AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

INTRODUCTION TO DATA SCIENCE

Summer 2024-2025

Section: A

Group: 17

PROJECT REPORT ON

Applying Data Pre-processing on a Dataset

Supervised By

TOHEDUL ISLAM

Submitted By

Name	ID
1. MD. IMRAN AHMED	**20-43738-2**
2. BORTOMAN DAS	**21-44429-1**
3. MD. MIRAZUL HASAN	**22-46674-1**

Date of Submission: **July 14, 2024**

Introduction:

The dataset contains detailed information about caesarian section results for 80 pregnant women, highlighting significant characteristics of delivery problems in the medical field. Given the diverse data sources, the dataset may contain errors and inconsistencies, necessitating a thorough pre-processing stage to clean and transform the raw data into a format suitable for machine learning models. This pre-processing is crucial to ensure the data's accuracy and usability, enabling meaningful insights into the factors influencing caesarian sections and contributing to the improvement of maternal healthcare services in rural Bangladesh.

We classify delivery time to 0 = Timely, 1 = Premature and 2 = Latecomer. As for the blood pressure in three statuses of 0 = Low, 1 = Normal and 2 = High moods. Heart Problem is classified as 0 = apt and 1 = inept. And Caesarian as 0 = No and 1 = Yes.

Data Exploration:

Import the dataset:

The dataset is saved as the 'MPD_Sec-A.csv' file. To begin preprocessing data in R Studio, we have to import the file first. After importing the dataset, the data is stored in the 'mydata' variable,

```
> mydata <- read.csv("D:/MPD_Sec-A.csv", header = TRUE, sep =",")
```

	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	7	27		64.0	2	0	normal	0	0
8	8	32	male	70.0	3	0	normal	0	1
9	9	28	female	63.5	2	0		0	0
10	10	27	male	64.5	1	1	normal	0	1
11	11	36	male	75.0	1	0	normal	0	0
12	12	33	male	NA	1	1	low	0	1
13	12	33	male	70.0	1	1	low	0	1
14	13	23	female	58.0	1	1	normal	0	0
15	14	20	male	55.0	1	0	normal	1	0
16	15	29	male	65.0	1	NA		1	1

Figure 01: Imported main dataset.

Required Library:

For our data Pre-processing purpose, we will use '', 'tibble', 'dplyr' and 'ggplot2' library,

```
install.packages("dplyr")
install.packages("tibble")
install.packages("ggplot2")
library(dplyr)
library(tibble)
library(ggplot2)
```

Summary of the dataset:

```
> summary(mydata)
   Patient_id          Age            Gender            weight.kg.      Delivery_number
 Min.   : 1.00   Min.   : 18.00   Length:83          Min.   : 49.00   Length:83
 1st Qu.:20.50   1st Qu.: 25.50   Class :character   1st Qu.: 61.50   Class :character
 Median :40.00   Median : 28.00   Mode  :character   Median : 64.00   Mode  :character
 Mean   :40.08   Mean   : 32.27                      Mean   : 66.64
 3rd Qu.:59.50   3rd Qu.: 32.00                      3rd Qu.: 68.25
 Max.   :80.00   Max.   :135.00                      Max.   :140.00
                 NA's   :4                           NA's   :4
 Delivery_time       Blood              Heart          Caesarian
 Min.   :0.0000   Length:83          Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   Class :character   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Mode  :character   Median :0.0000   Median :1.0000
 Mean   :0.6375                      Mean   :0.3735   Mean   :0.6667
 3rd Qu.:1.0000                      3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :2.0000                      Max.   :1.0000   Max.   :1.0000
 NA's   :3                                            NA's   :2
```

In the summary of the main dataset, we can see the dataset has 9 attributes where some attributes have missing values.

To show the data types of the attributes in the dataset,

```
> str(mydata)
'data.frame':   83 obs. of  9 variables:
 $ Patient_id     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Age            : int  22 26 26 28 22 26 27 32 28 27 ...
 $ Gender         : chr  "female" "male" "male" "male" ...
 $ weight.kg.     : num  57.7 63 62 65 58 63 64 70 63.5 64.5 ...
 $ Delivery_number: chr  "1" "2" "2" "1" ...
 $ Delivery_time  : int  0 0 1 0 0 1 0 0 0 1 ...
 $ Blood          : chr  "high" "normal" "normal" "high" ...
 $ Heart          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Caesarian      : int  0 1 0 0 1 0 0 1 0 1 ...
```

We can see there are 83 instances in the dataset. The ‘Patient_id’, ‘Age’, ‘Delivery_time’, ‘Heart’ and ‘Caesarian’ attributes are integer type, the ‘weight.kg’. attribute is numerical type and the ‘Gender’, ‘Delivery_number’ and ‘Blood’ attributes are character type.

The dataset has 13 missing values distributed in the whole dataset,

```
> sum(is.na(mydata))
[1] 13
```

Project Solution Design:

The dataset shows there are missing values in a couple attributes, there may be some invalid values. These values must be recovered properly. Moreover, if there are any outliers in particular attribute, we must recover them using proper method. Finally, we also need to convert attributes to their proper data type. For example, we need to convert the ‘Delivery_time’, ‘Heart’ and ‘Caesarian’ attribute to character datatype. For recovering missing values, invalid values, or outliers we may use average (mean) or most frequent value (mode) depending on the datatype of the attribute.

Data Pre-processing:

Patient_id Attribute:

The ‘Patient_id’ column is an integer type attribute.

```
> class(mydata$Patient_id)
[1] "integer"
```

There are no missing values in the ‘Patient_id’ column.

```
> which(is.na(mydata$Patient_id))
integer(0)
```

There are 3 duplicated ids in the ‘Patient_id’ column,

```
> mydata[duplicated(mydata$Patient_id), ]
   Patient_id Age Gender weight.kg. Delivery_number Delivery_time Blood Heart Caesarian
13         12  33   male         70               1             1   low     0         1
24         22  33   male         75               2             0   low     1         1
56         53  30   male         68               3             2  high     0         1
```

Remove the duplicate values from the ‘Patient_id’ column,

```
> mydata <- distinct(mydata, Patient_id, .keep_all = TRUE)
```

	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	7	27		64.0	2	0	normal	0	0
8	8	32	male	70.0	3	0	normal	0	1
9	9	28	female	63.5	2	0		0	0
10	10	27	male	64.5	1	1	normal	0	1
11	11	36	male	75.0	1	0	normal	0	0
12	12	33	male	NA	1	1	low	0	1
13	13	23	female	58.0	1	1	normal	0	0

Figure 02: Dataset after removing duplicate values from ‘Patient_id’ column.

The column name is in the wrong format,

```
> names(mydata)[names(mydata) == "Patient_id"] <- "Patient ID"
```

	Patient ID	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0

Figure 03: Dataset after renaming the ‘Patient_id’ column to ‘Patient ID’.

Age Attribute:

The ‘Age’ column is an integer type attribute.

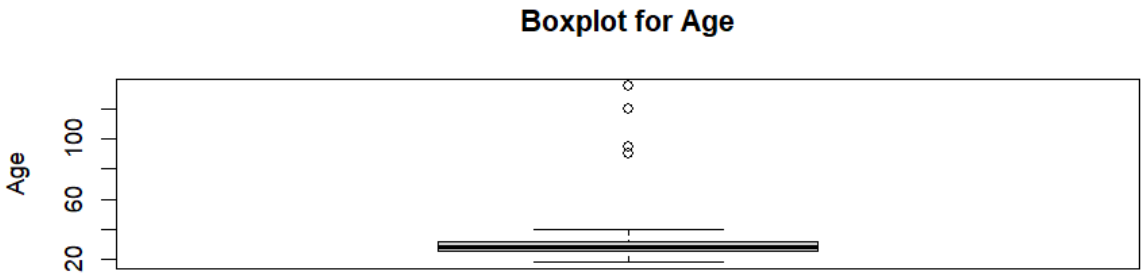
```
> class(mydata$Age)
[1] "integer"
```

There are 4 missing values in the ‘Age’ column,

```
> which(is.na(mydata$Age))
[1] 50 62 66 78
```

Check for outliers in the ‘Age’ column using boxplot,

```
> boxplot(mydata$Age, main = "Boxplot for Age", ylab = "Age")
```



There are 4 outliers in the ‘Age’ column,

```
> boxplot.stats(mydata$Age)$out
[1]  95  90 135 120
```

Before recovering missing values, we need to remove outliers. Because we are going to use the average (mean) value to recover the missing values. If we didn’t remove the outliers the average (mean) values would not be accurate.

To remove the outliers, we will use Interquartile Range (IQR) method,

```
> Q1 <- quantile(mydata$Age, 0.25, na.rm = TRUE)
> Q3 <- quantile(mydata$Age, 0.75, na.rm = TRUE)
> IQR <- Q3 - Q1
> lower_bound <- Q1 - 1.5 * IQR
> upper_bound <- Q3 + 1.5 * IQR
> mydata <- mydata %>% filter(is.na(Age) | (Age >= lower_bound & Age <= upper_bound))
```

The outliers are removed, now we can apply the average (mean) values to recover the missing values.

```
> mean_Age <- mean(mydata$Age, na.rm = TRUE)
> mydata$Age[is.na(mydata$Age)] <- round(mean_Age)
```

	Patient ID	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	7	27		64.0	2	0	normal	0	0
8	8	32	male	70.0	3	0	normal	0	1
9	9	28	female	63.5	2	0		0	0
10	10	27	male	64.5	1	1	normal	0	1
11	11	36	male	75.0	1	0	normal	0	0
12	12	33	male	*NA*	1	1	low	0	1
13	13	23	female	58.0	1	1	normal	0	0

Figure 04: Dataset after removing outliers & recovering missing values from 'Age' column.

Gender Attribute:

The 'Gender' column is a character type attribute,

```
> class(mydata$Gender)
[1] "character"
```

Check what values are there in the 'Gender' column,

```
> unique(mydata$Gender)
[1] "female"   "male"     ""         "mmale"    "Male-ish" "maile"
```

Here, the valid values are male and female but the value 'mmale', 'Male-ish', 'maile' and missing values are invalid. As the 'Gender' column is categorical, we will convert it in numerical values to recover any missing values or invalid values.

Replace values of 'Gender' column from with 1 and 2,

```
> mydata$Gender <- factor(mydata$Gender, levels = c("male", "female"), labels = c(1,2))
```

	Patient ID	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	2	57.7	1	0	high	0	0
2	2	26	1	63.0	2	0	normal	0	1
3	3	26	1	62.0	2	1	normal	0	0
4	4	28	1	65.0	1	0	high	0	0
5	5	22	1	58.0	2	0	normal	0	1
6	6	26	1	63.0	1	1	low	0	0
7	7	27	*NA*	64.0	2	0	normal	0	0
8	8	32	1	70.0	3	0	normal	0	1
9	9	28	2	63.5	2	0		0	0
10	10	27	1	64.5	1	1	normal	0	1
11	11	36	1	75.0	1	0	normal	0	0
12	12	33	1	*NA*	1	1	low	0	1

Figure 05: Dataset after replacing values of 'Gender' column.

There are 6 missing values in the ‘Gender’ column,

```
> sum(is.na(mydata$Gender))
[1] 6
```

The ‘Gender’ column is categorical, recover these missing values using the most frequent (mode) value,

```
> most_frequent_ge <- names(sort(table(mydata$Gender), decreasing = TRUE))[1]
> mydata$Gender[is.na(mydata$Gender)] <- most_frequent_ge
```

	Patient ID	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	2	57.7	1	0	high	0	0
2	2	26	1	63.0	2	0	normal	0	1
3	3	26	1	62.0	2	1	normal	0	0
4	4	28	1	65.0	1	0	high	0	0
5	5	22	1	58.0	2	0	normal	0	1
6	6	26	1	63.0	1	1	low	0	0
7	7	27	1	64.0	2	0	normal	0	0
8	8	32	1	70.0	3	0	normal	0	1
9	9	28	2	63.5	2	0		0	0
10	10	27	1	64.5	1	1	normal	0	1
11	11	36	1	75.0	1	0	normal	0	0
12	12	33	1	NA	1	1	low	0	1
13	13	23	2	58.0	1	1	normal	0	0
14	14	20	1	55.0	1	0	normal	1	0

Figure 06: Dataset after recovering missing values using the most frequent value in the ‘Gender’ column.

Replace the ‘Gender’ attribute’s values back to ‘male’ and ‘female’.

```
> mydata$Gender <- factor(mydata$Gender, levels = c(1,2), labels = c("male", "female"))
```

	Patient ID	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	7	27	male	64.0	2	0	normal	0	0
8	8	32	male	70.0	3	0	normal	0	1
9	9	28	female	63.5	2	0		0	0
10	10	27	male	64.5	1	1	normal	0	1
11	11	36	male	75.0	1	0	normal	0	0
12	12	33	male	NA	1	1	low	0	1
13	13	23	female	58.0	1	1	normal	0	0
14	14	20	male	55.0	1	0	normal	1	0

Figure 07: Dataset after replacing the ‘Gender’ column values.

weight.kg. Attribute:

The ‘weight.kg.’ column is a numeric type of attribute,

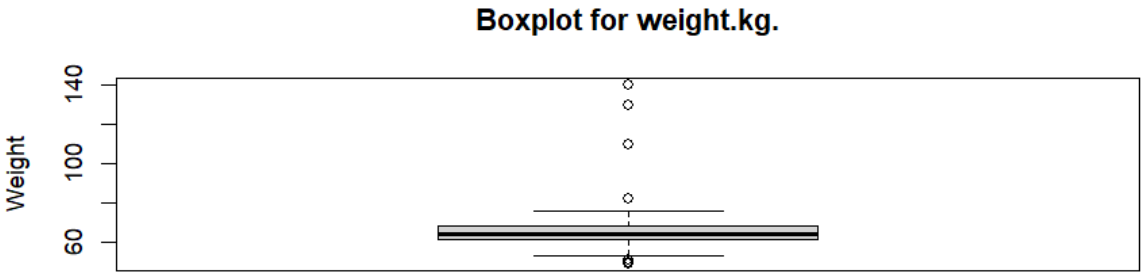
```
> class(mydata$weight.kg.)
[1] "numeric"
```

There are 4 missing values in the ‘weight.kg.’ column,

```
> which(is.na(mydata$weight.kg.))
[1] 12 49 52 65
```

Check for outliers in the ‘weight.kg.’ column using boxplot,

```
> boxplot(mydata$weight.kg., main = "Boxplot for weight.kg.", ylab = "Weight")
```



There are 7 outliers in the ‘weight.kg.’ column,

```
> boxplot.stats(mydata$weight.kg.)$out
[1]  49  50  82  51 110 130 140
```

Before recovering missing values, we need to remove outliers. Because we are going to use the average (mean) value to recover the missing values. If we didn’t remove the outliers the average (mean) values would not be accurate.

To remove the outliers, we will use Interquartile Range (IQR) method,

```
> Q1 <- quantile(mydata$weight.kg., 0.25, na.rm = TRUE)
> Q3 <- quantile(mydata$weight.kg., 0.75, na.rm = TRUE)
> IQR <- Q3 - Q1
> lower_bound <- Q1 - 1.5 * IQR
> upper_bound <- Q3 + 1.5 * IQR
> mydata <- mydata %>% filter(is.na(weight.kg.) | (weight.kg. >= lower_bound & weight.kg. <= upper_bound))
```

The outliers are removed, now we can apply the average (mean) values to recover the missing values.

```
> mean_weight.kg. <- mean(mydata$weight.kg., na.rm = TRUE)
> mydata$weight.kg.[is.na(mydata$weight.kg.)] <- round(mean_weight.kg.)
```

	Patient ID	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	7	27	male	64.0	2	0	normal	0	0
8	8	32	male	70.0	3	0	normal	0	1
9	9	28	female	63.5	2	0		0	0
10	10	27	male	64.5	1	1	normal	0	1
11	11	36	male	75.0	1	0	normal	0	0
12	12	33	male	64.0	1	1	low	0	1
13	13	23	female	58.0	1	1	normal	0	0
14	14	20	male	55.0	1	0	normal	1	0
15	15	29	male	65.0	1	NA		1	1

Figure 08: Dataset after removing outliers & recovering missing values from 'weight.kg.' column.

The column name is in the wrong format,

```
> names(mydata)[names(mydata) == "weight.kg."] <- "Weight (Kg)"
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0

Figure 09: Dataset after renaming the 'weight.kg.' column to 'Weight (Kg)'.

Delivery_number Attribute:

The 'Delivery_number' column is a character type of attribute,

```
> class(mydata$Delivery_number)
[1] "character"
```

Check what values are there in the 'Delivery_number' column,

```
> unique(mydata$Delivery_number)
[1] "1"  "2"  "3"  ""   "1y" "4"
```

Here, the valid values are 1, 2, 3 and 4 but the value '1y' and missing values are invalid. As the 'Delivery_number' column is character type, we will convert it in numerical values to recover any missing values or invalid values.

```
> mydata$Delivery_number <- as.integer(mydata$Delivery_number)
Warning message:
NAs introduced by coercion
```

After converting the ‘Delivery_number’ column from character to integer type, all the invalid values are converted to missing values (NAs).

There are 2 missing values in the ‘Delivery_number’ column,

```
> which(is.na(mydata$Delivery_number))
[1] 24 36
```

Recover missing values using most frequent (mode) values,

```
> most_frequent_dn <- names(sort(table(mydata$Delivery_number), decreasing = TRUE))[1]
> mydata$Delivery_number[is.na(mydata$Delivery_number)] <- most_frequent_dn
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	7	27	male	64.0	2	0	normal	0	0
8	8	32	male	70.0	3	0	normal	0	1
9	9	28	female	63.5	2	0		0	0
10	10	27	male	64.5	1	1	normal	0	1
11	11	36	male	75.0	1	0	normal	0	0
12	12	33	male	64.0	1	1	low	0	1

Figure 10: Dataset after recovering missing values from ‘Delivery_number’ column.

The column name is in the wrong format,

```
> names(mydata)[names(mydata) == "Delivery_number"] <- "Delivery Number"
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0

Figure 11: Dataset after renaming the ‘Delivery_number’ column to ‘Delivery Number’.

Delivery_time Attribute:

The ‘Delivery_time’ column is an integer type of attribute,

```
> class(mydata$Delivery_time)
[1] "integer"
```

Let’s check what values are there in the ‘Delivery_time’ column,

```
> unique(mydata$Delivery_time)
[1]  0  1 NA  2
```

Here, the valid values are 0, 1 and 2 but the missing values are invalid. As the 'Delivery_time' column is integer type, we will recover any missing values and convert it to character type attribute.

There are 2 missing values in the 'Delivery_time' column,

```
> which(is.na(mydata$Delivery_time))
[1] 15 24
```

Recover missing values using most frequent (mode) values,

```
> most_frequent_dt <- names(sort(table(mydata$Delivery_time), decreasing = TRUE))[1]
> mydata$Delivery_time[is.na(mydata$Delivery_time)] <- most_frequent_dt
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	7	27	male	64.0	2	0	normal	0	0
8	8	32	male	70.0	3	0	normal	0	1
9	9	28	female	63.5	2	0		0	0
10	10	27	male	64.5	1	1	normal	0	1
11	11	36	male	75.0	1	0	normal	0	0

Figure 12: Dataset after recovering missing values from 'Delivery_time' column.

We need to convert the 'Delivery_time' column to character type attribute,

```
> mydata$Delivery_time <- as.character(mydata$Delivery_time)
```

We will replace the values of 'Delivery_time' column as like the description of the dataset (0 = timely, 1 = premature, 2 = latecomer),

```
> mydata$Delivery_time <- factor(mydata$Delivery_time, levels = c(0,1,2), labels = c("timely",
"premature", "latecomer"))
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	timely	high	0	0
2	2	26	male	63.0	2	timely	normal	0	1
3	3	26	male	62.0	2	premature	normal	0	0
4	4	28	male	65.0	1	timely	high	0	0
5	5	22	male	58.0	2	timely	normal	0	1
6	6	26	male	63.0	1	premature	low	0	0
7	7	27	male	64.0	2	timely	normal	0	0
8	8	32	male	70.0	3	timely	normal	0	1
9	9	28	female	63.5	2	timely		0	0
10	10	27	male	64.5	1	premature	normal	0	1
11	11	36	male	75.0	1	timely	normal	0	0

Figure 13: Dataset after replacing values 'Delivery_time' column.

The column name is in the wrong format,

```
> mydata$Delivery_time <- as.character(mydata$Delivery_time)
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	timely	high	0	0
2	2	26	male	63.0	2	timely	normal	0	1
3	3	26	male	62.0	2	premature	normal	0	0
4	4	28	male	65.0	1	timely	high	0	0

Figure 14: Dataset after renaming the 'Delivery_time' column to 'Delivery Time'.

Blood Attribute:

The 'Blood' column is a character type of attribute,

```
> class(mydata$Blood)
[1] "character"
```

Check what values are there in the 'Blood' column,

```
> unique(mydata$Blood)
[1] "high"   "normal" "low"    ""
```

Here, the valid values are high, normal and low but the missing values are invalid. As the 'Blood' column is character type, we will convert it in numerical values to recover any missing values or invalid values,

```
> mydata$Blood <- factor(mydata$Blood, levels = c("low", "normal", "high"), labels = c(0, 1, 2))
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	timely	2	0	0
2	2	26	male	63.0	2	timely	1	0	1
3	3	26	male	62.0	2	premature	1	0	0
4	4	28	male	65.0	1	timely	2	0	0
5	5	22	male	58.0	2	timely	1	0	1
6	6	26	male	63.0	1	premature	0	0	0
7	7	27	male	64.0	2	timely	1	0	0
8	8	32	male	70.0	3	timely	1	0	1
9	9	28	female	63.5	2	timely	*NA*	0	0
10	10	27	male	64.5	1	premature	1	0	1
11	11	36	male	75.0	1	timely	1	0	0
12	12	33	male	64.0	1	premature	0	0	1
13	13	23	female	58.0	1	premature	1	0	0
14	14	20	male	55.0	1	timely	1	1	0

Figure 15: Dataset after replacing values of 'Blood' column.

There are 2 missing values in the 'Blood' column,

```
> which(is.na(mydata$Blood))
[1]  9 15 66
```

Recover missing values using most frequent (mode) values,

```
> most_frequent_bl <- names(sort(table(mydata$Blood), decreasing = TRUE))[1]
> mydata$Blood[is.na(mydata$Blood)] <- most_frequent_bl
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	timely	2	0	0
2	2	26	male	63.0	2	timely	1	0	1
3	3	26	male	62.0	2	premature	1	0	0
4	4	28	male	65.0	1	timely	2	0	0
5	5	22	male	58.0	2	timely	1	0	1
6	6	26	male	63.0	1	premature	0	0	0
7	7	27	male	64.0	2	timely	1	0	0
8	8	32	male	70.0	3	timely	1	0	1
9	9	28	female	63.5	2	timely	1	0	0
10	10	27	male	64.5	1	premature	1	0	1
11	11	36	male	75.0	1	timely	1	0	0
12	12	33	male	64.0	1	premature	0	0	1
13	13	23	female	58.0	1	premature	1	0	0
14	14	20	male	55.0	1	timely	1	1	0
15	15	29	male	65.0	1	timely	1	1	1

Figure 16: Dataset after recovering missing values from the 'Blood' column.

Replace the 'Blood' attribute's values back to the categorical value's 'low', 'normal' and 'high',

```
> mydata$Blood <- factor(mydata$Blood, levels = c(0, 1, 2), labels = c("low", "normal", "high"))
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	timely	high	0	0
2	2	26	male	63.0	2	timely	normal	0	1
3	3	26	male	62.0	2	premature	normal	0	0
4	4	28	male	65.0	1	timely	high	0	0
5	5	22	male	58.0	2	timely	normal	0	1
6	6	26	male	63.0	1	premature	low	0	0
7	7	27	male	64.0	2	timely	normal	0	0
8	8	32	male	70.0	3	timely	normal	0	1
9	9	28	female	63.5	2	timely	normal	0	0
10	10	27	male	64.5	1	premature	normal	0	1
11	11	36	male	75.0	1	timely	normal	0	0
12	12	33	male	64.0	1	premature	low	0	1
13	13	23	female	58.0	1	premature	normal	0	0
14	14	20	male	55.0	1	timely	normal	1	0
15	15	29	male	65.0	1	timely	normal	1	1

Figure 17: Dataset after replacing the 'Blood' column values.

The column name is in the wrong format,

```
> names(mydata)[names(mydata) == "Blood"] <- "Blood of Pressure"
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood of Pressure	Heart	Caesarian
1	1	22	female	57.7	1	timely	high	0	0
2	2	26	male	63.0	2	timely	normal	0	1
3	3	26	male	62.0	2	premature	normal	0	0
4	4	28	male	65.0	1	timely	high	0	0

Figure 18: Dataset after renaming the 'Blood' column to 'Blood of Pressure'.

Blood Attribute:

The 'Heart' column is an integer type of attribute,

```
> class(mydata$Heart)
[1] "integer"
```

The 'Heart' column has 0 and 1 as values and no missing or invalid values,

```
> unique(mydata$Heart)
[1] 0 1
```

Convert the 'Heart' attribute's values to the categorical value's,

```
> mydata$Heart <- as.character(mydata$Heart)
```

Replace the 'Heart' attribute's values to 'apt' and 'inept',

```
> mydata$Heart <- factor(mydata$Heart, levels = c(0, 1), labels = c("apt", "inept"))
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood of Pressure	Heart	Caesarian
1	1	22	female	57.7	1	timely	high	apt	0
2	2	26	male	63.0	2	timely	normal	apt	1
3	3	26	male	62.0	2	premature	normal	apt	0
4	4	28	male	65.0	1	timely	high	apt	0
5	5	22	male	58.0	2	timely	normal	apt	1
6	6	26	male	63.0	1	premature	low	apt	0
7	7	27	male	64.0	2	timely	normal	apt	0
8	8	32	male	70.0	3	timely	normal	apt	1
9	9	28	female	63.5	2	timely	normal	apt	0
10	10	27	male	64.5	1	premature	normal	apt	1
11	11	36	male	75.0	1	timely	normal	apt	0
12	12	33	male	64.0	1	premature	low	apt	1

Figure 19: Dataset after replacing the 'Heart' column values.

The column name is in the wrong format.

```
> names(mydata)[names(mydata) == "Heart"] <- "Heart Problem"
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood of Pressure	Heart Problem	Caesarian
1	1	22	female	57.7	1	timely	high	apt	0
2	2	26	male	63.0	2	timely	normal	apt	1
3	3	26	male	62.0	2	premature	normal	apt	0
4	4	28	male	65.0	1	timely	high	apt	0

Figure 20: Dataset after renaming the 'Heart' column to 'Heart Problem'.

Caesarian Attribute:

The ‘Caesarian’ column is an integer type of attribute,

```
> class(mydata$Caesarian)
[1] "integer"
```

The ‘Caesarian’ column has 0 and 1 as values and some missing or invalid values,

```
> unique(mydata$Caesarian)
[1]  0  1 NA
```

There are 2 missing values,

```
> which(is.na(mydata$Caesarian))
[1] 55 70
```

Recover them using most frequent (mode) values,

```
> most_frequent_ca <- names(sort(table(mydata$Caesarian), decreasing = TRUE))[1]
> mydata$Caesarian[is.na(mydata$Caesarian)] <- most_frequent_ca
```

Convert the ‘Caesarian’ attribute’s values to the categorical value’s,

```
> mydata$Caesarian <- as.character(mydata$Caesarian)
```

Replace the ‘Caesarian’ attribute’s values to ‘No’ and ‘Yes’,

```
> mydata$Caesarian <- factor(mydata$Caesarian, levels = c(0, 1), labels = c("No", "Yes"))
```

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood of Pressure	Heart Problem	Caesarian
1	1	22	female	57.7	1	timely	high	apt	No
2	2	26	male	63.0	2	timely	normal	apt	Yes
3	3	26	male	62.0	2	premature	normal	apt	No
4	4	28	male	65.0	1	timely	high	apt	No
5	5	22	male	58.0	2	timely	normal	apt	Yes
6	6	26	male	63.0	1	premature	low	apt	No
7	7	27	male	64.0	2	timely	normal	apt	No
8	8	32	male	70.0	3	timely	normal	apt	Yes
9	9	28	female	63.5	2	timely	normal	apt	No
10	10	27	male	64.5	1	premature	normal	apt	Yes
11	11	36	male	75.0	1	timely	normal	apt	No
12	12	33	male	64.0	1	premature	low	apt	Yes
13	13	23	female	58.0	1	premature	normal	apt	No
14	14	20	male	55.0	1	timely	normal	inept	No
15	15	29	male	65.0	1	timely	normal	inept	Yes
16	16	25	female	61.5	1	latecomer	low	apt	No

Figure 21: Dataset after replacing the ‘Caesarian’ column values.

After applying necessary pre-processing methods, the dataset looks like this,

	Patient ID	Age	Gender	Weight (Kg)	Delivery Number	Delivery Time	Blood of Pressure	Heart Problem	Caesarian
1	1	22	female	57.7	1	timely	high	apt	No
2	2	26	male	63.0	2	timely	normal	apt	Yes
3	3	26	male	62.0	2	premature	normal	apt	No
4	4	28	male	65.0	1	timely	high	apt	No
5	5	22	male	58.0	2	timely	normal	apt	Yes
6	6	26	male	63.0	1	premature	low	apt	No
7	7	27	male	64.0	2	timely	normal	apt	No
8	8	32	male	70.0	3	timely	normal	apt	Yes
9	9	28	female	63.5	2	timely	normal	apt	No
10	10	27	male	64.5	1	premature	normal	apt	Yes
11	11	36	male	75.0	1	timely	normal	apt	No
12	12	33	male	64.0	1	premature	low	apt	Yes
13	13	23	female	58.0	1	premature	normal	apt	No
14	14	20	male	55.0	1	timely	normal	inept	No
15	15	29	male	65.0	1	timely	normal	inept	Yes
16	16	25	female	61.5	1	latecomer	low	apt	No
17	17	25	male	61.5	1	timely	normal	apt	No
18	18	20	male	55.5	1	latecomer	high	apt	Yes
19	19	37	male	76.0	3	timely	normal	inept	Yes
20	20	24	male	56.6	1	latecomer	low	inept	Yes
21	21	26	male	62.0	1	premature	normal	apt	No
22	22	33	male	75.0	2	timely	low	inept	Yes
23	23	25	male	62.0	1	premature	high	apt	No
24	24	27	male	65.0	1	timely	low	inept	Yes
25	25	20	male	55.0	1	timely	high	inept	Yes
26	28	30	female	68.0	1	timely	normal	apt	No
27	29	32	male	73.0	1	timely	high	inept	Yes
28	30	26	male	62.5	2	premature	normal	inept	No
29	31	25	male	58.0	1	timely	low	apt	Yes

Figure 22: Dataset after pre-processing.

From the summary we can see that all the attributes are in correct format and there are no missing or invalid values.

```
> summary(mydata)
   Patient ID          Age           Gender    Weight (Kg)    Delivery Number      Delivery Time
 Min.   : 1.00   Min.   :20.00   male  :61   Min.   :53.00   Length:71          timely   :41
 1st Qu.:18.50   1st Qu.:26.00   female:10   1st Qu.:61.75   Class :character   premature:16
 Median :39.00   Median :28.00               Median :64.00   Mode  :character   latecomer:14
 Mean   :38.76   Mean   :28.24               Mean   :64.43
 3rd Qu.:58.00   3rd Qu.:32.00               3rd Qu.:67.50
 Max.   :78.00   Max.   :38.00               Max.   :76.00
 Blood of Pressure Heart Problem Caesarian
 low   :15         apt  :43      No :24
 normal:37         inept:28      Yes:47
 high  :19
```

Figure 23: Dataset summary after pre-processing.

Descriptive Statistics:

Measure of Central Tendency:

Basically ‘Central Tendency’ is measured by Mean, Median and Mode of continuous attributes,

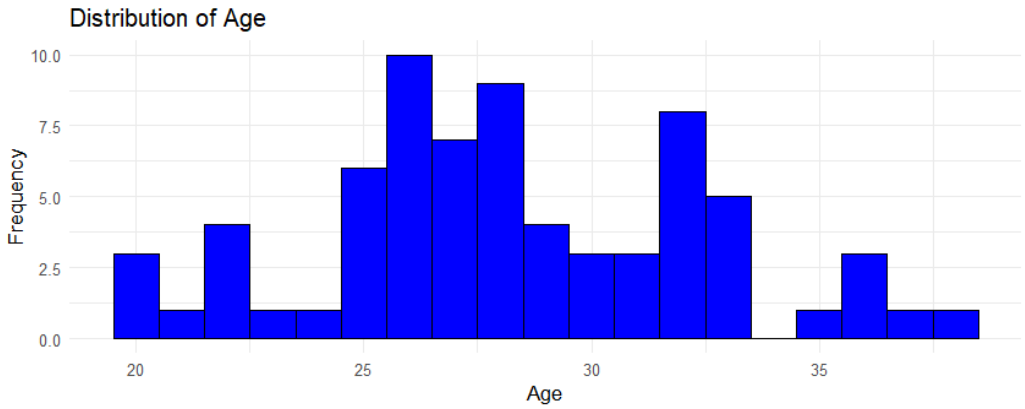
```
> mydata$Age <- as.numeric(mydata$Age)
> mydata$`Weight (Kg)` <- as.numeric(mydata$`Weight (Kg)`)

> columns_numeric <- c("Age", "Weight (Kg)")

> central_tendency_numeric <- function(column)
+   {
+   mean_val <- mean(column, na.rm = TRUE)
+   median_val <- median(column, na.rm = TRUE)
+   mode_val <- as.numeric(names(sort(table(column), decreasing = TRUE))[1])
+   list(mean = mean_val, median = median_val, mode = mode_val)
+   }

> for (col in columns_numeric)
+   {
+   print(paste("Central tendency for", col))
+   print(central_tendency_numeric(mydata[[col]]))
+   }
[1] "Central tendency for Age"
$mean
[1] 28.23944

$median
[1] 28

$mode
[1] 26

[1] "Central tendency for Weight (Kg)"
$mean
[1] 64.4338

$median
[1] 64

$mode
[1] 63
```

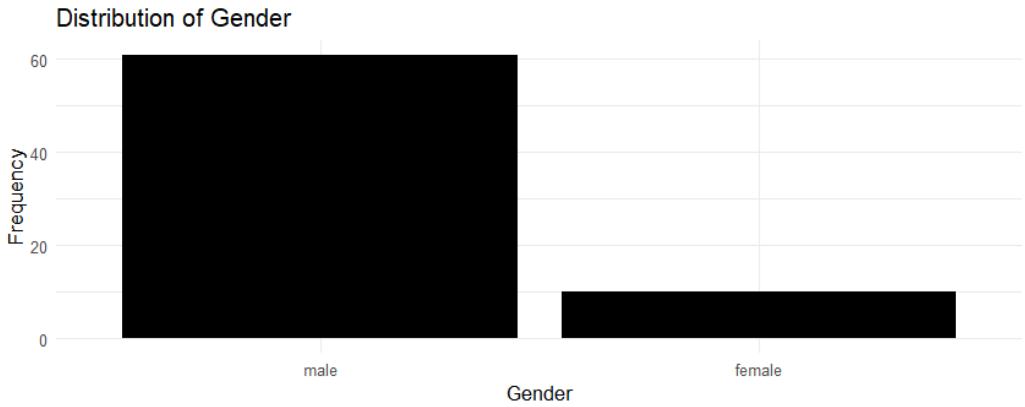
Measure of Spread:

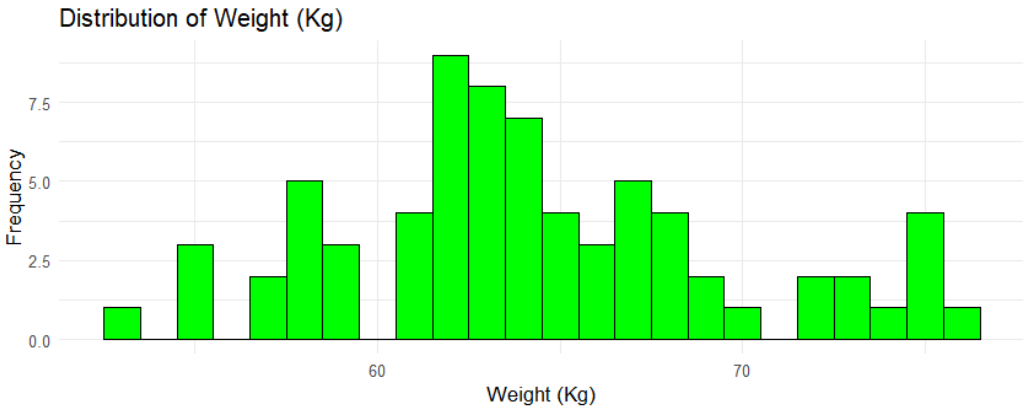
Basically ‘Spread’ is measured by Range, Variance and Standard deviation of continuous attributes,

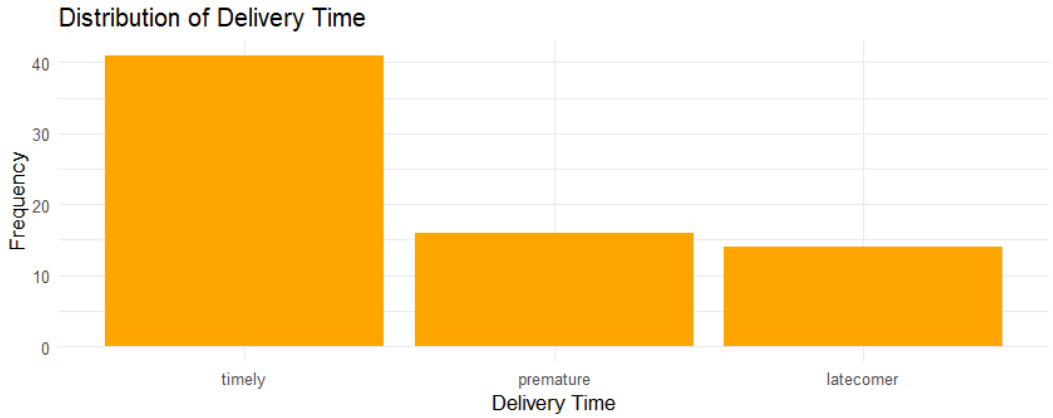
```
> measure_of_spread <- function(column)
+   {
+   range_val <- diff(range(column, na.rm = FALSE))
+   variance_val <- var(column, na.rm = FALSE)
+   sd_val <- sd(column, na.rm = FALSE)
+   list(range = range_val, variance = variance_val, sd = sd_val)
+   }

> for (col in columns_numeric) {
+   print(paste("Measures of spread for", col))
+   print(measure_of_spread(mydata[[col]]))
+ }
[1] "Measures of spread for Age"
$range
[1] 18

$variance
[1] 17.72757

$sd
[1] 4.210412

[1] "Measures of spread for Weight (Kg)"
$range
[1] 23

$variance
[1] 28.74056

$sd
[1] 5.361022
```

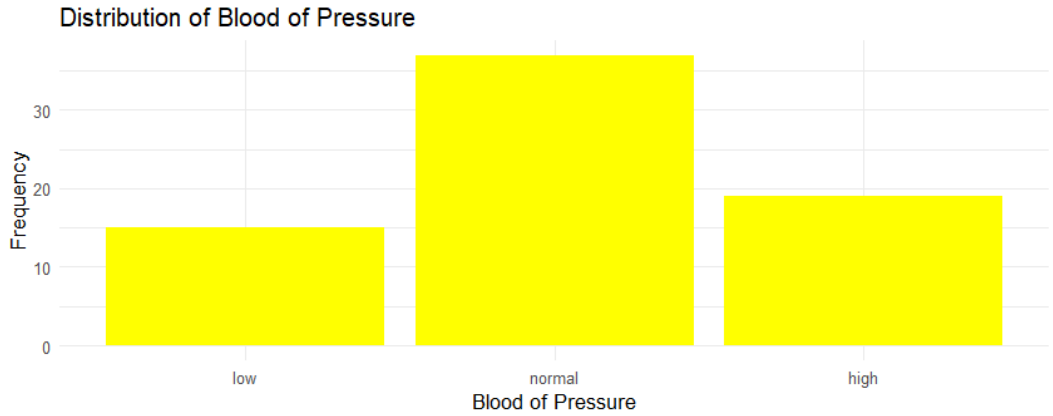
Distribution of Attributes Data:

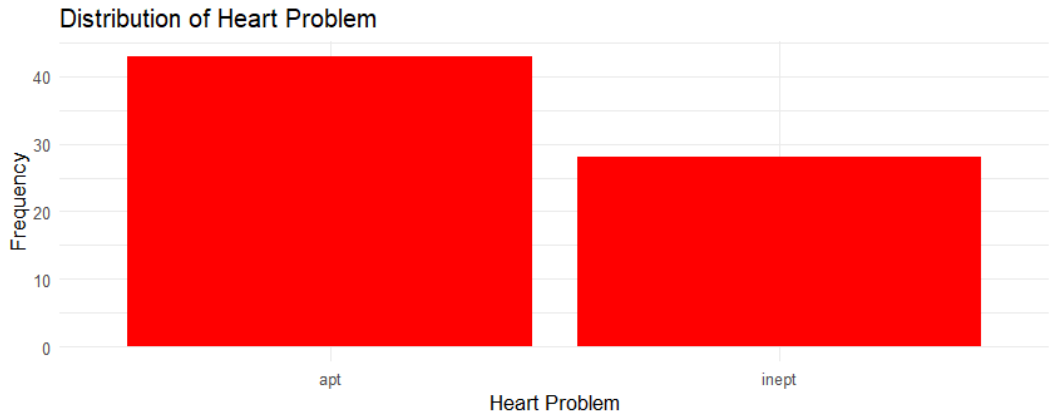


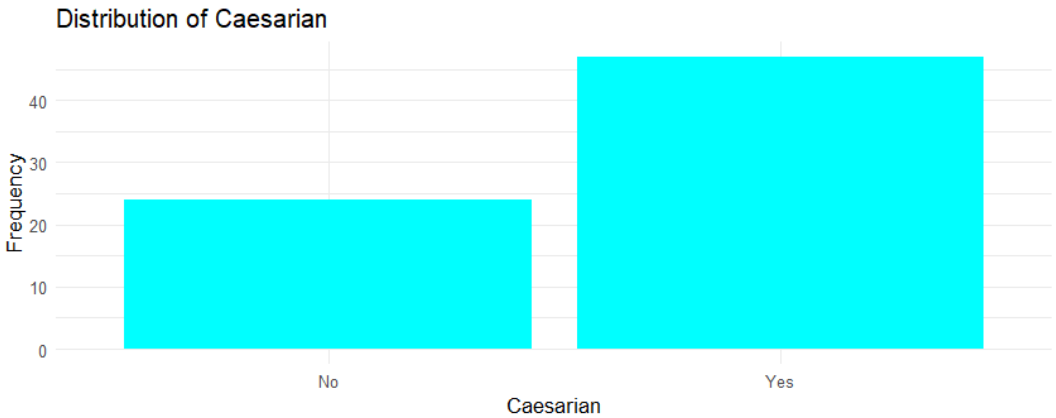





Distribution of Delivery Time
Frequency
40
30
20
10
0
timely
premature
latecomer
Delivery Time


Distribution of Blood of Pressure
Frequency
30
20
10
0
low
normal
high
Blood of Pressure


Distribution of Heart Problem
Frequency
40
30
20
10
0
apt
inept
Heart Problem


Distribution of Caesarian
Frequency
40
30
20
10
0
No
Yes
Caesarian