoking_history column where the values are not NA. After that the result was sorted using the **sort** method. **decreasing = TRUE** was used in order to sort the result in a decreasing order. Then the **names** method returned the name of index 1 and the result was stored in **smokingHistoryMode. cat** method was used to print the mode value and lastly all the NA values in the smoking_history column were replaced by **smokingHistoryMode.**

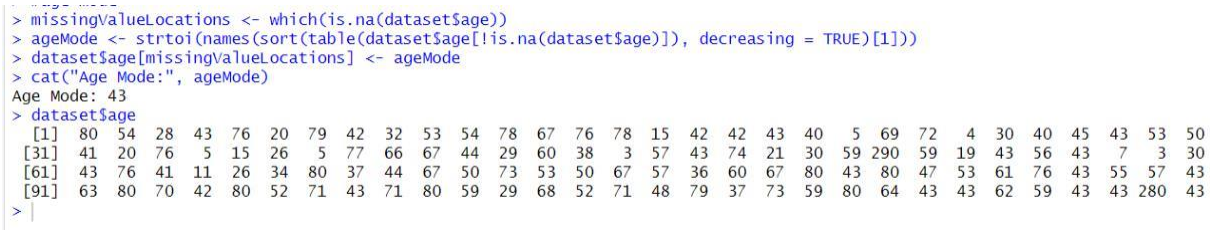
- **Replacing NA values with mode for Age**

Code

```
missingValueLocations <- which(is.na(dataset$age))
ageMode <- strtoi(names(sort(table(dataset$age[!is.na(dataset$age)]), decreasing = TRUE)[1]))
dataset$age[missingValueLocations] <- ageMode cat("Age Mode:", ageMode)

dataset$age
```

Output



```
> missingValueLocations <- which(is.na(dataset$age))
> ageMode <- strtoi(names(sort(table(dataset$age[!is.na(dataset$age)]), decreasing = TRUE)[1]))
> dataset$age[missingValueLocations] <- ageMode
> cat("Age Mode:", ageMode)
Age Mode: 43
> dataset$age
  [1]  80  54  28  43  76  20  79  42  32  53  54  78  67  76  78  15  42  42  43  40   5  69  72   4  30  40  45  43  53  50
 [31]  41  20  76   5  15  26   5  77  66  67  44  29  60  38   3  57  43  74  21  30  59 290  59  19  43  56  43   7   3  30
 [61]  43  76  41  11  26  34  80  37  44  67  50  73  53  50  67  57  36  60  67  80  43  80  47  53  61  76  43  55  57  43
 [91]  63  80  70  42  80  52  71  43  71  80  59  29  68  52  71  48  79  37  73  59  80  64  43  43  62  59  43  43 280  43
>
```

Explanation

First, locations of the NA values in the age column were stored in **missingValueLocations** using the **which(is.na(dataset$age))** method which returns the indexes of the NA values. Then **table** method was used which returns a categorical representation of data with variable name and the frequency in the form of a table. **dataset$age[!is.na(dataset$age)]** was passed as an argument to tabulate only the age column where the values are not NA. After that the result was sorted using the **sort** method. **decreasing = TRUE** was used in order to sort the result in a decreasing order. Then the **names** method returned the name of index 1 and the result was stored in **ageMode. cat** method was used to print the mode value and lastly all the NA values in the age column were replaced by **ageMode.**

- **Replacing NA values with median for Age**

Code

```
missingValueLocations <- which(is.na(dataset$age))
ageMedian <- floor(median(dataset$age, na.rm =
TRUE)) dataset$age[missingValueLocations] <-
ageMedian cat("Age Median:", ageMedian) dataset$age
```

Output

```
> missingValueLocations <- which(is.na(dataset$age))
> ageMedian <- floor(median(dataset$age, na.rm = TRUE))
> dataset$age[missingValueLocations] <- ageMedian
> cat("Age Median:", ageMedian)
Age Median: 52
> dataset$age
  [1]  80  54  28  52  76  20  79  42  32  53  54  78  67  76  78  15  42  42  52  40   5  69  72   4  30  40  45
 [28]  43  53  50  41  20  76   5  15  26   5  77  66  67  44  29  60  38   3  57  43  74  21  30  59 290  59  19
 [55]  52  56  43   7   3  30  43  76  41  11  26  34  80  37  44  67  50  73  53  50  67  57  36  60  67  80  52
 [82]  80  47  53  61  76  43  55  57  43  63  80  70  42  80  52  71  43  71  80  59  29  68  52  71  48  79  37
[109]  73  59  80  64  43  43  62  59  43  43 280  43
>
```

Explanation

First, locations of the NA values in the age column were stored in **missingValueLocations**. Then, the median value of the age column was stored in a variable **ageMedian** using the **median** method. **na.rm = TRUE** was passed as an argument so that the NA values are not in consideration while calculating median. **floor** was used in order to convert the result to an integer value. **cat** method was used to print the mode value and lastly all the NA values in the age column were replaced by **ageMedian.**

- **Replacing NA values with mean for Age**

Code

```
missingValueLocations <- which(is.na(dataset$age))
ageMean <- floor(mean(dataset$age, na.rm = TRUE))
dataset$age[missingValueLocations] <- ageMean
cat("Age Mean:", ageMean) dataset$age
```

Output

```
> missingValueLocations <- which(is.na(dataset$age))
> ageMean <- floor(mean(dataset$age, na.rm = TRUE))
> dataset$age[missingValueLocations] <- ageMean
> cat("Age Mean:", ageMean)
Age Mean: 54
> dataset$age
  [1]  80  54  28  52  76  20  79  42  32  53  54  78  67  76  78  15  42  42  52  40   5  69  72   4  30  40  45  43
 [29]  53  50  41  20  76   5  15  26   5  77  66  67  44  29  60  38   3  57  43  74  21  30  59 290  59  19  52  56
 [57]  43   7   3  30  43  76  41  11  26  34  80  37  44  67  50  73  53  50  67  57  36  60  67  80  52  80  47  53
 [85]  61  76  43  55  57  43  63  80  70  42  80  52  71  43  71  80  59  29  68  52  71  48  79  37  73  59  80  64
[113]  43  43  62  59  43  43 280  43
>
```

Explanation

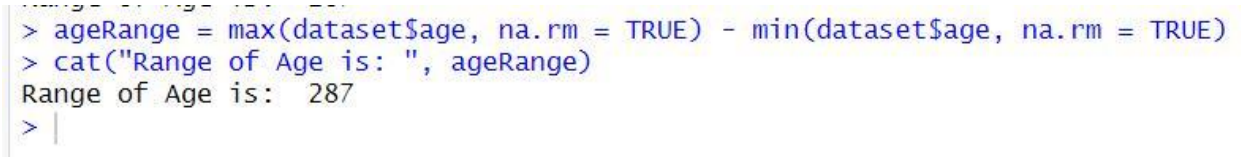
First, locations of the NA values in the age column were stored in **missingValueLocations**. Then, the mean value of the age column was stored in a variable **ageMean** using the **mean** method. **na.rm = TRUE** was passed as an argument so that the NA values are not in consideration while calculating mean. **floor** was used in order to convert the result to an integer value. **cat** method was used to print the mean value and lastly all the NA values in the age column were replaced by **ageMean.**

- **Range of Age**

Code

```
ageRange = max(dataset$age, na.rm = TRUE) - min(dataset$age, na.rm = TRUE)
cat("Range of Age is: ", ageRange)
```

Output



Explanation

The range can be defined as the difference between the maximum and minimum elements in the given data. **max** method returns the maximum value in age column and **min** method returns the minimum. **na.rm = TRUE** was used to ignore the NA values. The result of the subtraction was stored in **ageRange** which was then printed by the **cat** method.

- **Range of BMI**

Code

```
bmiRange = max(dataset$bmi, na.rm = TRUE) - min(dataset$bmi, na.rm = TRUE)
cat("Range of BMI is: ", bmiRange)
```

Output

```
> bmiRange = max(dataset$bmi, na.rm = TRUE) - min(dataset$bmi, na.rm = TRUE)
> cat("Range of BMI is: ", bmiRange)
Range of BMI is:  1.24
>
```

Explanation

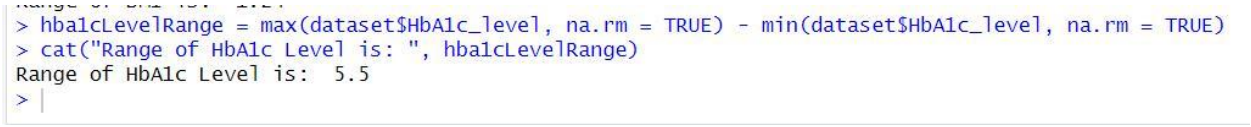
The range can be defined as the difference between the maximum and minimum elements in the given data. **max** method returns the maximum value in age column and **min** method returns the minimum. **na.rm = TRUE** was used to ignore the NA values. The result of the subtraction was stored in **bmiRange** which was then printed by the **cat** method.

- **Range of HbA1c Level**

Code

```
hba1cLevelRange = max(dataset$HbA1c_level, na.rm = TRUE) -
min(dataset$HbA1c_level, na.rm = TRUE)
cat("Range of HbA1c Level is: ", hba1cLevelRange)
```

Output



Explanation

The range can be defined as the difference between the maximum and minimum elements in the given data. **max** method returns the maximum value in age column and **min** method returns the minimum. **na.rm = TRUE** was used to ignore the NA values. The result of the subtraction was stored in **hba1cLevelRange** which was then printed by the **cat** method.

- **Range of Blood Glucose Level**

Code

```
bloodGlucoseLevelRange = max(dataset$blood_glucose_level, na.rm = TRUE) -
min(dataset$blood_glucose_level, na.rm = TRUE)
cat("Range of Blood Glucose Level is: ", bloodGlucoseLevelRange)
```

Output

```
> bloodGlucoseLevelRange = max(dataset$blood_glucose_level, na.rm = TRUE) - min(dataset$blood_glucose_level, na.rm = TRUE)
> cat("Range of Blood Glucose Level is: ", bloodGlucoseLevelRange)
Range of Blood Glucose Level is:  220
>
```

Explanation

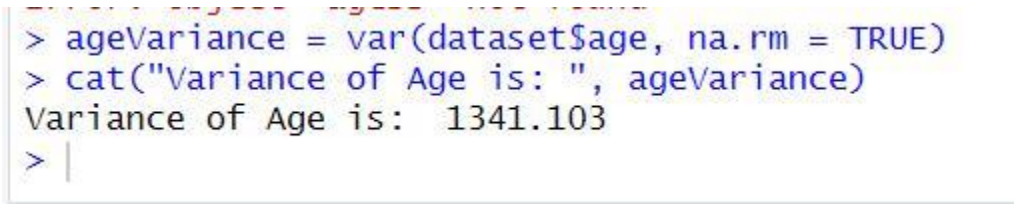
The range can be defined as the difference between the maximum and minimum elements in the given data. **max** method returns the maximum value in age column and **min** method returns the minimum. **na.rm = TRUE** was used to ignore the NA values. The result of the subtraction was stored in **bloodGlucoseLevelRange** which was then printed by the **cat** method.

- **Variance of Age**

Code

```
ageVariance = var(dataset$age, na.rm = TRUE)
cat("Variance of Age is: ", ageVariance)
```

Output



Explanation

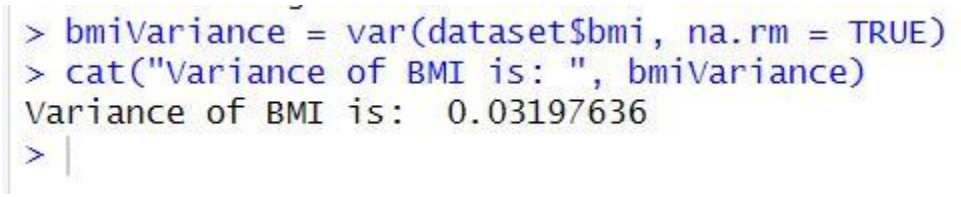
var is the method used to calculate variance. The age column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **ageVariance** variable which was then printed by the **cat** method.

- **Variance of BMI**

Code

```
bmiVariance = var(dataset$bmi, na.rm = TRUE)
cat("Variance of BMI is: ", bmiVariance)
```

Output



Explanation

var is the method used to calculate variance. The bmi column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **bmiVariance** variable which was then printed by the **cat** method.

- **Variance of HbA1c Level**

Code

```
hba1cLevelVariance = var(dataset$HbA1c_level, na.rm = TRUE)
cat("Variance of HbA1c Level is: ", hba1cLevelVariance)
```

Output

```
> hba1cLevelVariance = var(dataset$HbA1c_level, na.rm = TRUE)
> cat("Variance of HbA1c Level is: ", hba1cLevelVariance)
Variance of HbA1c Level is:  1.910294
>
```

Explanation

var is the method used to calculate variance. The HbA1c_level column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **hba1cLevelVariance** variable which was then printed by the **cat** method.

- **Variance of Blood Glucose Level**

Code

```
bloodGlucoseLevelVariance = var(dataset$blood_glucose_level, na.rm = TRUE)
cat("Variance of Blood Glucose Level is: ", bloodGlucoseLevelVariance)
```

Output

```
> bloodGlucoseLevelVariance = var(dataset$blood_glucose_level, na.rm = TRUE)
> cat("Variance of Blood Glucose Level is: ", bloodGlucoseLevelVariance)
Variance of Blood Glucose Level is:  2573.853
>
```

Explanation

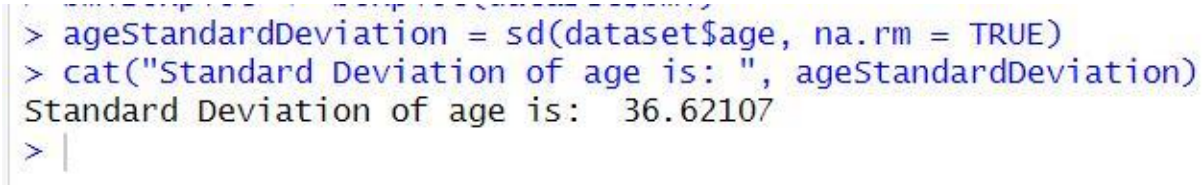
var is the method used to calculate variance. The blood_glucose_level column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **bloodGlucoseLevelVariance** variable which was then printed by the **cat** method.

- **Standard Deviation of Age**

Code

```
ageStandardDeviation = sd(dataset$age, na.rm = TRUE)
cat("Standard Deviation of age is: ", ageStandardDeviation)
```

Output



Explanation

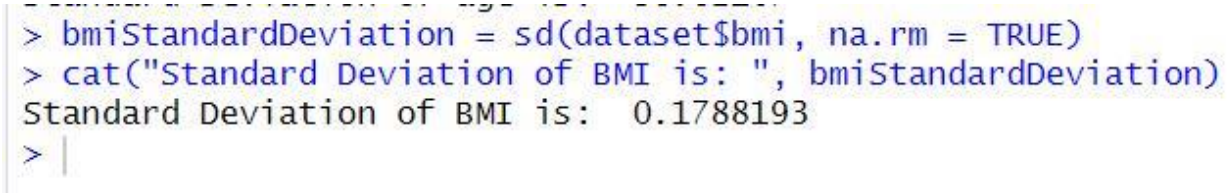
sd is the method used to calculate standard deviation. The age column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **ageStandardDeviation** variable which was then printed by the **cat** method.

- **Standard Deviation of BMI**

Code

```
bmiStandardDeviation = sd(dataset$bmi, na.rm = TRUE)
cat("Standard Deviation of BMI is: ", bmiStandardDeviation)
```

Output



Explanation

sd is the method used to calculate standard deviation. The bmi column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **bmiStandardDeviation** variable which was then printed by the **cat** method.

- **Standard Deviation of HbA1c Level**

Code

```
hba1cLevelStandardDeviation = sd(dataset$HbA1c_level, na.rm = TRUE)
cat("Standard Deviation of HbA1c Level is: ", hba1cLevelStandardDeviation)
```

Output

```
> hba1cLevelStandardDeviation = sd(dataset$HbA1c_level, na.rm = TRUE)
> cat("Standard Deviation of HbA1c Level is: ", hba1cLevelStandardDeviation)
Standard Deviation of HbA1c Level is:  1.382134
>
```

Explanation

sd is the method used to calculate standard deviation. The HbA1c_level column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **hba1cLevelStandardDeviation** variable which was then printed by the **cat** method.

- **Standard Deviation of Blood Glucose Level**

Code

```
bloodGlucoseLevelStandardDeviation = sd(dataset$ blood_glucose_level, na.rm = TRUE)
cat("Standard Deviation of Blood Glucose Level is: ", bloodGlucoseLevelStandardDeviation)
```

Output

```
> bloodGlucoseLevelStandardDeviation = sd(dataset$ blood_glucose_level, na.rm = TRUE)
> cat("Standard Deviation of Blood Glucose Level is: ", bloodGlucoseLevelStandardDeviation)
Standard Deviation of Blood Glucose Level is:  50.73315
>
```

Explanation

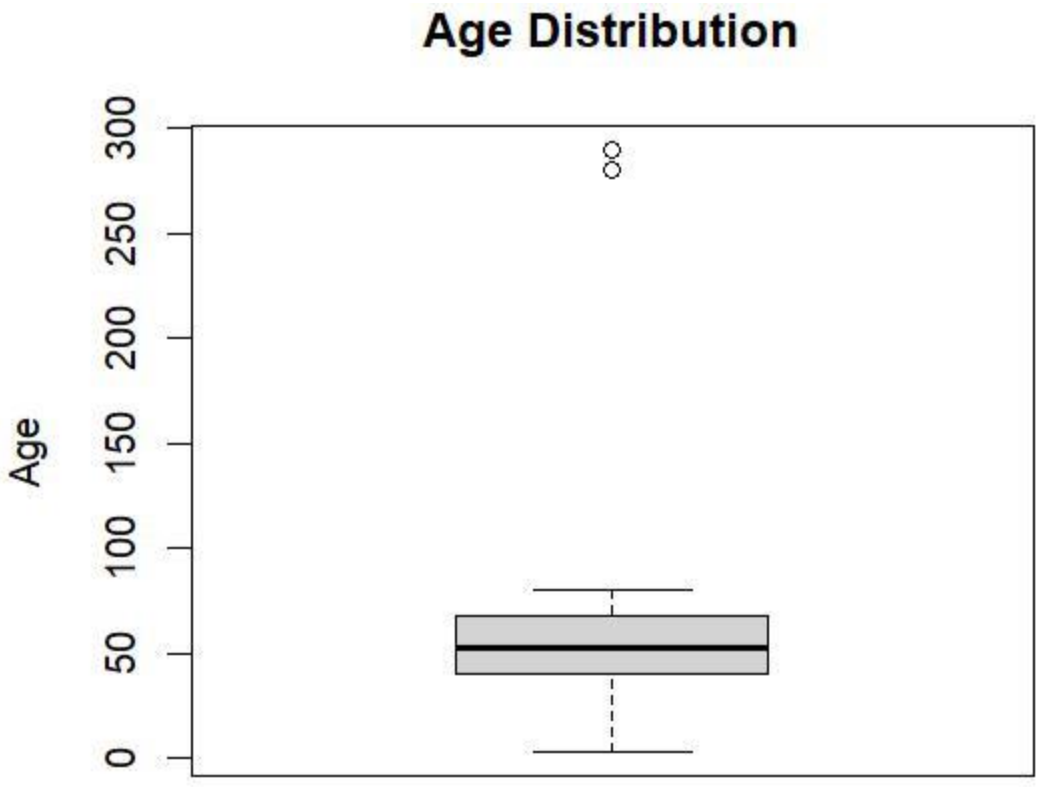
sd is the method used to calculate standard deviation. The blood_glucose_level column of the dataset was passed as an argument along with **na.rm = TRUE** which ignores the NA values. The result was stored in **bloodGlucoseLevelStandardDeviation** variable which was then printed by the **cat** method.

- **Box plot for Age**

Code

```
boxplot(dataset$age, main = "Age Distribution", ylab = "Age")
```

Output



Explanation

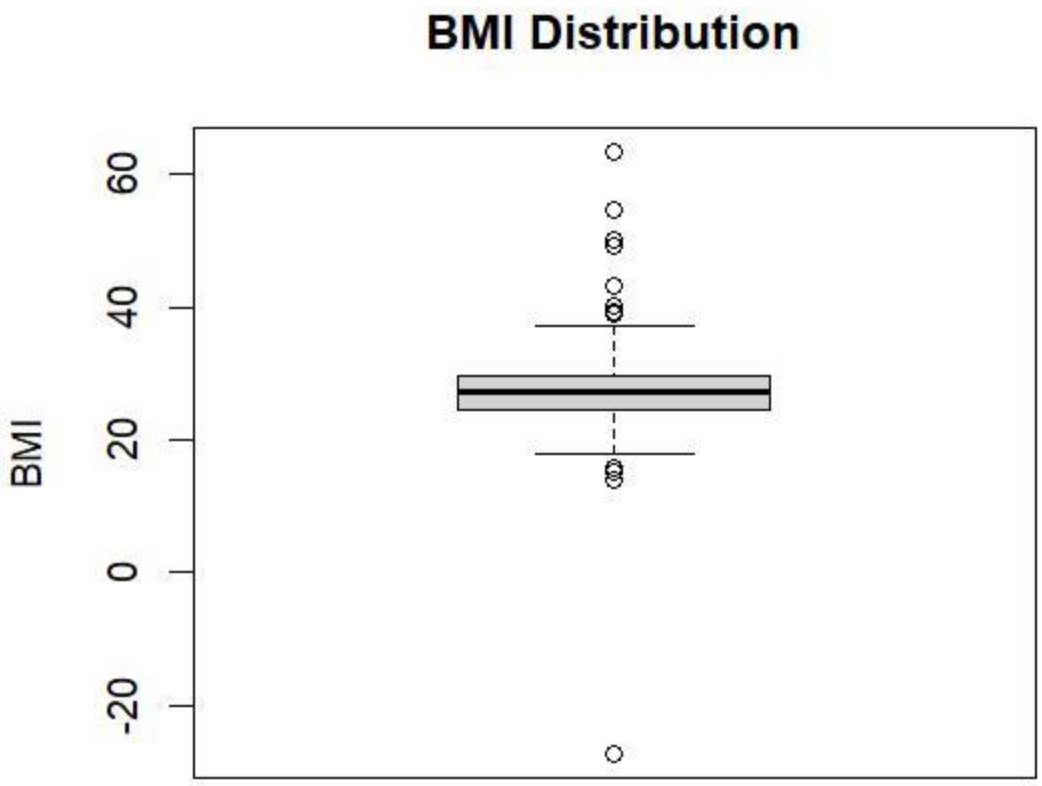
boxplot is the method which is used in order to create box plots. **dataset$age** was passed as an argument to create a box plot of the age column, **main** to give the title and **ylab** to provide a label for the y-axis.

- **Box plot for BMI**

Code

```
boxplot(dataset$bmi, main = "BMI Distribution", ylab = "BMI")
```

Output



Explanation

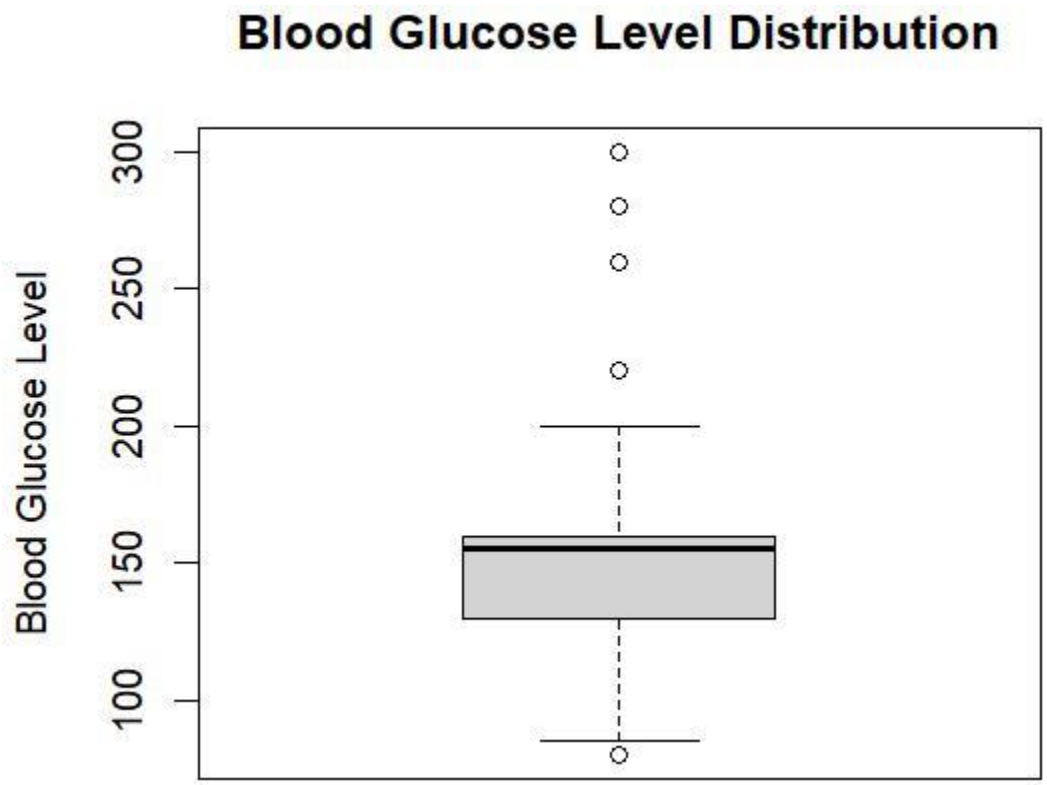
boxplot is the method which is used in order to create box plots. **dataset$bmi** was passed as an argument to create a box plot of the bmi column, **main** to give the title and **ylab** to provide a label for the y-axis.

- **Box plot for Blood Glucose Level**

Code

```
boxplot(dataset$blood_glucose_level, main = "Blood Glucose Level Distribution", ylab
= "Blood Glucose Level")
```

Output



Explanation

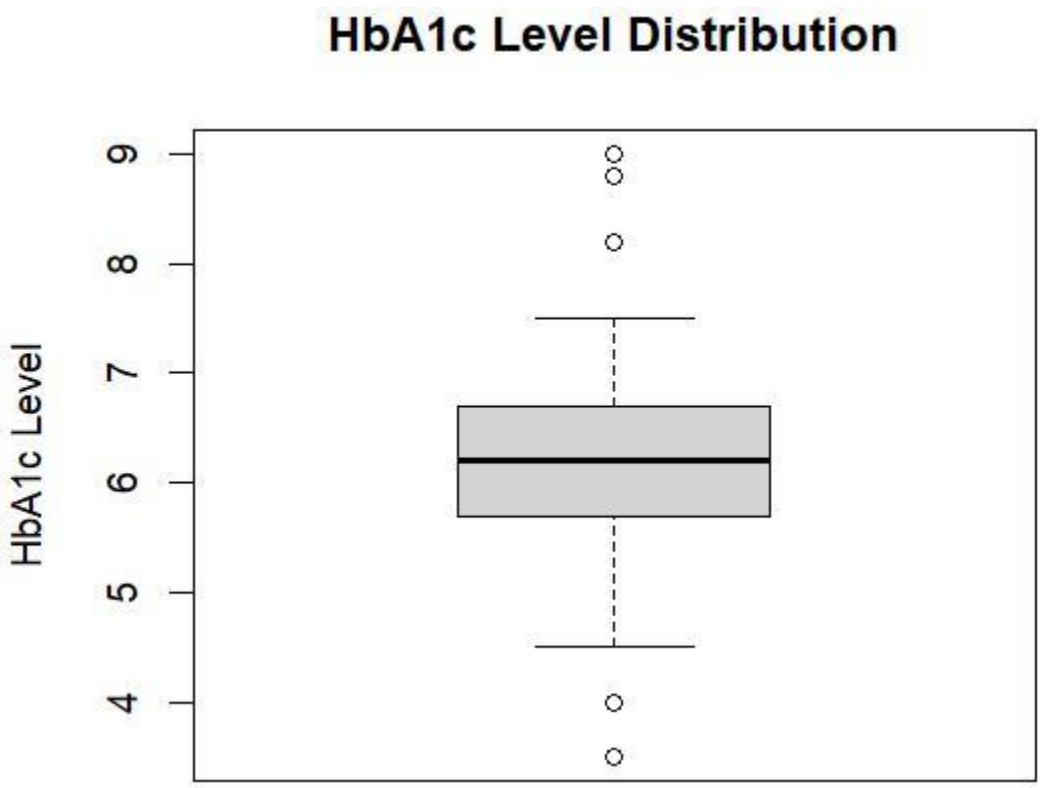
boxplot is the method which is used in order to create box plots. **dataset$blood_glucose_level** was passed as an argument to create a box plot of the blood_glucose_level column, **main** to give the title and **ylab** to provide a label for the y-axis.

- **Box plot for HbA1c Level**

Code

```
boxplot(dataset$HbA1c_level, main = "HbA1c Level Distribution", ylab = "HbA1c Level")
```

Output



Explanation

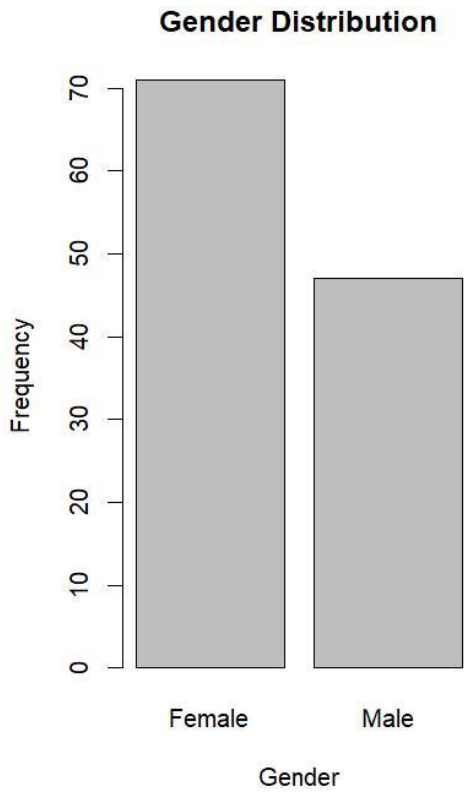
boxplot is the method which is used in order to create box plots. **dataset$HbA1c_level** was passed as an argument to create a box plot of the HbA1c_level column, **main** to give the title and **ylab** to provide a label for the y-axis.

- **Bar Plot for Gender**

Code

barplot(table(dataset$gender), main="Gender Distribution", xlab="Gender", ylab="Frequency")

Output



Explanation

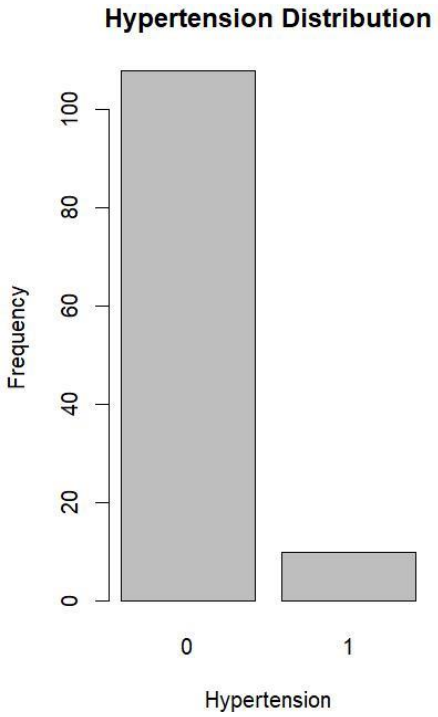
barplot is the method which is used in order to create bar plots. First, **table** method was used with an argument **dataset$gender** to tabulate the gender column with variable name and the frequency in the form of a table**.** The result was passed into the **barplot** method along with **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Bar Plot for Hypertension**

Code

```
barplot(table(dataset$hypertension), main="Hypertension Distribution",
xlab="Hypertension", ylab="Frequency")
```

Output



Explanation

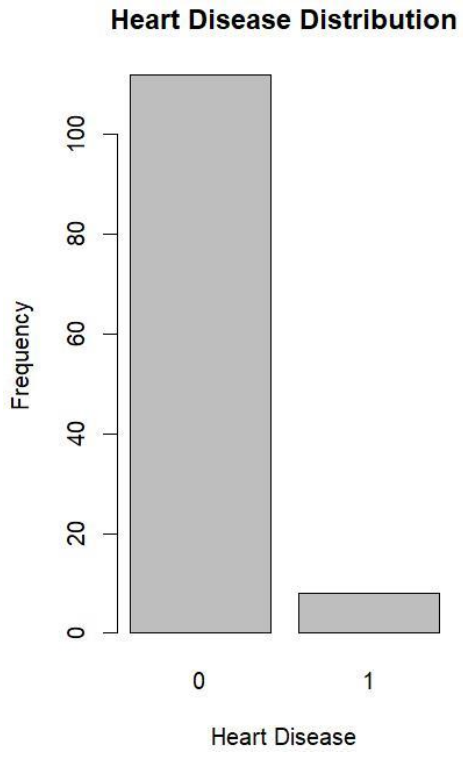
barplot is the method which is used in order to create bar plots. First, **table** method was used with an argument **dataset$hypertension** to tabulate the hypertension column with variable name and the frequency in the form of a table**.** The result was passed into the **barplot** method along with **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Bar Plot for Heart Disease**

Code

```
barplot(table(dataset$heart_disease), main="Heart Disease Distribution", xlab="Heart Disease", ylab="Frequency")
```

Output



Explanation

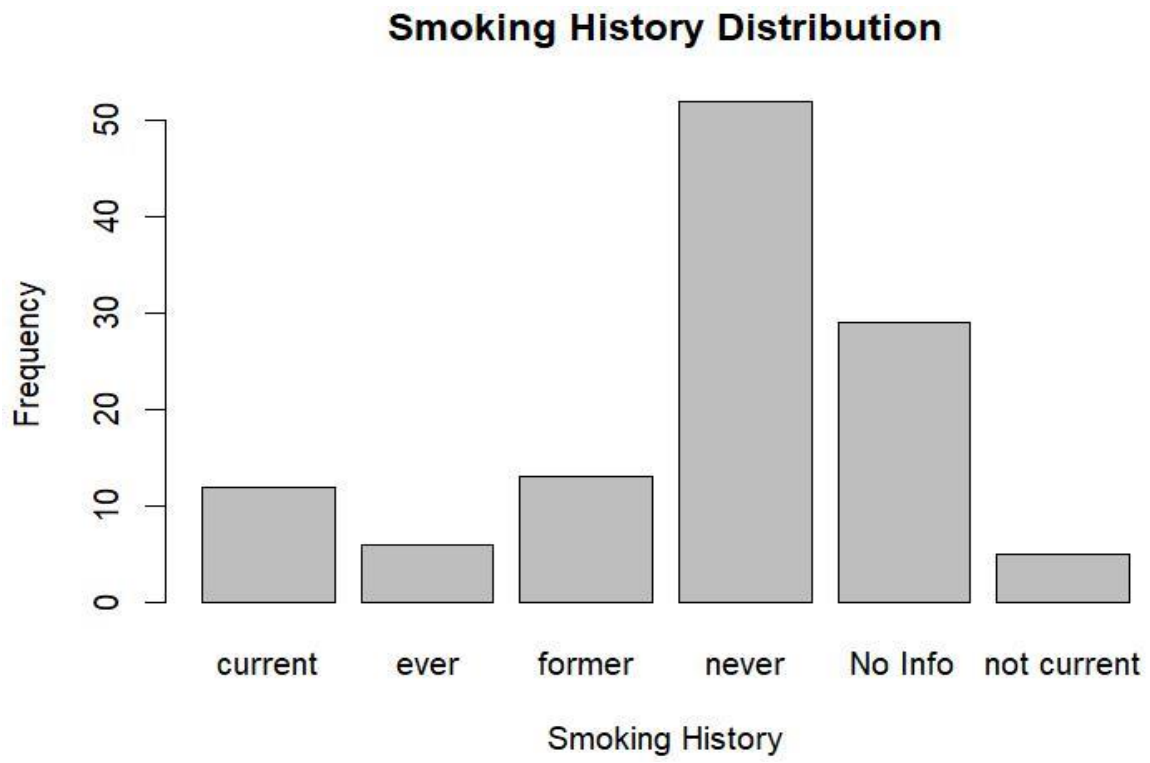
barplot is the method which is used in order to create bar plots. First, **table** method was used with an argument **dataset$heart_disease** to tabulate the heart_disease column with variable name and the frequency in the form of a table**.** The result was passed into the **barplot** method along with **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Bar Plot for Smoking History**

Code

```
barplot(table(dataset$smoking_history), main="Smoking History Distribution", xlab="Smoking History", ylab="Frequency")
```

Output



Explanation

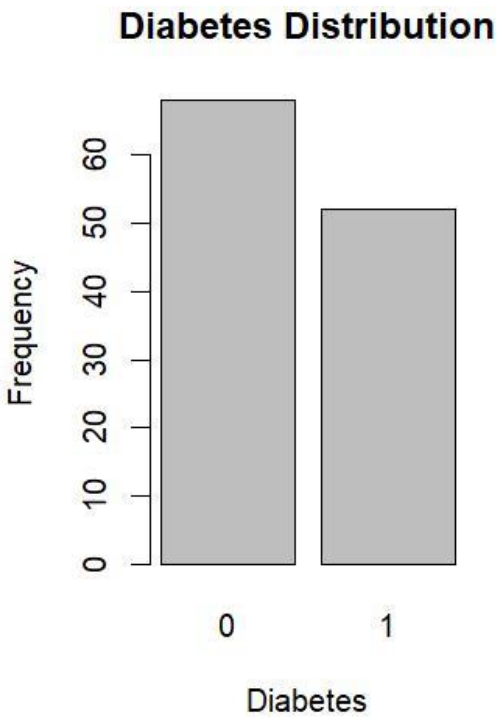
barplot is the method which is used in order to create bar plots. First, **table** method was used with an argument **dataset$smoking_history** to tabulate the smoking_history column with variable name and the frequency in the form of a table**.** The result was passed into the **barplot** method along with **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Bar Plot for Diabetes**

Code

```
barplot(table(dataset$diabetes), main="Diabetes Distribution", xlab="Diabetes",
ylab="Frequency")
```

Output



Explanation

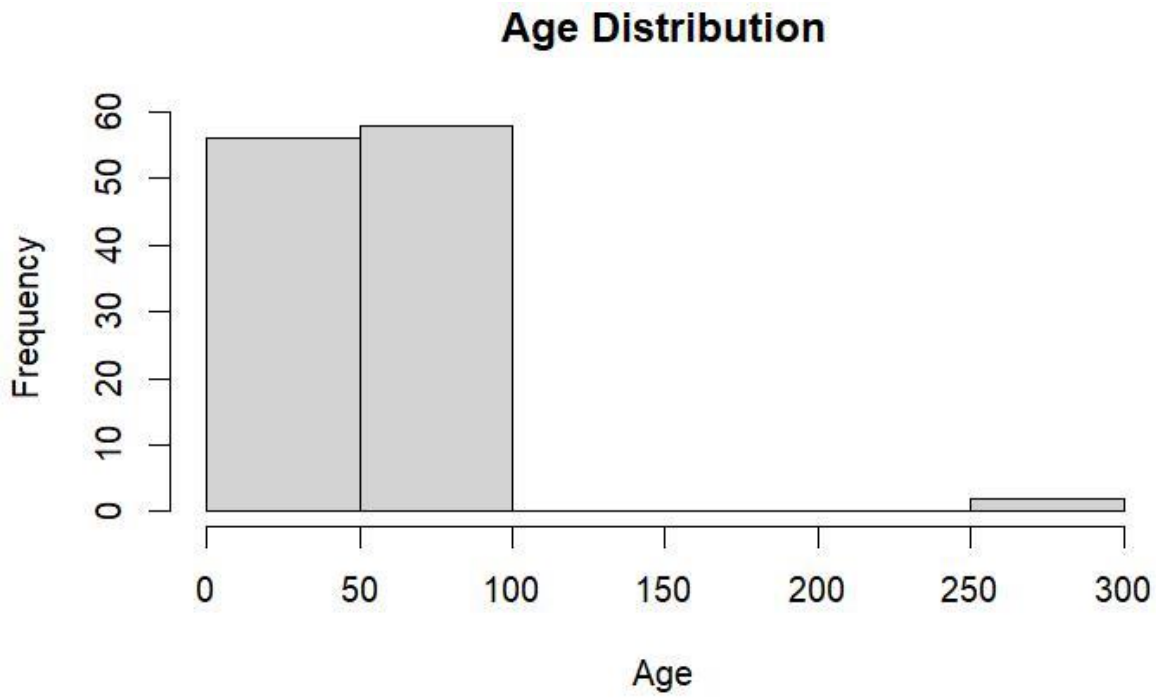
barplot is the method which is used in order to create bar plots. First, **table** method was used with an argument **dataset$diabetes** to tabulate the diabetes column with variable name and the frequency in the form of a table**.** The result was passed into the **barplot** method along with **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Histogram for Age**

Code

```
hist(dataset$age, main = "Age Distribution", xlab = "Age", ylab="Frequency")
```

Output



Explanation

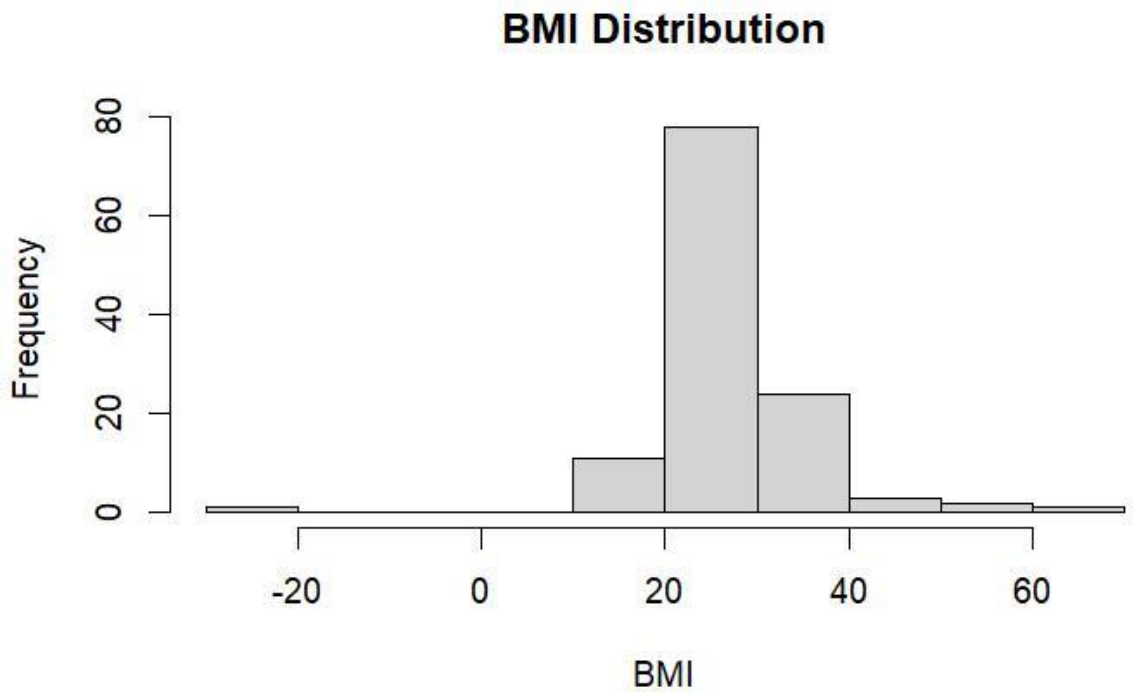
hist is the method which is used in order to create histograms. **dataset$age** was passed as an argument to create a histogram of the age column, **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Histogram for BMI**

Code

```
hist(dataset$bmi, main = "BMI Distribution", xlab = "BMI", ylab="Frequency")
```

Output



Explanation

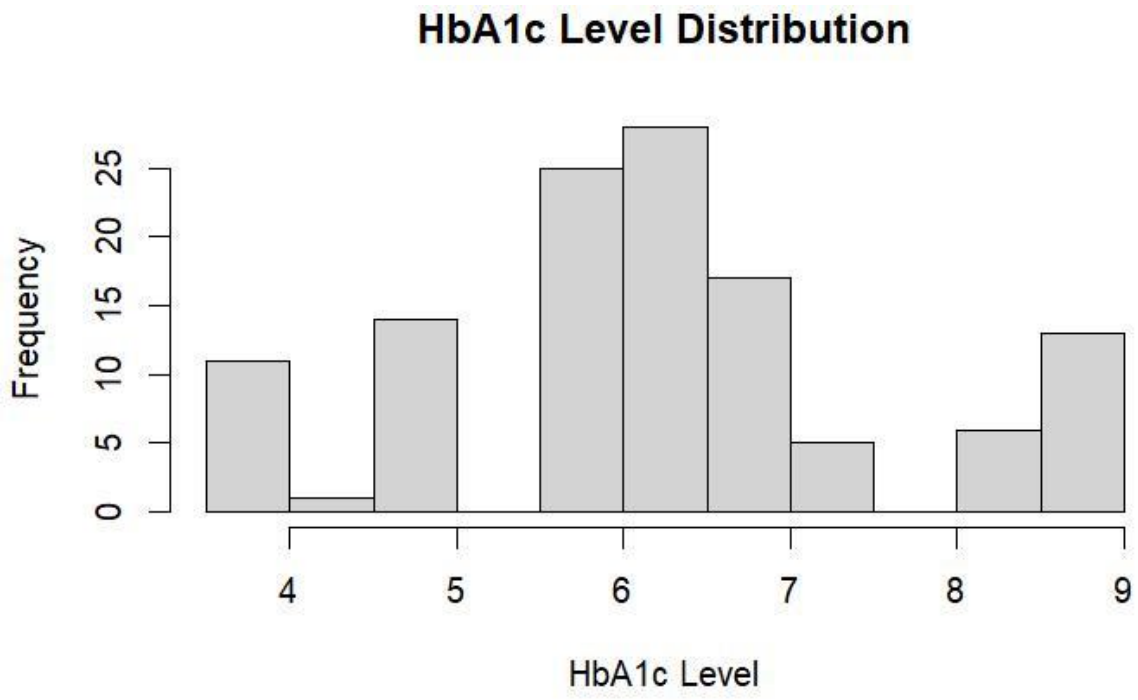
hist is the method which is used in order to create histograms. **dataset$bmi** was passed as an argument to create a histogram of the bmi column, **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Histogram for HbA1c Level**

Code

```
hist(dataset$HbA1c_level, main = "HbA1c Level Distribution", xlab = "HbA1c
Level", ylab="Frequency")
```

Output



Explanation

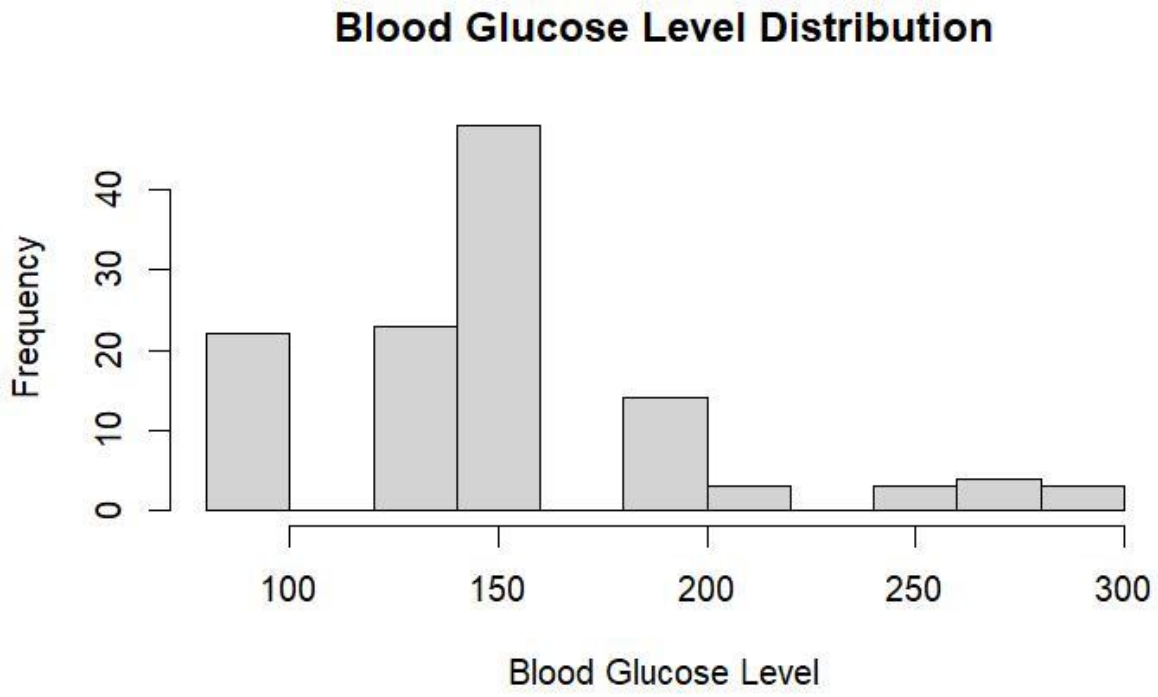
hist is the method which is used in order to create histograms. **dataset$HbA1c_level** was passed as an argument to create a histogram of the HbA1c_level column, **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Histogram for Blood Glucose Level**

Code

```
hist(dataset$blood_glucose_level, main = "Blood Glucose Level Distribution", xlab =
"Blood Glucose Level", ylab="Frequency")
```

Output



Explanation

hist is the method which is used in order to create histograms. **dataset$blood_glucose_level** was passed as an argument to create a histogram of the blood_glucose_level column, **main** to give the title, **xlab** to provide a label for the x-axis and **ylab** to provide a label for the y-axis.

- **Replacing Age outliers with mean value**

Code

```
ageBoxplot <- boxplot(dataset$age)
outliers <- ageBoxplot$out
cat("Outliers are", outliers)
ageMean <- mean(dataset$age, na.rm = TRUE)
outlierPositions <- match(outliers, dataset$age)
dataset$age[outlierPositions] <- as.integer (ageMean)
```

Output

```
> ageBoxplot <- boxplot(dataset$age)
> outliers <- ageBoxplot$out
> cat("Outliers are", outliers)
Outliers are 290 280
> ageMean <- mean(dataset$age, na.rm = TRUE)
> outlierPositions <- match(outliers, dataset$age)
> dataset$age[outlierPositions] <- as.integer (ageMean)
```

Explanation

age attribute was boxplotted from the dataset in order to find the outliers and saved its instance as **ageBoxplot**. Then, the outliers were extracted using **ageBoxplot$out**. After that**, cat** method was used in order to print the outliers. The mean value of the age column was stored in a variable **ageMean** using the **mean** method. **na.rm = TRUE** was passed as an argument so that the NA values are not in consideration while calculating mean. After that, the positions of the outliers were stored in **outlierPositions** using the **match** method which returns the outlier positions in the age column. Then the mean value was converted to integer and replaced all the outliers in the age column.

- **Replacing Age outliers with median value**

Code

```
ageBoxplot <- boxplot(dataset$age)
outliers <- ageBoxplot$out
cat("Outliers are", outliers)
ageMedian <- median(dataset$age, na.rm = TRUE)
outlierPositions <- match(outliers, dataset$age)
dataset$age[outlierPositions] <- as.integer (ageMedian)
```

Output

```
> ageBoxplot <- boxplot(dataset$age)
> outliers <- ageBoxplot$out
> cat("Outliers are", outliers)
Outliers are 290 280
> ageMedian <- median(dataset$age, na.rm = TRUE)
> outlierPositions <- match(outliers, dataset$age)
> dataset$age[outlierPositions] <- as.integer (ageMedian)
```

Explanation

age attribute was boxplotted from the dataset in order to find the outliers and saved its instance as **ageBoxplot**. Then, the outliers were extracted using **ageBoxplot$out**. After that**, cat** method was used in order to print the outliers. The mean value of the age column was stored in a variable **ageMedian** using the **median** method. **na.rm = TRUE** was passed as an argument so that the NA values are not in consideration while calculating mean. After that, the positions of the outliers were stored in **outlierPositions** using the **match** method which returns the outlier positions in the age column. Then the median value was converted to integer and replaced all the outliers in the age column.

- **Replacing Age outliers with mode value**

Code

```
ageBoxplot <- boxplot(dataset$age)
outliers <- ageBoxplot$out
cat("Outliers are", outliers)
ageMode <- strtoi(names(sort(table(dataset$age[!is.na(dataset$age)]), decreasing = TRUE)[1]))
outlierPositions <- match(outliers, dataset$age)
dataset$age[outlierPositions] <- as.integer (ageMode)
```

Output

```
> ageBoxplot <- boxplot(dataset$age)
> outliers <- ageBoxplot$out
> cat("Outliers are", outliers)
Outliers are 290 280
> ageMode <- strtoi(names(sort(table(dataset$age[!is.na(dataset$age)]), decreasing = TRUE)[1]))
> outlierPositions <- match(outliers, dataset$age)
> dataset$age[outlierPositions] <- as.integer (ageMode)
```

Explanation

age attribute was boxplotted from the dataset in order to find the outliers and saved its instance as **ageBoxplot**. Then, the outliers were extracted using **ageBoxplot$out**. After that**, cat** method was used in order to print the outliers. Then **table** method was used which returns a categorical representation of data with variable name and the frequency in the form of a table. **dataset$age[!is.na(dataset$age)]** was passed as an argument to tabulate only the age column where the values are not NA. After that the result was sorted using the **sort** method. **decreasing** = **TRUE** was used in order to sort the result in a decreasing order. Then the **names** method returned the name of index 1 and the result was stored in **ageMode.**. After that, the positions of the outliers were stored in **outlierPositions** using the **match** method which returns the outlier positions in the age column. Then the mode value was converted to integer and replaced all the outliers in the age column.