AMERICAN INTERNATIONALUNIVERSITY-BANGLADESH
Faculty of Engineering

Assignment Cover Page

Assignment Title:	Mid Term Data Science Project Report			
Assignment No:	1		Date of Submission:	18 July 2024
Course Title:	Introduction to Data Science			
Course Code:	01666		Section:	A
Semester:	Summer	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.: 11

No	Name	ID	Program	Signature
1	Sumiya Hur Tasnim	21-44851-2	BSc [CSE]	
2	Ahmed Safat	21-45017-2	BSc [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	

Data Set Description

The dataset from the provided link
"https://archive.ics.uci.edu/ml/datasets/Caesarian+Section+Classification+Dataset " .
is titled "The Caesarian Section Classification Dataset" from the UCI Machine Learning Repository contains information on 80 pregnant women, focusing on characteristics relevant to delivery problems. "
The dataset includes the following columns:

- **Age**: The age of the individuals, typically in years.
- **Gender:** The gender of the individuals usually categorized as 'Male' or 'Female'.
- **weight(kg)**: The weight of the individuals in kilograms.
- **Heart problem**: A binary attribute indicating the presence (1) or absence (0) of heart-related conditions.
- **Caesarian:** A binary attribute indicating whether the individual has undergone a Caesarian section (1 for Yes, 0 for No).
- **Blood of Pressure**: The blood of pressure of the individuals usually categorized as 'Normal' , 'Low' or 'High'.
- **Delivery number**: Indicates the number of deliveries a woman has had (values can be 1, 2, 3, or 4).
- **Delivery time**: Categorizes the timing of the delivery into three groups:
  - `0`: Timely
  - `1`: Premature
  - `2`: Latecomer

These attributes help in classifying and understanding delivery patterns and their potential impact on the need for a Caesarian section.

Missing Value Handling (Replace)

1. Dataset Include

```
library(readr)

file_pathMMM <- "D:/Uni_Semester/Semester 10/Mid/Data Science/Lab/Midterm_Project_Dataset_section(A).csv"
dataMMM <- read_csv(file_pathMMM)

head(dataMMM)
```

Output

```
> head(dataMMM)
# A tibble: 6 × 9
  Patient_id   Age Gender `weight(kg)` Delivery_number Delivery_time Blood  Heart Caesarian
       <dbl> <dbl> <chr>          <dbl> <chr>                 <dbl> <chr>  <dbl>     <dbl>
1          1    22 female          57.7 1                         0 high       0         0
2          2    26 male            63   2                         0 normal     0         1
3          3    26 male            62   2                         1 normal     0         0
4          4    28 male            65   1                         0 high       0         0
5          5    22 male            58   2                         0 normal     0         1
6          6    26 male            63   1                         1 low        0         0
```


At first, the dataset is included and stored in read.csv. Then read.csv is executed to show all the data.

2. Appling Mean, Median and Mode

```
dataMMM$Age[is.na(dataMMM$Age)] <- mean(dataMMM$Age, na.rm = TRUE)
dataMMM$`weight(kg)`[is.na(dataMMM$`weight(kg)`)] <- mean(dataMMM$`weight(kg)`, na.rm = TRUE)
dataMMM$Delivery_time[is.na(dataMMM$Delivery_time)] <- mean(dataMMM$Delivery_time, na.rm = TRUE)

dataMMM$Delivery_number <- as.numeric(gsub("y", "", dataMMM$Delivery_number))
dataMMM$Delivery_number[is.na(dataMMM$Delivery_number)] <- median(dataMMM$Delivery_number, na.rm = TRUE)
dataMMM$Heart[is.na(dataMMM$Heart)] <- median(dataMMM$Heart, na.rm = TRUE)
dataMMM$Caesarian[is.na(dataMMM$Caesarian)] <- median(dataMMM$Caesarian, na.rm = TRUE)

get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

dataMMM$Gender[is.na(dataMMM$Gender)] <- get_mode(dataMMM$Gender)
dataMMM$Blood[is.na(dataMMM$Blood)] <- get_mode(dataMMM$Blood)

sum(is.na(dataMMM))
```

Output

```
> # Replace missing values in numeric columns with mean
> dataMMM$Age[is.na(dataMMM$Age)] <- mean(dataMMM$Age, na.rm = TRUE)
> dataMMM$`weight(kg)`[is.na(dataMMM$`weight(kg)`)] <- mean(dataMMM$`weight(kg)`, na.rm = TRUE)
> dataMMM$Delivery_time[is.na(dataMMM$Delivery_time)] <- mean(dataMMM$Delivery_time, na.rm = TRUE)
> # Replace missing values in count columns with median
> dataMMM$Delivery_number <- as.numeric(gsub("y", "", dataMMM$Delivery_number))  # Convert non-numeric entries to numeric
> dataMMM$Delivery_number[is.na(dataMMM$Delivery_number)] <- median(dataMMM$Delivery_number, na.rm = TRUE)
> dataMMM$Heart[is.na(dataMMM$Heart)] <- median(dataMMM$Heart, na.rm = TRUE)
> dataMMM$Caesarian[is.na(dataMMM$Caesarian)] <- median(dataMMM$Caesarian, na.rm = TRUE)
> # Replace missing values in categorical columns with mode
> get_mode <- function(v) {
+   uniqv <- unique(v)
+   uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
> dataMMM$Gender[is.na(dataMMM$Gender)] <- get_mode(dataMMM$Gender)
> dataMMM$Blood[is.na(dataMMM$Blood)] <- get_mode(dataMMM$Blood)
> # Verify if all missing values are handled
> sum(is.na(dataMMM))
[1] 0
```

We handle missing value using measure of central tendency and replace that missing value

1. **Numeric columns:** Replaces missing values with the column's mean.
   - **Mean**
     - **Usage:** For numeric columns (`Age`, `weight(kg)`, `Delivery_time`).
     - **Function:** `mean(dataMMM$Age, na.rm = TRUE)`
     - **Operation:** Calculates the average of the non-missing values in the column and replaces missing values with this average.
2. **Count columns:** Replaces missing values with the column's median.
   - **Median**
     - **Usage:** For count columns (`Delivery_number`, `Heart`, `Caesarian`).
     - **Function:** `median(dataMMM$Delivery_number, na.rm = TRUE)`
     - **Operation:** Finds the middle value of the non-missing values when sorted and replaces missing values with this middle value.
3. **Categorical columns:** Replaces missing values with the most frequent value (mode).
   - **Mode**
     - **Usage:** For categorical columns (`Gender`, `Blood`).
     - **Function:** `get_mode(dataMMM$Gender)`
     - **Operation:** Finds the most frequently occurring value in the column and replaces missing values with this most frequent value.

Finally, it verifies that no missing values remain in the dataset.

3. Identifying Duplicate Patient IDs

```
duplicate_patient_ids <- dataMMM %>%
  group_by(Patient_id) %>%
  filter(n() > 1) %>%
  arrange(Patient_id)

print(duplicate_patient_ids)
```

Output

Filter

	Patient_id	Age	Gender	weight(kg)	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	12	33	male	66.64304	1	1	low	0	1
2	12	33	male	70.00000	1	1	low	0	1
3	22	33	male	75.00000	2	0	low	1	1
4	22	33	male	75.00000	2	0	low	1	1
5	53	30	male	68.00000	3	2	high	0	1
6	53	30	male	68.00000	3	2	high	0	1

Here we try to find the duplicate value from patient id because the patient id is a unique number which can't be the same for multiple persons. We find the multiple patient id in the dataset after handling the missing value using measure of central tendency.

4. Remove duplicate rows

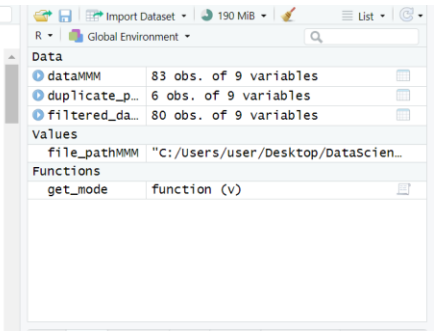
```
filtered_data_no_duplicates <- dataMMM %>%
  distinct(Patient_id, .keep_all = TRUE)

print(head(filtered_data_no_duplicates))
```

Output

Filter

	Patient_id	Age	Gender	weight(kg)	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22.00000	female	57.70000	1	0.0000	high	0	0
2	2	26.00000	male	63.00000	2	0.0000	normal	0	1
3	3	26.00000	male	62.00000	2	1.0000	normal	0	0
4	4	28.00000	male	65.00000	1	0.0000	high	0	0
5	5	22.00000	male	58.00000	2	0.0000	normal	0	1
6	6	26.00000	male	63.00000	1	1.0000	low	0	0
7	7	27.00000	male	64.00000	2	0.0000	normal	0	0
8	8	32.00000	male	70.00000	3	0.0000	normal	0	1
9	9	28.00000	female	63.50000	2	0.0000	normal	0	0
10	10	27.00000	male	64.50000	1	1.0000	normal	0	1
11	11	36.00000	male	75.00000	1	0.0000	normal	0	0
12	12	33.00000	male	66.64304	1	1.0000	low	0	1
13	13	23.00000	female	58.00000	1	1.0000	normal	0	0



Here we try to remove duplicate rows from the dataMMM dataset based on the Patient_id column, keeping only the first occurrence of each Patient_id and retaining all columns. It then prints the first few rows of the cleaned dataset for verification. Then the update dataset will be store in filtered_data_no_duplicates.

5. Noisy value handling:

➢

```
convert_data <- filtered_data_no_duplicates %>%
  mutate(Gender = ifelse(Gender %in% c("male", "female"), Gender, "male"))
```

Output

	Patient_id	Age	Gender	weight(kg)	Delivery_number	Delivery_time	Blood	Heart	Caesarian
48	48	32.00000	male	67.50000	2	0.0000	high	1	1
49	49	26.00000	male	62.50000	2	2.0000	normal	0	1
50	50	32.26582	male	66.64304	2	0.0000	low	1	1
51	51	33.00000	male	68.50000	3	2.0000	normal	1	1
52	52	21.00000	male	53.00000	2	1.0000	low	1	1
53	53	30.00000	male	68.00000	3	2.0000	high	0	1
54	54	35.00000	male	74.00000	1	1.0000	low	0	0
55	55	29.00000	male	63.50000	2	0.0000	normal	1	1
56	56	25.00000	male	59.00000	2	0.0000	normal	0	0
57	57	32.00000	male	67.50000	3	1.0000	low	1	1
58	58	95.00000	male	110.00000	1	0.0000	low	0	1
59	59	26.00000	male	61.50000	1	0.0000	high	0	1

After removing the duplicate values we find some noisy value in the filtered_data_no_duplicates which was present in patient_id 36 and 56. The value was like the ‘male’ so we convert the noisy value into male and get a update dateset named convert_data.

➢

```
delivery_time_filtered_data <- convert_data %>%
  mutate(Delivery_time = case_when(
    Delivery_time >= 0 & Delivery_time < 1 ~ 0,
    Delivery_time >= 1 & Delivery_time < 2 ~ 1,
    Delivery_time >= 2 & Delivery_time < 3 ~ 2,
    TRUE ~ Delivery_time
  ))
```

Output

	Patient_id	Age	Gender	weight(kg)	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22.00000	female	57.70000	1	0	high	0	0
2	2	26.00000	male	63.00000	2	0	normal	0	1
3	3	26.00000	male	62.00000	2	1	normal	0	0
4	4	28.00000	male	65.00000	1	0	high	0	0
5	5	22.00000	male	58.00000	2	0	normal	0	1
6	6	26.00000	male	63.00000	1	1	low	0	0

After getting convert_data we find that the Delivery_time is converted into float because of mean, median and mode. Delivery_time can’t be the float value because where 0 means timely, 1 means premature and 2 means latecomers so that we convert that 0,1 and 2 according to the condition and save the update dataset named delivery_time_filtered_data.

➢

```
age_filtered_data <- delivery_time_filtered_data %>%
  mutate(Age = as.integer(Age))
```

Output

	Patient_id	Age	Gender	weight(kg)	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.70000	1	0	high	0	0
2	2	26	male	63.00000	2	0	normal	0	1
3	3	26	male	62.00000	2	1	normal	0	0
4	4	28	male	65.00000	1	0	high	0	0
5	5	22	male	58.00000	2	0	normal	0	1
6	6	26	male	63.00000	1	1	low	0	0
7	7	27	male	64.00000	2	0	normal	0	0
8	8	32	male	70.00000	3	0	normal	0	1
9	9	28	female	63.50000	2	0	normal	0	0

After getting delivery_time_filtered_data we find that the Age attribute is converted into float because of mean, median and mode. Delivery_time can't be the float value so we change and save the update dataset named age_filtered_data.

➢

```
weight_filtered_data<- age_filtered_data %>%
  mutate(`weight(kg)` = round(`weight(kg)`, 1))
```

Output

	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	normal	0	0
10	10	27	male	64.5	1	1	normal	0	1

After getting age_filtered_data we find that the weight(kg) attribute is converted all value into float because of mean, median and mode. All weight(kg) value get many digits after point, so we convert that into one digit after point, so we change and save the update dataset named weight_filtered_data.

Missing Value Handling (Delete)

1. Dataset Include

```
path <- "C:/Users/user/Desktop/DataScience/Midterm_Project_Dataset_section(A).csv"

data <- read.csv(path)
```

Output

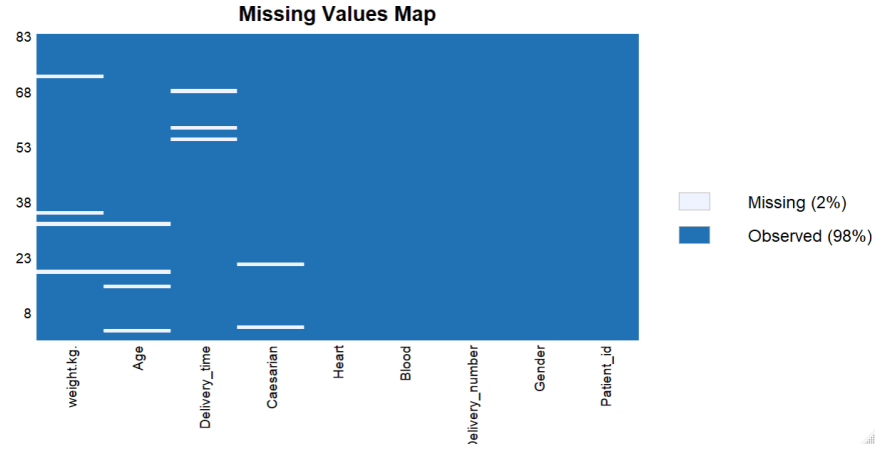
	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0

At first, the dataset is included and stored in read.csv. Then read.csv is executed to show all the data and store that into a variable called data.

2. Visualize Missing Values

```
missmap(data, main = "Missing Values Map")
```

Output



Using missmap function we see the missing value on a graph where the missing value was around 2%.

3. Detect Missing Values

```
dataset <- read_csv("C:/Users/user/Desktop/DataScience/Midterm_Project_Dataset_section(A).csv"

missing_values <- sapply(dataset, function(x) sum(is.na(x)))

print(missing_values)

total_missing_values <- sum(is.na(dataset))

print(total_missing_values)
```

Output

```
> print(missing_values)
     Patient_id             Age          Gender      weight(kg) Delivery_number   Delivery_time           Blood
              0               4               3               4               2               3               3
          Heart       Caesarian
              0               2
> |
```

missing_values	Named int [1:9] 0 4 3 4 2 3 3 0 2

We detect the missing value are present in the dataset and find around 21 missing values are present in the dataset.

4. Filter The Dataset

```
filtered_data <- data %>%
  filter(Gender %in% c("male", "female")) %>%
  filter(Blood %in% c("high", "normal", "low")) %>%
  filter(Heart %in% c("0", "1")) %>%
  filter(Caesarian %in% c("0", "1")) %>%
  filter(Delivery_number %in% c("1", "2", "3", "4")) %>%
  filter(!is.na(weight.kg.)) %>%
  filter(!is.na(Delivery_time)) %>%
  filter(!is.na(Age)) %>%
  filter(!is.na(Patient_id))
```

Output

filtered_data 62 obs. of 9 variables

	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

We filter all the column according to the condition for each individual column like gender should have male and female , blood pressure should be high/ normal/low etc. In weight , delivery_time, age and patient_id can not be null. By appling this condition we find new dataset name filtered_data. Where have 62 obs before appling the filtering process there was 83 obs.

5.Filter duplicate rows

```
duplicates <- filtered_data[duplicated(filtered_data),]
cat("Number of duplicate rows:", nrow(duplicates), "\n")
```

Output

duplicates 2 obs. of 9 variables

	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
20	22	33	male	75	2	0	low	1	1
43	53	30	male	68	3	2	high	0	1

```
> cat("Number of duplicate rows:", nrow(duplicates), "\n")
Number of duplicate rows: 2
```

We filter duplicate rows the duplicates value are present in the dataset and find 2 duplicates values are present in the dataset.

6. Remove duplicate rows

```
duplicates_clean <- sum(duplicated(filtered_data))
cat("Number of duplicate rows after cleaning:", duplicates_clean, "\n")
```

Output

duplicates_filtered_data 60 obs. of 9 variables

	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
18	21	26	male	62.0	1	1	normal	0	0
19	22	33	male	75.0	2	0	low	1	1
21	23	25	male	62.0	1	1	high	0	0

```
> cat("Number of duplicate rows after cleaning:", duplicates_clean, "\n")
Number of duplicate rows after cleaning: 2
```

We remove duplicate rows the remove value are present in the dataset and find now 60 object of 9 variable values are present in the dataset. Where have 60 obs before appling the remove duplicate rows process there was 62 obs.

7. Verify removal of duplicates

```
duplicates_clean <- sum(duplicated(filtered_data))
cat("Number of duplicate rows after cleaning:", duplicates_clean, "\n")
```

Output

```
duplicates_clean    2L
```

```
> duplicates_clean <- sum(duplicated(filtered_data))
> cat("Number of duplicate rows after cleaning:", duplicates_clean, "\n")
Number of duplicate rows after cleaning: 2
```

We verify removal of duplicates rows the duplicates value are present in the dataset and find 2 duplicates values are present in the dataset. When we apply duplicates_clean function then finally verify removal 2 values.

8. Recheck missing value using Mean, Median and Mode

```
data_clean_mean <- duplicates_filtered_data
num_cols <- sapply(duplicates_filtered_data, is.numeric)
data_clean_mean[, num_cols] <- lapply(data_clean_mean[, num_cols], function(x) replace(x, is.na(x), mean(x, na.rm = TRUE)))

data_clean_median <- duplicates_filtered_data
data_clean_median[, num_cols] <- lapply(data_clean_median[, num_cols], function(x) replace(x, is.na(x), median(x, na.rm = TRUE)))

get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

data_clean_mode <- duplicates_filtered_data
data_clean_mode[, num_cols] <- lapply(data_clean_mode[, num_cols], function(x) replace(x, is.na(x), get_mode(x[!is.na(x)])))

data_clean <- data_clean_mean
```

Output

data_clean	60 obs. of 9 variables
data_clean_mean	60 obs. of 9 variables
data_clean_median	60 obs. of 9 variables
data_clean_mode	60 obs. of 9 variables

After cleaning all duplicates values, we try to recheck is there any missing value left in the dataset using Mean, Median and Mode function. Appling those function we find in every step like Mean, Median and Mode there were 60 obs and 9 variables. In short no missing value were found in the dataset.

9. Missing values after cleaning verification

```
missing_values_clean <- sum(is.na(data_clean))
cat("Number of missing values after cleaning:", missing_values_clean, "\n")
```

Output

missing_values_clean	0L

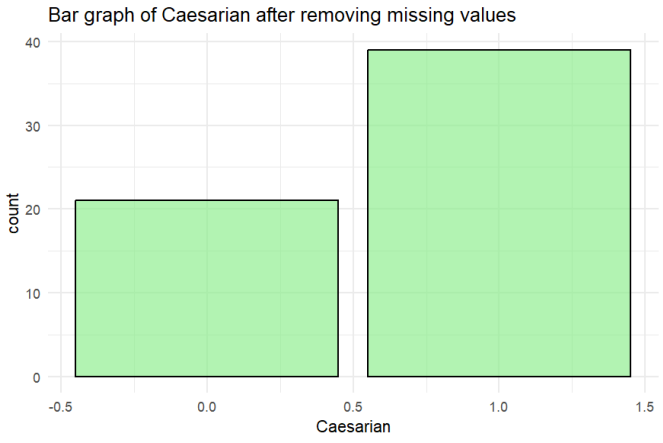
```
> missing_values_clean <- sum(is.na(data_clean))
> cat("Number of missing values after cleaning:", missing_values_clean, "\n")
Number of missing values after cleaning: 0
```

We apply missing values after cleaning verification the missing value are null in the dataset and find 0 missing values are present in the dataset.

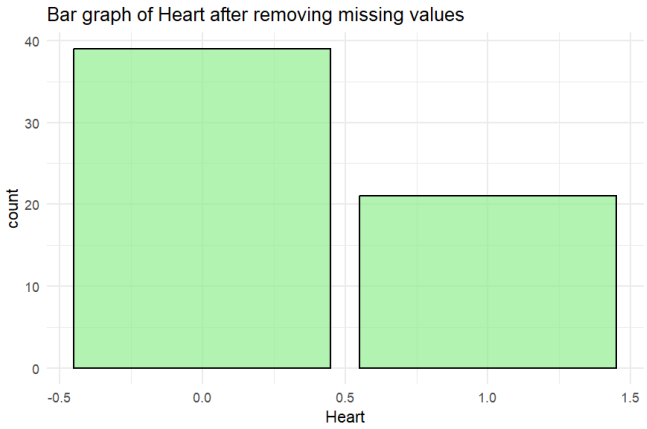
10. Graph after removing missing values

```
bar_graphs_after <- function(df) {
  for (col in colnames(df)) {
    p <- ggplot(df, aes_string(col)) +
      geom_bar(fill = 'lightgreen', color = 'black', alpha = 0.7) +
      ggtitle(paste("Bar graph of", col, "after removing missing values")) +
      theme_minimal()
    print(p)
  }
}
```

Output

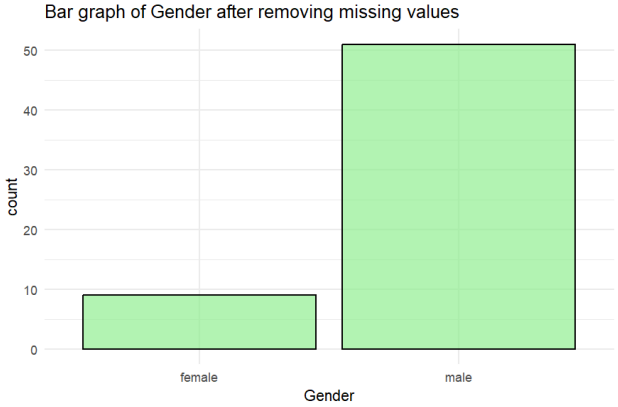


The bar chart shows the distribution of Caesarian sections after cleaning the dataset. About 20 instances had no Caesarian (0), while around 40 had Caesarian sections (1), indicating Caesarians were twice as common.

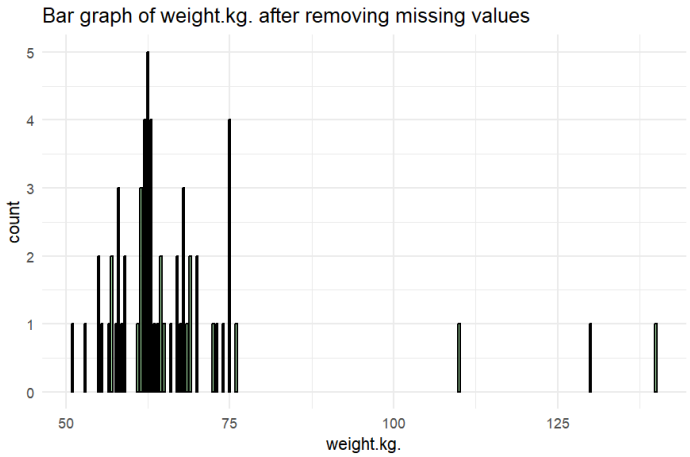


The bar graph displays the distribution of heart disease after removing missing values. Most individuals have heart disease (0.5), while fewer do not have heart disease (1.0).

The graph shows most ages are concentrated between 20 and 40 years, with a peak around 30 years. A few outliers exist above 90 years.

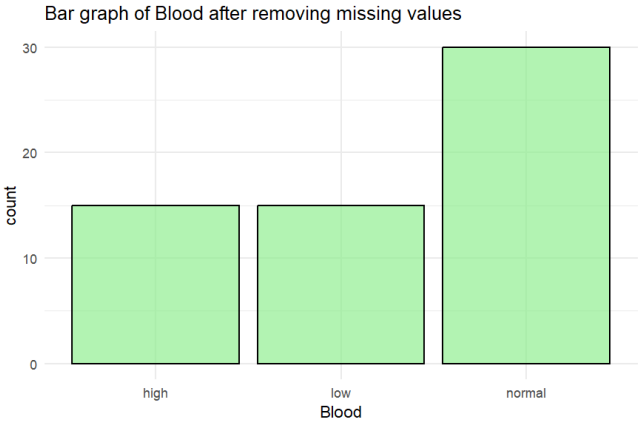


The bar graph shows the gender distribution after removing missing values. The x-axis represents the categories (female and male), and the y-axis indicates the count of individuals. There are approximately 10 females and 50 males in the dataset.



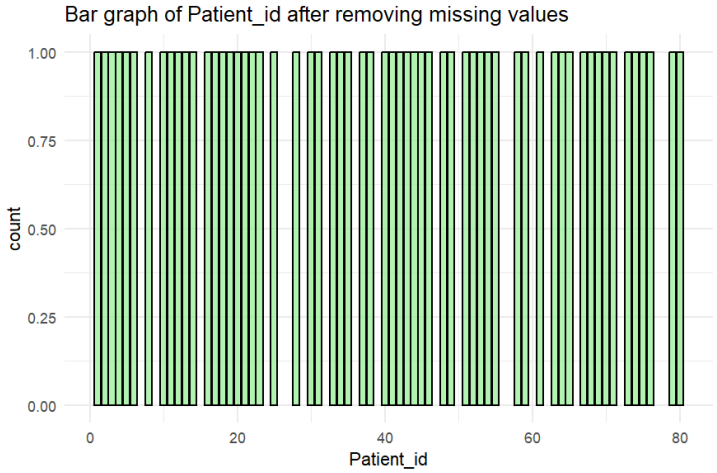
The graph shows most weights between 50-75 kg, with a peak around 70 kg. There are a few outliers at 100 kg and 125 kg.

The graph shows the counts of delivery numbers from 1 to 4. Delivery **1** has the highest count (~30), followed by **2** (~22), **3** (~10), and **4** has the lowest count (~2).



The bar graph shows the count of individuals with "high," "low," and "normal" blood pressure after removing missing data. Most have normal blood pressure (around 30), while high and low categories each have about 15 occurrences.

The graph shows delivery times, with 32 occurrences for time "0," about 12 for time "1," and around 14 for time "2." Delivery time "0" is the most frequent.



The bar graph shows the distribution of `Patient_id` values after removing missing data. The x-axis represents the `Patient_id` values, and the y-axis shows their occurrence counts. Each `Patient_id` appears only once, indicating no duplicates in the dataset.

11. Summary After Mean, Median and Mode

```
summary_stats <- duplicates_filtered_data %>%
  summarise(across(where(is.numeric), list(mean = mean, median = median)))

print(summary_stats)
```

Output

summary_stats | 1 obs. of 12 variables

	Patient_id_mean	Patient_id_median	Age_mean	Age_median	weight.kg._mean	weight.kg._median	Delivery_time_mean	Delivery_time_median	Heart_mean	Heart_median	Caesarian_mean	Caesarian_median
1	39.51667	40.5	33.33333	27	67.02167	63	0.7166667	0	0.35	0	0.65	1

We apply mean and median on those columns which are only numeric get such type of output.

12. Calculate Mode after removing missing values

```
modes <- sapply(duplicates_filtered_data %>% select(where(is.numeric)), get_mode)
cat("Modes of numeric attributes:\n")
print(modes)
```

Output

```
> print(modes)
   Patient_id           Age     weight.kg. Delivery_time         Heart     Caesarian
          1.0          26.0          62.5           0.0           0.0           1.0
```

We apply mode on those columns which are only numeric get such type of output.

13. Plot mean, median, and mode on graph

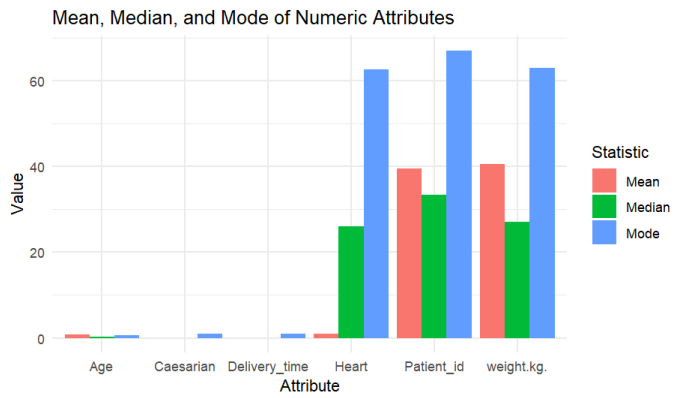
```
numeric_columns <- duplicates_filtered_data %>% select(where(is.numeric))
stats <- data.frame(Attribute = rep(names(numeric_columns), each = 3),
                    Statistic = rep(c("Mean", "Median", "Mode"), times = ncol(numeric_columns)),
                    Value = c(sapply(numeric_columns, mean),
                              sapply(numeric_columns, median),
                              sapply(numeric_columns, get_mode)))

ggplot(stats, aes(x = Attribute, y = Value, fill = Statistic)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  ggtitle("Mean, Median, and Mode of Numeric Attributes")
```

	Patient_id	Age	weight.kg.	Delivery_time	Heart	Caesarian
1	1	22	57.7	0	0	0
2	2	26	63.0	0	0	1
3	3	26	62.0	1	0	0
4	4	28	65.0	0	0	0
5	5	22	58.0	0	0	1
6	6	26	63.0	1	0	0
7	8	32	70.0	0	0	1
8	10	27	64.5	1	0	1
9	11	36	75.0	0	0	0
10	12	33	70.0	1	0	1
11	13	23	58.0	1	0	0
12	14	20	55.0	0	1	0
13	16	25	61.5	2	0	0
14	17	25	61.5	0	0	0
15	18	20	55.5	2	0	1
16	19	37	76.0	0	1	1

Output

numeric_columns | 60 obs. of 6 variables



The bar graph shows the mean (red), median (green), and mode (blue) for various numeric attributes (Age, Caesarian, Delivery_time, Heart, Patient_id, weight.kg.). The differences in these values indicate varying data distributions across the attributes.

14. Convert Heart and Caesarian columns into categorical attributes

```
categorical_dataset_after_convert <- duplicates_filtered_data %>%
  mutate(Heart = ifelse(Heart == 1, "Yes", "No"),
         Caesarian = ifelse(Caesarian == 1, "Yes", "No"))
```

Output

categorical_dataset_after_convert | 60 obs. of 9 variables

	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	No	No
2	2	26	male	63.0	2	0	normal	No	Yes
3	3	26	male	62.0	2	1	normal	No	No
4	4	28	male	65.0	1	0	high	No	No
5	5	22	male	58.0	2	0	normal	No	Yes
6	6	26	male	63.0	1	1	low	No	No
7	8	32	male	70.0	3	0	normal	No	Yes
8	10	27	male	64.5	1	1	normal	No	Yes
9	11	36	male	75.0	1	0	normal	No	No
10	12	33	male	70.0	1	1	low	No	Yes
11	13	23	female	58.0	1	1	normal	No	No
12	14	20	male	55.0	1	0	normal	Yes	No
13	16	25	female	61.5	1	2	low	No	No
14	17	25	male	61.5	1	0	normal	No	No
15	18	20	male	55.5	1	2	high	No	Yes
16	19	37	male	76.0	3	0	normal	Yes	Yes

In the dataset for Heart and Casesarian attributes, 1 represents yes, and 0 means no that's why we replaced the 1 with yes and 0 with no. Because for converting numerical to categorical we have done these steps.

15. Apply Normalization function for Weight attribute

```
min_max_normalize <- function(x) {
  (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
}

normalized_dataset <- duplicates_filtered_data %>%
  mutate(NormalizedWeight = min_max_normalize(weight.kg.))

head(normalized_dataset)
```

Output

	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian	NormalizedWeight
1	1	22	female	57.7	1	0	high	0	0	0.07528090
2	2	26	male	63.0	2	0	normal	0	1	0.13483146
3	3	26	male	62.0	2	1	normal	0	0	0.12359551
4	4	28	male	65.0	1	0	high	0	0	0.15730337
5	5	22	male	58.0	2	0	normal	0	1	0.07865169
6	6	26	male	63.0	1	1	low	0	0	0.13483146

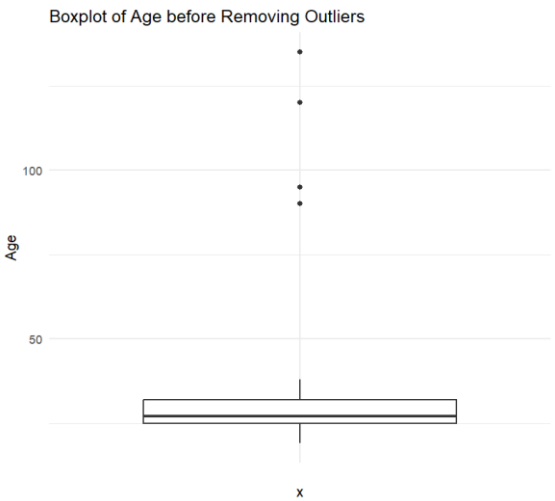
normalized_dataset	60 obs. of 10 variables

To apply the normalization function we have to know the min and max number after finding the normalize value for weight attribute for each number we build a column named Normalized weight**.**

16. Plot age data before removing outliers

```
ggplot(duplicates_filtered_data, aes(x = "", y = Age)) +
  geom_boxplot() +
  labs(title = "Boxplot of Age before Removing Outliers") +
  theme_minimal()
```

Output



The boxplot shows the age distribution before removing outliers. Most individuals are in their 20s or 30s, with a few outliers in their 80s or 90s, represented by dots.

17. Removing outliers

```
filtered_data_no_outliers <- duplicates_filtered_data %>%
  filter(Age <= 40)
```

Output

filtered_data_no_outliers | 56 obs. of 9 variables

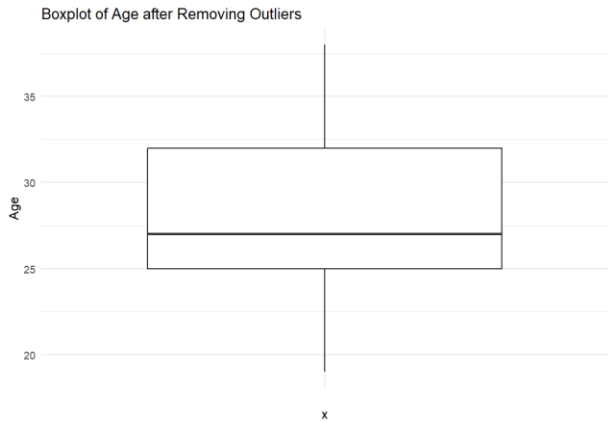
	Patient_id	Age	Gender	weight.kg.	Delivery_number	Delivery_time	Blood	Heart	Caesarian
1	1	22	female	57.7	1	0	high	0	0
2	2	26	male	63.0	2	0	normal	0	1
3	3	26	male	62.0	2	1	normal	0	0
4	4	28	male	65.0	1	0	high	0	0
5	5	22	male	58.0	2	0	normal	0	1
6	6	26	male	63.0	1	1	low	0	0
7	8	32	male	70.0	3	0	normal	0	1
8	10	27	male	64.5	1	1	normal	0	1
9	11	36	male	75.0	1	0	normal	0	0
10	12	33	male	70.0	1	1	low	0	1
11	13	23	female	58.0	1	1	normal	0	0
12	14	20	male	55.0	1	0	normal	1	0
13	16	25	female	61.5	1	2	low	0	0
14	17	25	male	61.5	1	0	normal	0	0
15	18	20	male	55.5	1	2	high	0	1

We observed the full attribute and fixed a value 40. If the vale is greeter or equal 40 then the value will be count as outliers. So we find 4 outliers and remove them from the dataset.

18. Plot age data after removing outliers

```
ggplot(filtered_data_no_outliers, aes(x = "", y = Age)) +
  geom_boxplot() +
  labs(title = "Boxplot of Age after Removing Outliers") +
  theme_minimal()
```

Output



The boxplot shows the age distribution after removing outliers. Most individuals are in their 20s or 30s. The box represents the interquartile range, with the median marked by a line, and the whiskers show the minimum and maximum values. No outliers remain, so the data is more centered around the middle values.Finally, we remove the outliers and show that on plot.

********** *End* **********