AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

Introduction to Data Science

Spring 2024-25

Section: A

Group: 2

Mid-Project Report on

Data Preparation and Exploratory Analysis of a Heart Disease Dataset

Supervised By

DR. ABDUS SALAM

Submitted By

Name	ID	Contribution
Sristi Paul	22-47195-1	
Zarin Tasnim	21-44898-2	
Mussa Alam	21-45036-2	
Mahfujul Islam	21-44629-1	

Date of Submission: **26th April 2025**

TABLE OF CONTENTS

1. Dataset Description:

This research uses the Cleveland dataset, originally part of a collection from four sources. The dataset aims to predict heart disease likelihood. Although it initially contained 76 attributes, most studies focus on 14 key features. Here, 216 records and 16 variables are selected for analysis. Here is a more detailed explanation of the key attributes:

- **Age:** The patient's age in years. This numerical variable represents the patient's age, which is a significant factor in assessing heart disease risk.

- **Sex:** Gender of the patient. This attribute is important as heart disease can manifest differently in males and females.

- **Chest Pain Type (cp):** Type of chest pain experienced by the patient. This is a categorical variable with four possible values:
  - 0 = Typical angina: Chest pain that occurs with exertion and is relieved by rest.
  - 1 = Atypical angina: Chest pain that does not fit the typical pattern of angina.
  - 2 = Non-anginal pain: Chest pain that is not related to heart disease.
  - 3 = Asymptomatic: The patient does not experience chest pain.

- **Resting Blood Pressure (trestbps):** Resting blood pressure of the patient, measured in mm Hg. This numerical variable indicates the pressure of blood against the artery walls when the heart is at rest. High resting blood pressure is a major risk factor for heart disease.

- **Serum Cholesterol (chol):** Serum cholesterol level, measured in mg/dl. This numerical variable represents the amount of cholesterol in the patient's blood. High cholesterol levels can lead to the buildup of plaque in the arteries, increasing the risk of heart disease.

- **Fasting Blood Sugar (fbs):** Fasting blood sugar level, indicating whether the patient's blood sugar level is greater than 120 mg/dl:
  - 1 = true (if fasting blood sugar > 120 mg/dl)
  - 0 = false (if fasting blood sugar <= 120 mg/dl). Elevated fasting blood sugar can be a sign of diabetes, which is a risk factor for heart disease.

- **Resting Electrocardiographic Results (restecg):** Results of the patient's resting electrocardiogram, a test that measures the electrical activity of the heart. This categorical variable has three possible values:
  - 0 = Normal
  - 1 = ST-T wave abnormality: Indicates changes in the heart's electrical activity that may suggest heart disease.
  - 2 = Left ventricular hypertrophy: Indicates thickening of the heart's main pumping chamber, which can be a sign of heart disease.

- **Maximum Heart Rate Achieved (thalach):** The maximum heart rate achieved by the patient during exercise, measured numerically. This can provide insights into the heart's function under stress.

- **Exercise Induced Angina (exang):** Indicates whether the patient experiences chest pain during exercise:
  - 1 = yes
  - 0 = no

- **ST Depression (oldpeak):** ST depression induced by exercise relative to rest. This numerical variable measures the degree to which the ST segment on an electrocardiogram is depressed during exercise, which can indicate heart disease.

- **Slope:** The slope of the peak exercise ST segment, a categorical variable with three possible values:
  - 0 = Upsloping
  - 1 = Flat
  - 2 = Downsloping. This attribute reflects the change in the ST segment during exercise and can provide information about blood flow to the heart.

- **Number of Major Vessels (ca):** The number of major blood vessels (0-3) colored by fluoroscopy. Fluoroscopy is an imaging technique that can show how blood flows through the arteries. This numerical variable can indicate the extent of coronary artery disease.

- **Thal:** Thalassemia, a blood disorder, with three possible values:
  - 1 = Normal
  - 2 = Fixed defect
  - 3 = Reversible defect

- **Target (num):** Indicates the presence of heart disease:
  - 0 = No heart disease
  - 1 = Heart disease

This dataset is a valuable resource for training machine learning models aimed at the early detection and prevention of heart disease. The data can be used to identify patients who are at high risk of developing heart disease, allowing for timely interventions and lifestyle changes.

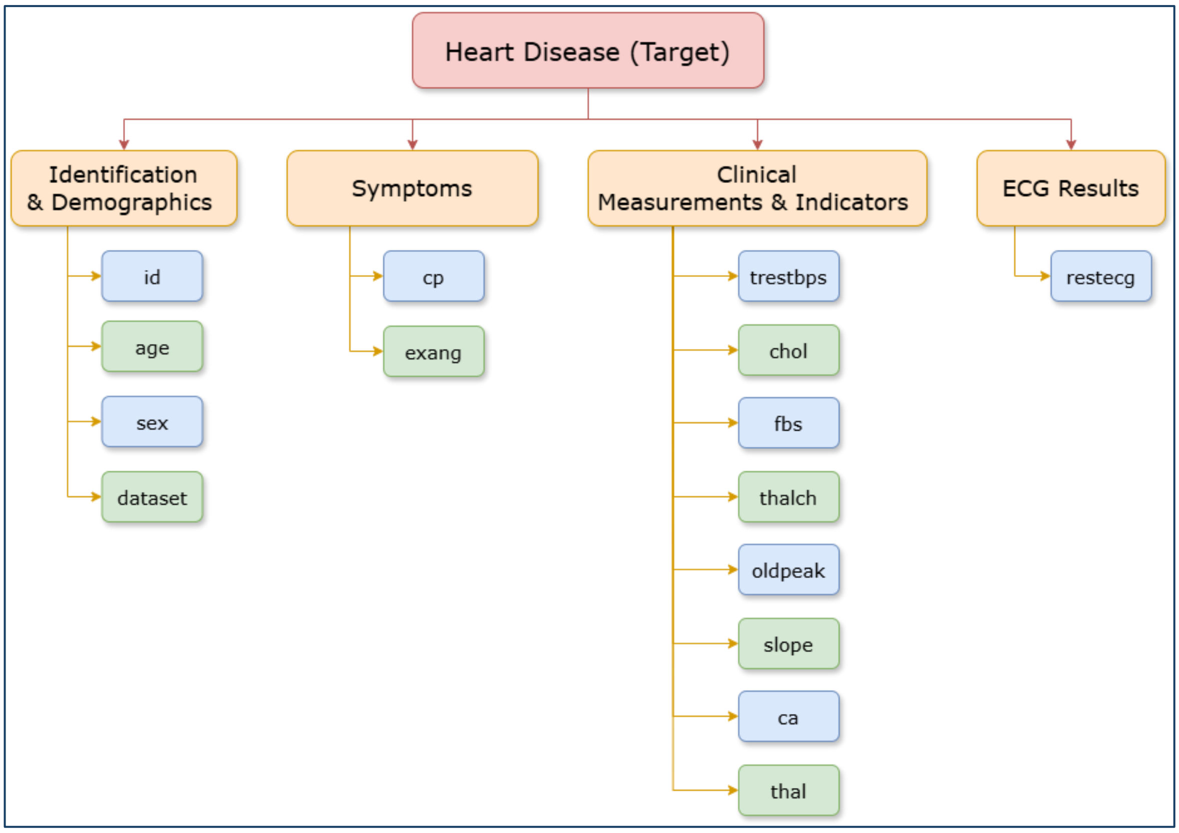
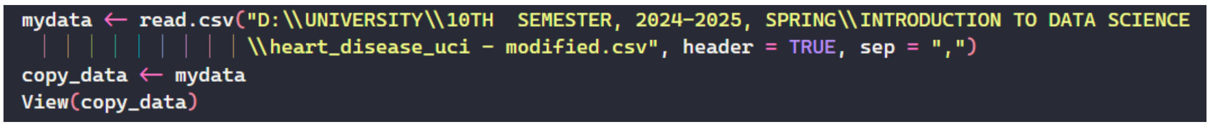


Figure 1: Flowchart of the Dataset

2. Data Preparation and Initial Analysis

2.1 Loading and Viewing the Dataset

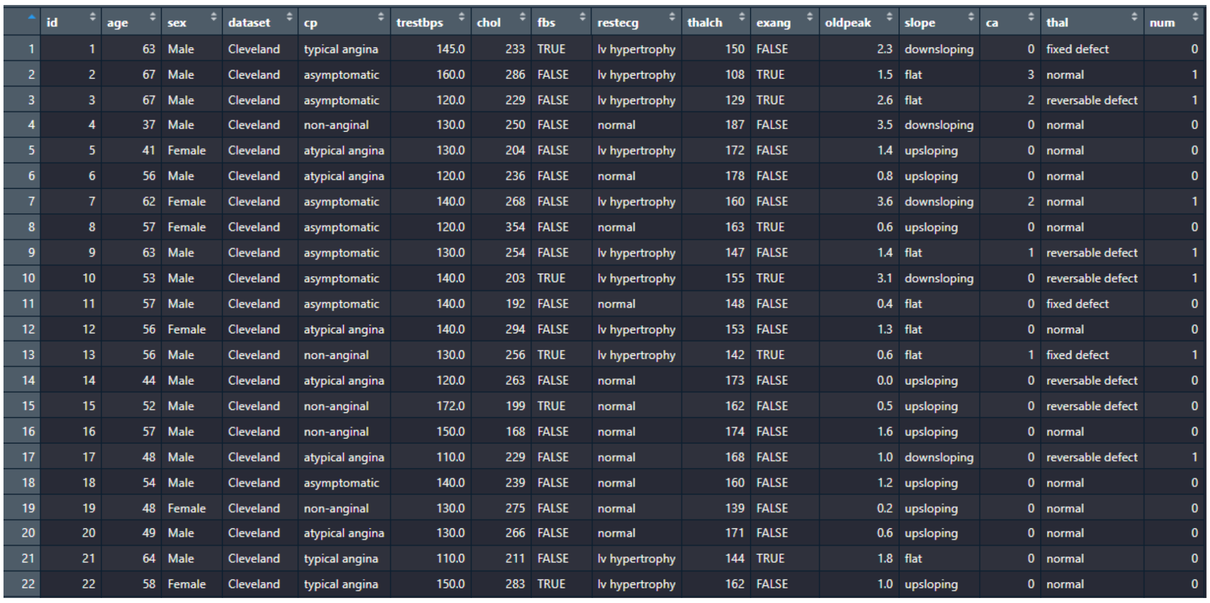
Code:



```r
mydata <- read.csv("D:\\UNIVERSITY\\10TH  SEMESTER, 2024-2025, SPRING\\INTRODUCTION TO DATA SCIENCE
                   \\heart_disease_uci - modified.csv", header = TRUE, sep = ",")
copy_data <- mydata
View(copy_data)
```

Code 1: Loading and Viewing the Dataset in R

Output:



	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
1	1	63	Male	Cleveland	typical angina	145.0	233	TRUE	lv hypertrophy	150	FALSE	2.3	downsloping	0	fixed defect	0
2	2	67	Male	Cleveland	asymptomatic	160.0	286	FALSE	lv hypertrophy	108	TRUE	1.5	flat	3	normal	1
3	3	67	Male	Cleveland	asymptomatic	120.0	229	FALSE	lv hypertrophy	129	TRUE	2.6	flat	2	reversable defect	1
4	4	37	Male	Cleveland	non-anginal	130.0	250	FALSE	normal	187	FALSE	3.5	downsloping	0	normal	0
5	5	41	Female	Cleveland	atypical angina	130.0	204	FALSE	lv hypertrophy	172	FALSE	1.4	upsloping	0	normal	0
6	6	56	Male	Cleveland	atypical angina	120.0	236	FALSE	normal	178	FALSE	0.8	upsloping	0	normal	0
7	7	62	Female	Cleveland	asymptomatic	140.0	268	FALSE	lv hypertrophy	160	FALSE	3.6	downsloping	2	normal	1
8	8	57	Female	Cleveland	asymptomatic	120.0	354	FALSE	normal	163	TRUE	0.6	upsloping	0	normal	0
9	9	63	Male	Cleveland	asymptomatic	130.0	254	FALSE	lv hypertrophy	147	FALSE	1.4	flat	1	reversable defect	1
10	10	53	Male	Cleveland	asymptomatic	140.0	203	TRUE	lv hypertrophy	155	TRUE	3.1	downsloping	0	reversable defect	1
11	11	57	Male	Cleveland	asymptomatic	140.0	192	FALSE	normal	148	FALSE	0.4	flat	0	fixed defect	0
12	12	56	Female	Cleveland	atypical angina	140.0	294	FALSE	lv hypertrophy	153	FALSE	1.3	flat	0	normal	0
13	13	56	Male	Cleveland	non-anginal	130.0	256	TRUE	lv hypertrophy	142	TRUE	0.6	flat	1	fixed defect	1
14	14	44	Male	Cleveland	atypical angina	120.0	263	FALSE	normal	173	FALSE	0.0	upsloping	0	reversable defect	0
15	15	52	Male	Cleveland	non-anginal	172.0	199	TRUE	normal	162	FALSE	0.5	upsloping	0	reversable defect	0
16	16	57	Male	Cleveland	non-anginal	150.0	168	FALSE	normal	174	FALSE	1.6	upsloping	0	normal	0
17	17	48	Male	Cleveland	atypical angina	110.0	229	FALSE	normal	168	FALSE	1.0	downsloping	0	reversable defect	1
18	18	54	Male	Cleveland	asymptomatic	140.0	239	FALSE	normal	160	FALSE	1.2	upsloping	0	normal	0
19	19	48	Female	Cleveland	non-anginal	130.0	275	FALSE	normal	139	FALSE	0.2	upsloping	0	normal	0
20	20	49	Male	Cleveland	atypical angina	130.0	266	FALSE	normal	171	FALSE	0.6	upsloping	0	normal	0
21	21	64	Male	Cleveland	typical angina	110.0	211	FALSE	lv hypertrophy	144	TRUE	1.8	flat	0	normal	0
22	22	58	Female	Cleveland	typical angina	150.0	283	TRUE	lv hypertrophy	162	FALSE	1.0	upsloping	0	normal	0

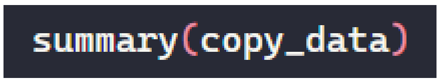
Output 1: First Few Rows of the Loaded Heart Disease Dataset in R

Description:

This code loads the dataset from a CSV file into the R environment using the read.csv() function. It assigns the data to the variable mydata. Here, a copy of the original dataset mydata is created. The new copy is stored in the variable copy_data, which will be used for further manipulations. The View() function is used to open the dataset in a separate viewer window, which allows for an easy, interactive inspection of the data in the copy_data variable.

2.2 Dataset Summary

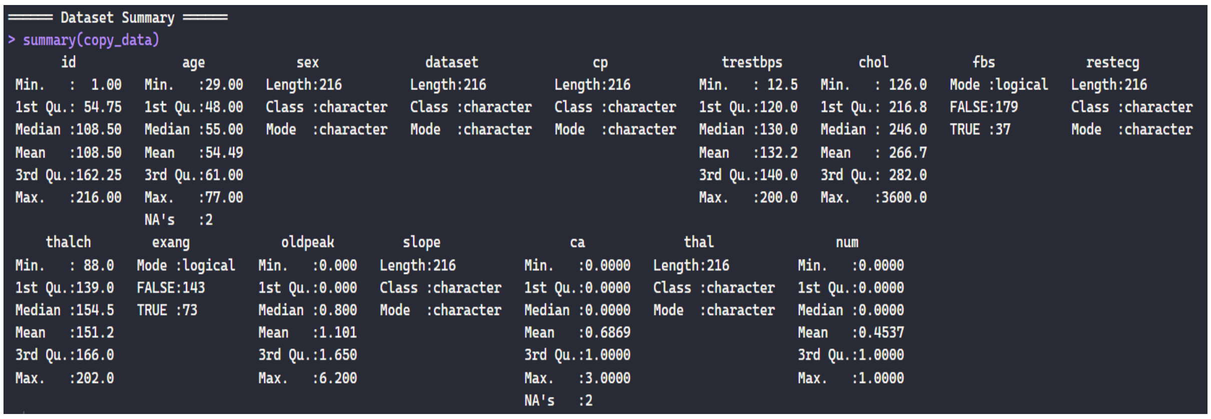
Code:



```r
summary(copy_data)
```

Code 2: R Script for Generating Descriptive Statistics

Output:



```
===== Dataset Summary =====
> summary(copy_data)
       id              age             sex              dataset               cp               trestbps          chol            fbs           restecg         
 Min.   :  1.00   Min.   :29.00   Length:216         Length:216         Length:216         Min.   : 12.5   Min.   : 126.0   Mode :logical   Length:216        
 1st Qu.: 54.75   1st Qu.:48.00   Class :character   Class :character   Class :character   1st Qu.:120.0   1st Qu.: 216.8   FALSE:179       Class :character  
 Median :108.50   Median :55.00   Mode  :character   Mode  :character   Mode  :character   Median :130.0   Median : 246.0   TRUE :37        Mode  :character  
 Mean   :108.50   Mean   :54.49                                                            Mean   :132.2   Mean   : 266.7                                     
 3rd Qu.:162.25   3rd Qu.:61.00                                                            3rd Qu.:140.0   3rd Qu.: 282.0                                     
 Max.   :216.00   Max.   :77.00                                                            Max.   :200.0   Max.   :3600.0                                     
                  NA's   :2                                                                                                                                 
     thalch         exang            oldpeak          slope                  ca              thal                 num        
 Min.   : 88.0   Mode :logical   Min.   :0.000   Length:216         Min.   :0.0000   Length:216         Min.   :0.0000  
 1st Qu.:139.0   FALSE:143       1st Qu.:0.000   Class :character   1st Qu.:0.0000   Class :character   1st Qu.:0.0000  
 Median :154.5   TRUE :73        Median :0.800   Mode  :character   Median :0.0000   Mode  :character   Median :0.0000  
 Mean   :151.2                   Mean   :1.101                      Mean   :0.6869                      Mean   :0.4537  
 3rd Qu.:166.0                   3rd Qu.:1.650                      3rd Qu.:1.0000                      3rd Qu.:1.0000  
 Max.   :202.0                   Max.   :6.200                      Max.   :3.0000                      Max.   :1.0000  
                                                                    NA's   :2                                            
```

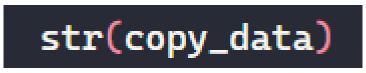
Output 2: Summary Statistics of the Heart Disease Dataset

Description:

The summary function was applied to the copy_data dataset, which provided a summary of the dataset, including statistics such as the minimum, maximum, mean, median, and quartiles for each column. This summary gave an overview of the central tendencies and distributions of the data.

2.3 Dataset Structure

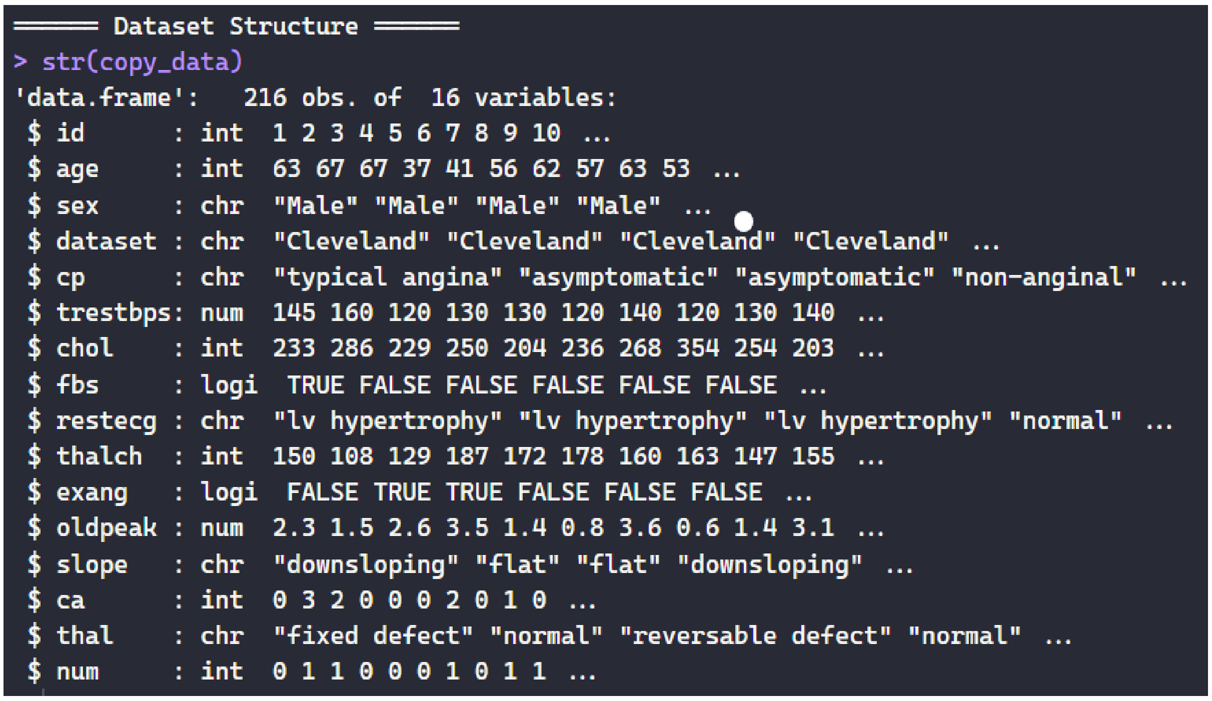
Code:



```
str(copy_data)
```

Code 3: R Script for Examining the Structure of the Heart Disease Dataset

Output:



```
===== Dataset Structure =====
> str(copy_data)
'data.frame':   216 obs. of  16 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
 $ sex     : chr  "Male" "Male" "Male" "Male" ...
 $ dataset : chr  "Cleveland" "Cleveland" "Cleveland" "Cleveland" ...
 $ cp      : chr  "typical angina" "asymptomatic" "asymptomatic" "non-anginal" ...
 $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
 $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
 $ fbs     : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
 $ restecg : chr  "lv hypertrophy" "lv hypertrophy" "lv hypertrophy" "normal" ...
 $ thalch  : int  150 108 129 187 172 178 160 163 147 155 ...
 $ exang   : logi  FALSE TRUE TRUE FALSE FALSE FALSE ...
 $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
 $ slope   : chr  "downsloping" "flat" "flat" "downsloping" ...
 $ ca      : int  0 3 2 0 0 0 2 0 1 0 ...
 $ thal    : chr  "fixed defect" "normal" "reversable defect" "normal" ...
 $ num     : int  0 1 1 0 0 0 1 0 1 1 ...
```

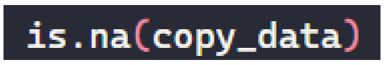
Output 3: Structure of the Heart Disease Dataset

Description:

The str function was used to display the structure of the copy_data dataset. This function provided a detailed breakdown of the dataset’s structure, including the data types of each column and a preview of the first few entries in each column, which helps in understanding the dataset's organization.

2.4 Missing Values Check:

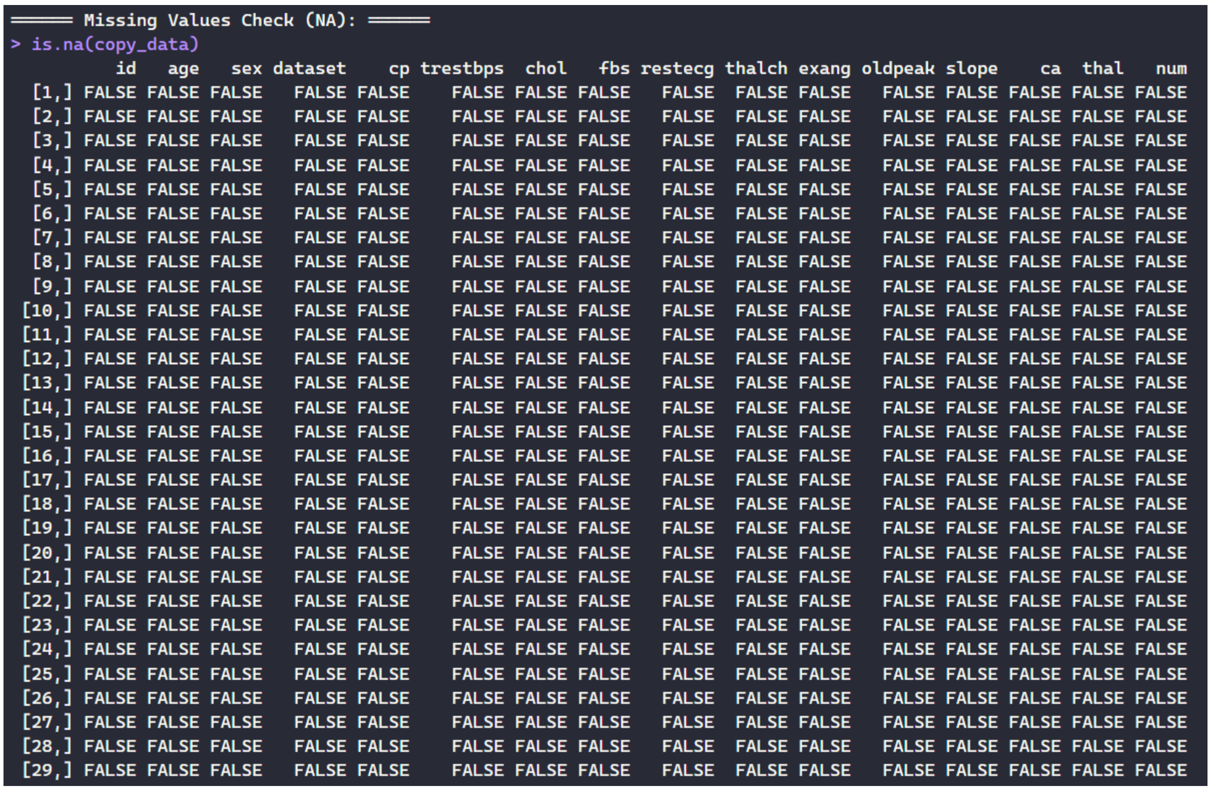
Code:



```r
is.na(copy_data)
```

Code 4: R Script for Initial Check of Missing Values

Output:



```
===== Missing Values Check (NA): =====
> is.na(copy_data)
          id   age   sex dataset    cp trestbps  chol   fbs restecg thalch exang oldpeak slope    ca  thal   num
  [1,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [2,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [3,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [4,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [5,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [6,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [7,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [8,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
  [9,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [10,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [11,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [12,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [13,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [14,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [15,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [16,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [17,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [18,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [19,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [20,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [21,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [22,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [23,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [24,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [25,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [26,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [27,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [28,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
 [29,] FALSE FALSE FALSE   FALSE FALSE    FALSE FALSE FALSE   FALSE  FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
```

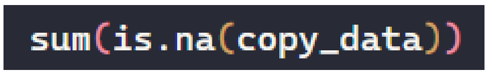
Output 4: Logical Matrix Indicating Missing Values in the Heart Disease Dataset

Description:

This code uses the is.na() function to check for missing values (NA) within the dataset. It returns a logical matrix where each element is TRUE if the corresponding value is missing and FALSE otherwise. This allows for easy identification of missing data points.

3 Total Missing Values

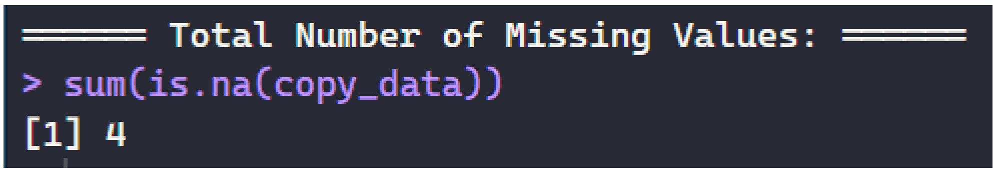
Code:



```
sum(is.na(copy_data))
```

Code 5: R Script for Counting Total Missing Values

Output:



```
====== Total Number of Missing Values: ======
> sum(is.na(copy_data))
[1] 4
```

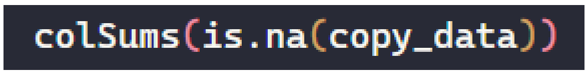
Output 5: Total Number of Missing Values in the Heart Disease Dataset

Description:

The total number of missing values in the dataset was calculated using sum(is.na(copy_data)). This count provided a measure of the extent of missing data, which helped assess how much of the dataset might need to be fixed or removed.

4 Missing Values Per Column:

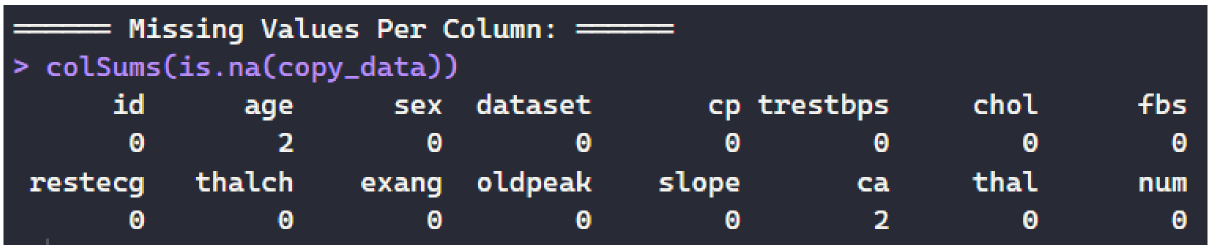
Code:



```r
colSums(is.na(copy_data))
```

Code 6: R Script for Counting Missing Values per Column

Output:



```
===== Missing Values Per Column: =====
> colSums(is.na(copy_data))
      id      age      sex  dataset       cp trestbps     chol      fbs 
       0        2        0        0        0        0        0        0 
 restecg   thalch    exang  oldpeak    slope       ca     thal      num 
       0        0        0        0        0        2        0        0 
```

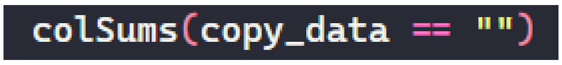
Output 6: Number of Missing Values per Column

Description:

The colSums(is.na(copy_data)) function was used to count the number of missing values in each column of the dataset. This allowed a column-by-column breakdown of missing data, helping to identify which columns had the most missing values.

5 Empty Strings Per Column:

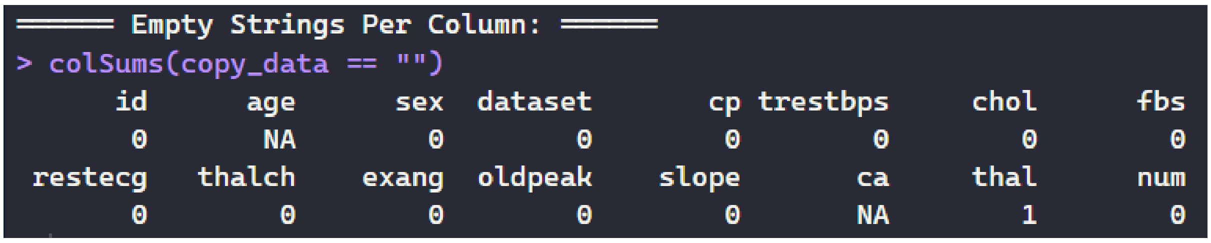
Code:



```
colSums(copy_data == "")
```

Code 7: R Script for Identifying Empty Strings

Output:



```
====== Empty Strings Per Column: ======
> colSums(copy_data == "")
      id      age      sex  dataset       cp trestbps     chol      fbs
       0       NA        0        0        0        0        0        0
 restecg   thalch    exang  oldpeak    slope       ca     thal      num
       0        0        0        0        0       NA        1        0
```

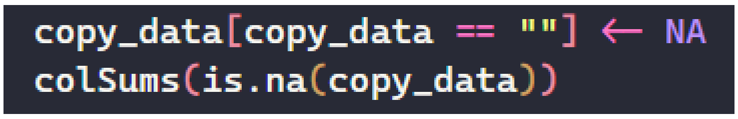
Output 7: Number of Empty Strings per Column

Description:

The colSums(copy_data == "") function was used to identify the number of empty strings in each column. This check highlighted where missing or incomplete data might have been represented as empty strings rather than NA, providing insight into the consistency of the dataset.

6 After Replacing Empty Strings:

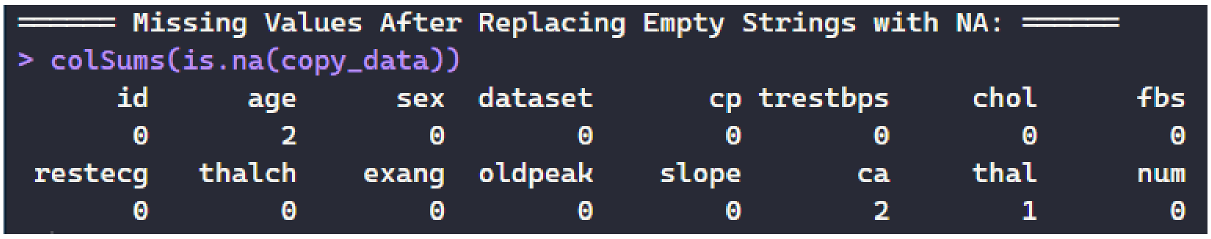
Code:



```
copy_data[copy_data == ""] <- NA
colSums(is.na(copy_data))
```

Code 8: Script for Replacing Empty Strings with NA

Output:



```
====== Missing Values After Replacing Empty Strings with NA: ======
> colSums(is.na(copy_data))
      id      age      sex  dataset       cp trestbps     chol      fbs 
       0        2        0        0        0        0        0        0 
 restecg  thalch    exang  oldpeak    slope       ca     thal      num 
       0        0        0        0        0        2        1        0 
```

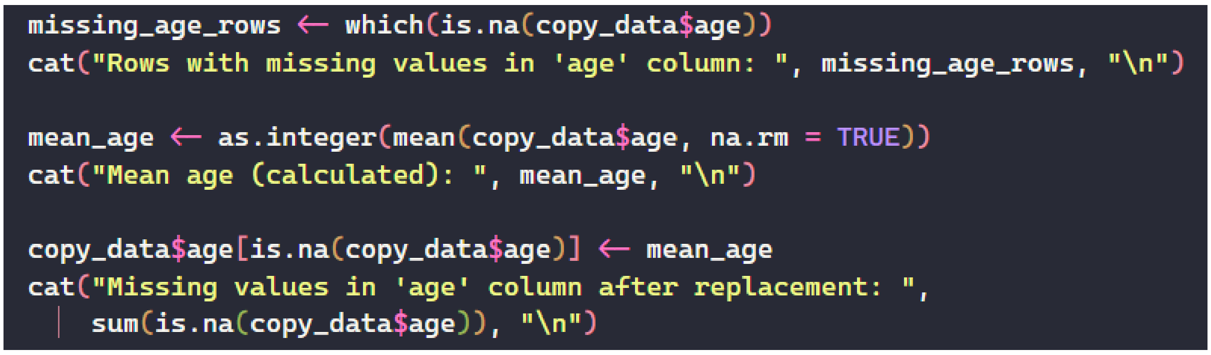
Output 8: Number of Missing Values per Column After Replacing Empty Strings with NA

Description:

The empty strings in the dataset were replaced with NA using copy_data[copy_data == ""] <- NA. After this replacement, the colSums(is.na(copy_data)) function was applied again to count the missing values across all columns. This ensured that all missing or empty data points were uniformly represented as NA.

7 Missing age Values Replaced with Mean

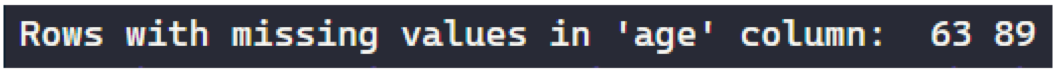
Code:



```r
missing_age_rows <- which(is.na(copy_data$age))
cat("Rows with missing values in 'age' column: ", missing_age_rows, "\n")

mean_age <- as.integer(mean(copy_data$age, na.rm = TRUE))
cat("Mean age (calculated): ", mean_age, "\n")

copy_data$age[is.na(copy_data$age)] <- mean_age
cat("Missing values in 'age' column after replacement: ",
  sum(is.na(copy_data$age)), "\n")
```

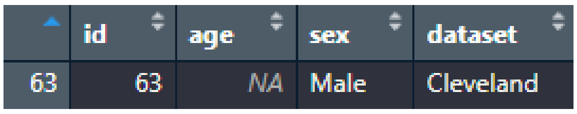
Code 9: R Script for Imputing Missing 'age' Values with the Mean

Output:

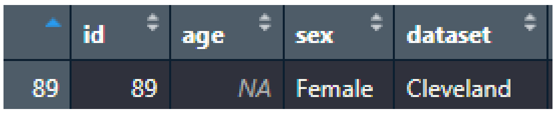


```
Rows with missing values in 'age' column:  63 89
```

Output 9.1: Rows with Missing Values in 'age' Column Before Imputation

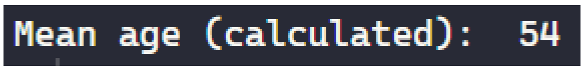


	id	age	sex	dataset
63	63	NA	Male	Cleveland



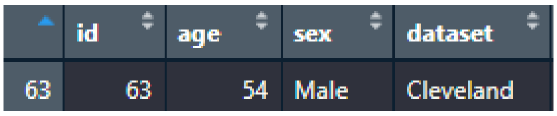
	id	age	sex	dataset
89	89	NA	Female	Cleveland

Output 9.2: Rows 63 and 89 with Missing Values in 'age' Column Before Imputation

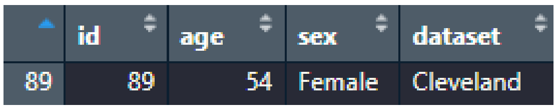


```
Mean age (calculated):  54
```

Output 9.3: Calculated Mean Value for 'age' Imputation



	id	age	sex	dataset
63	63	54	Male	Cleveland



	id	age	sex	dataset
89	89	54	Female	Cleveland

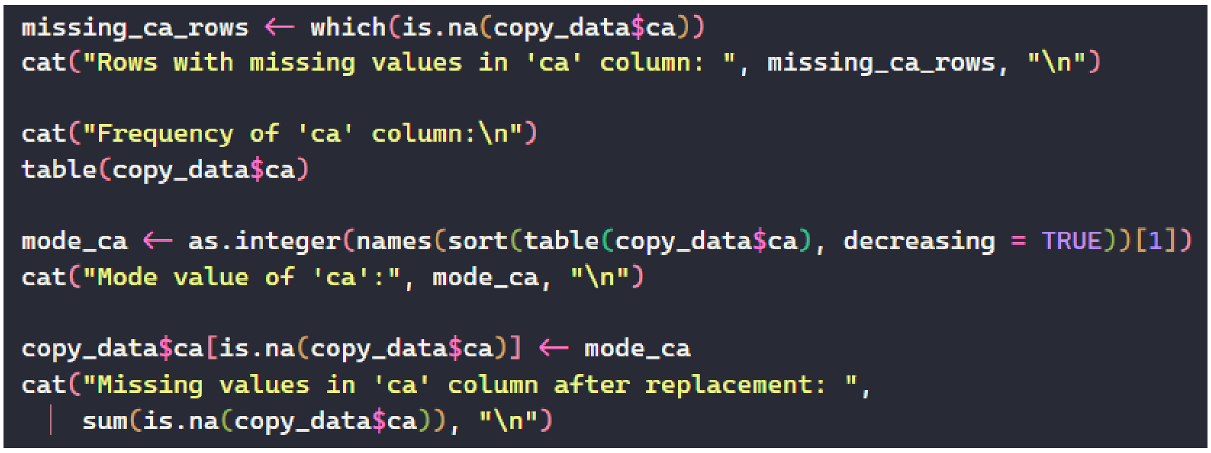
Output 9.4: Rows 63 and 89 with Missing Values in 'age' Column After Imputation

Description:

The rows containing missing values in the age column were identified. The mean of the non-missing age values was calculated and rounded. This mean value was then used to replace the missing entries. After this process, all missing values in the age column were successfully handled.

8 Missing ca Values Replaced with Mode

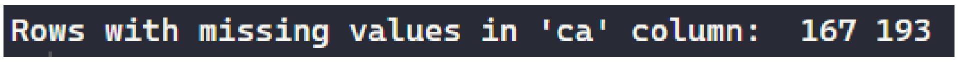
Code:



```r
missing_ca_rows <- which(is.na(copy_data$ca))
cat("Rows with missing values in 'ca' column: ", missing_ca_rows, "\n")

cat("Frequency of 'ca' column:\n")
table(copy_data$ca)

mode_ca <- as.integer(names(sort(table(copy_data$ca), decreasing = TRUE))[1])
cat("Mode value of 'ca':", mode_ca, "\n")

copy_data$ca[is.na(copy_data$ca)] <- mode_ca
cat("Missing values in 'ca' column after replacement: ",
    sum(is.na(copy_data$ca)), "\n")
```

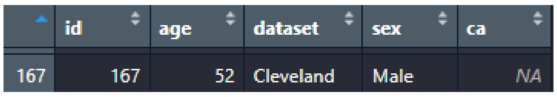
Code 10: R Script for Imputing Missing ca Values with the Mode

Output:

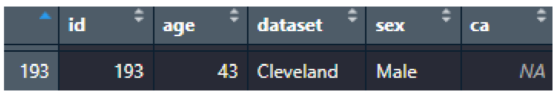


```
Rows with missing values in 'ca' column:  167 193
```

Output 10.1: Loading and Viewing the Dataset in R

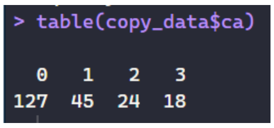


	id	age	dataset	sex	ca
167	167	52	Cleveland	Male	NA



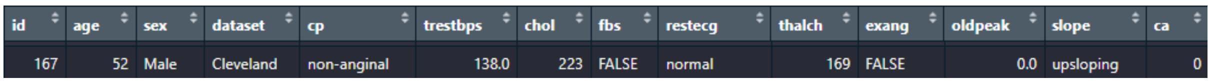
	id	age	dataset	sex	ca
193	193	43	Cleveland	Male	NA

Output 10.2: Rows with Missing Values in 'ca' Column Before Imputation



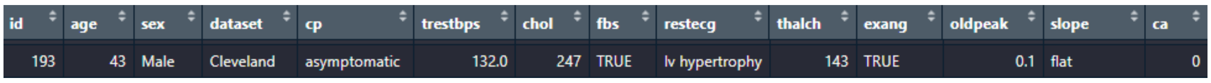
```
> table(copy_data$ca)

  0   1   2   3 
127  45  24  18 
```

Output 10.3: Frequency Distribution of 'ca' Column



id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca
167	52	Male	Cleveland	non-anginal	138.0	223	FALSE	normal	169	FALSE	0.0	upsloping	0

Output 10.4: Row 167 with Missing Values in 'ca' Column After Imputation



id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca
193	43	Male	Cleveland	asymptomatic	132.0	247	TRUE	lv hypertrophy	143	TRUE	0.1	flat	0

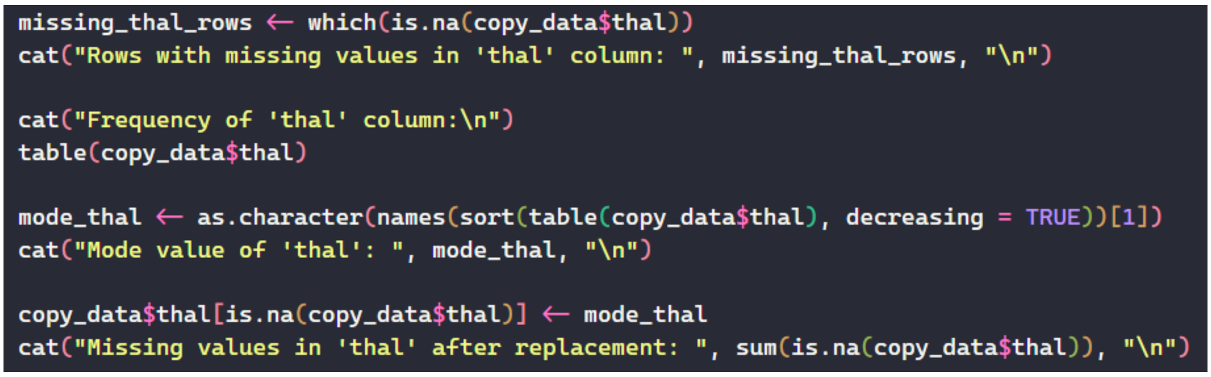
Output 10.5: Row 193 with Missing Values in 'ca' Column After Imputation

Description:

Rows with missing values in the ca column were detected. The frequency distribution of existing values was reviewed to find the most common value (mode). This mode was selected and used to fill in the missing entries. Following the replacement, no missing values remained in the column.

9 Missing thal Values Replaced with Mode

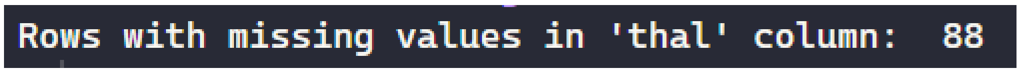
Code:



```r
missing_thal_rows <- which(is.na(copy_data$thal))
cat("Rows with missing values in 'thal' column: ", missing_thal_rows, "\n")

cat("Frequency of 'thal' column:\n")
table(copy_data$thal)

mode_thal <- as.character(names(sort(table(copy_data$thal), decreasing = TRUE))[1])
cat("Mode value of 'thal': ", mode_thal, "\n")

copy_data$thal[is.na(copy_data$thal)] <- mode_thal
cat("Missing values in 'thal' after replacement: ", sum(is.na(copy_data$thal)), "\n")
```

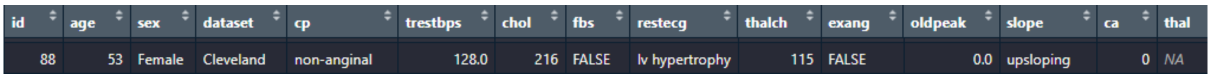
Code 11: R Script for Imputing Missing 'thal' Values with the Mode

Output:



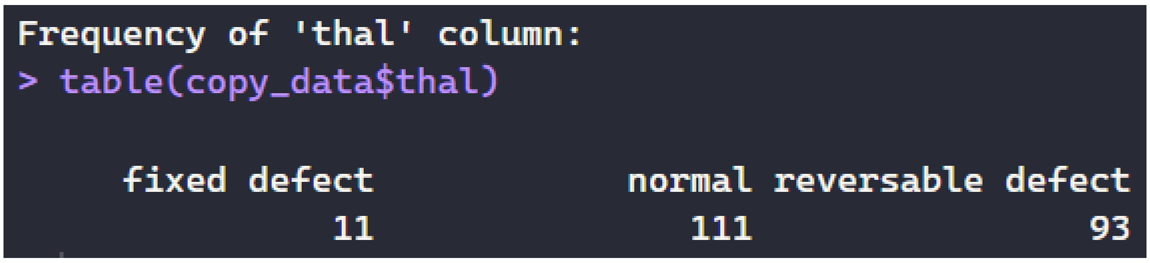
```
Rows with missing values in 'thal' column:  88
```

Output 11.1: Rows with Missing Values in 'thal' Column Before Imputation



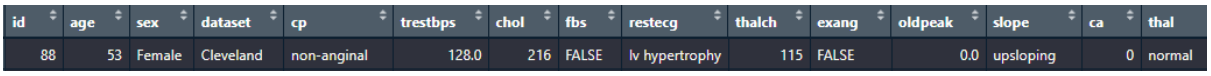
id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal
88	53	Female	Cleveland	non-anginal	128.0	216	FALSE	lv hypertrophy	115	FALSE	0.0	upsloping	0	NA

Output 11.2: Row 88 with Missing Values in 'thal' Column Before Imputation



```
Frequency of 'thal' column:
> table(copy_data$thal)

    fixed defect            normal reversable defect
              11               111                93
```

Output 11.3: Frequency Distribution of 'thal' Column



id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal
88	53	Female	Cleveland	non-anginal	128.0	216	FALSE	lv hypertrophy	115	FALSE	0.0	upsloping	0	normal

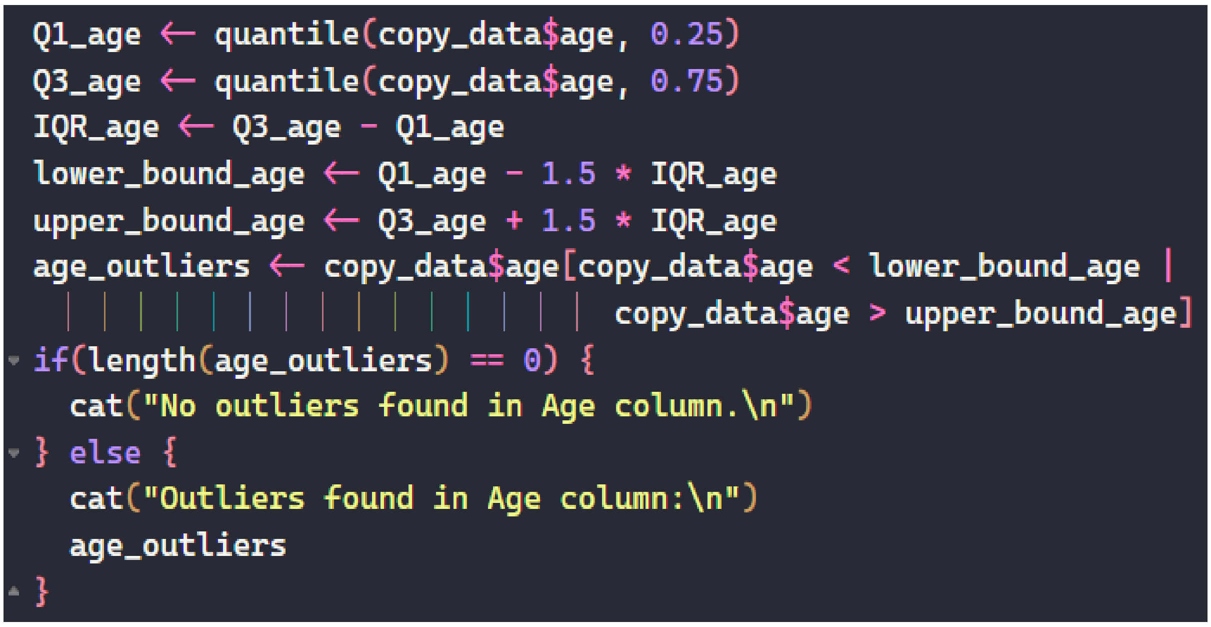
Output 11.4: Row 88 with Missing Values in 'thal' Column After Imputation

Description:

The rows with missing data in the thal column were identified. A frequency table was created to analyze the distribution of values, and the most frequent value (mode) was determined. This value was used to impute missing entries. As a result, all missing values in the thal column were successfully addressed.

10 Outlier Detection for age Column

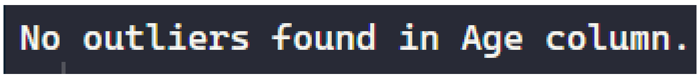
Code:



```r
Q1_age <- quantile(copy_data$age, 0.25)
Q3_age <- quantile(copy_data$age, 0.75)
IQR_age <- Q3_age - Q1_age
lower_bound_age <- Q1_age - 1.5 * IQR_age
upper_bound_age <- Q3_age + 1.5 * IQR_age
age_outliers <- copy_data$age[copy_data$age < lower_bound_age |
                              copy_data$age > upper_bound_age]
if(length(age_outliers) == 0) {
  cat("No outliers found in Age column.\n")
} else {
  cat("Outliers found in Age column:\n")
  age_outliers
}
```

Code 12: R Script for Outlier Detection in the 'age' Column using IQR.

Output:



```
No outliers found in Age column.
```

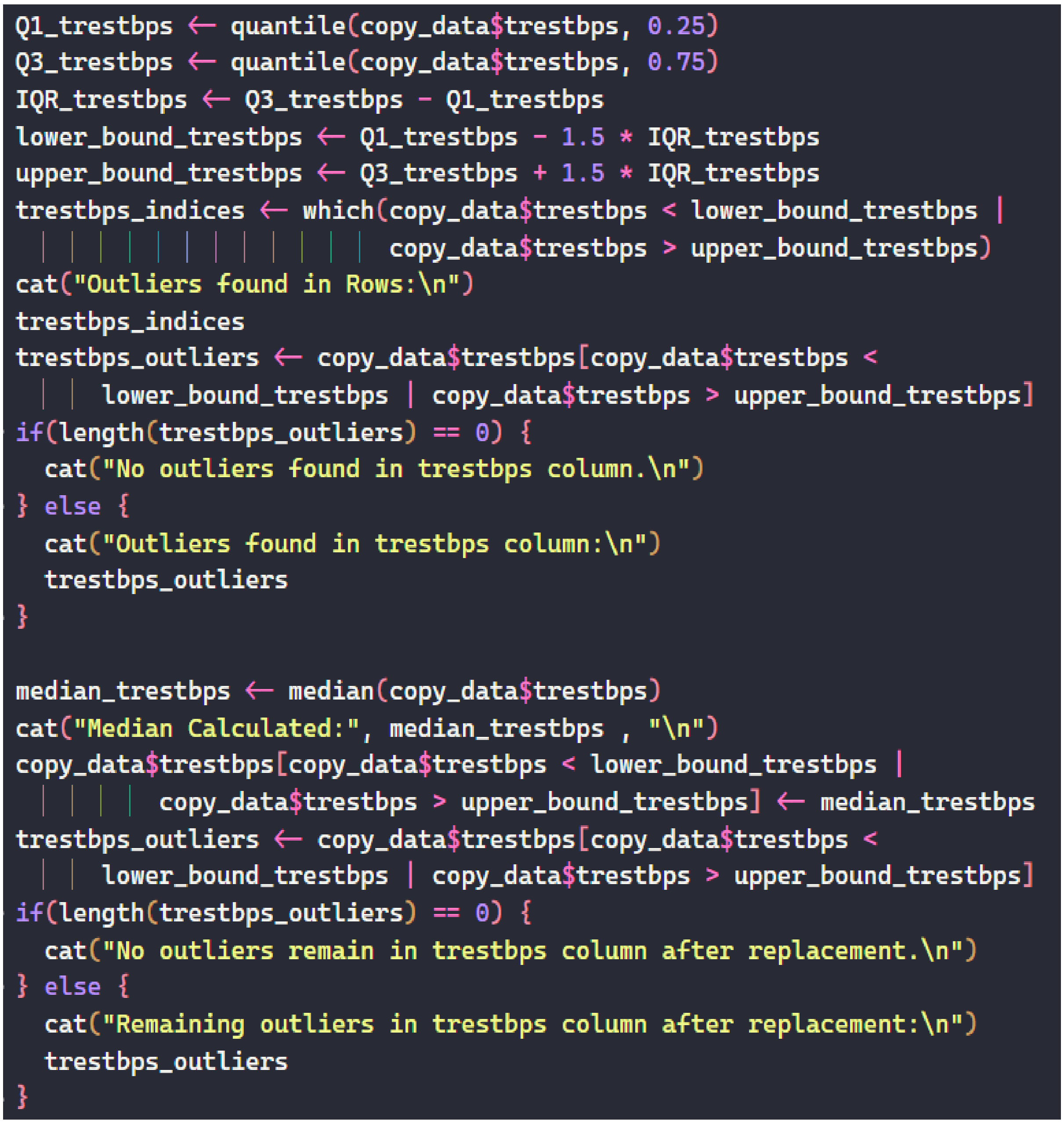
Output 12.1: Report on Outliers in 'age' Column using IQR.

Description:

The age column is checked for outliers using the Interquartile Range (IQR) method. The first (Q1) and third quartiles (Q3) are calculated, and the IQR is determined as the difference between Q3 and Q1. Outliers are defined as any value outside the range of Q1 - 1.5 * IQR and Q3 + 1.5 * IQR. If any outliers are identified in the age column, they are displayed. If no outliers are found, it is reported that no outliers exist in the age column.

11 Outlier Detection in trestbps Column

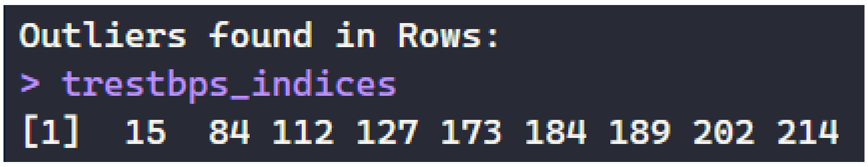
Code:



```r
Q1_trestbps <- quantile(copy_data$trestbps, 0.25)
Q3_trestbps <- quantile(copy_data$trestbps, 0.75)
IQR_trestbps <- Q3_trestbps - Q1_trestbps
lower_bound_trestbps <- Q1_trestbps - 1.5 * IQR_trestbps
upper_bound_trestbps <- Q3_trestbps + 1.5 * IQR_trestbps
trestbps_indices <- which(copy_data$trestbps < lower_bound_trestbps |
                          copy_data$trestbps > upper_bound_trestbps)
cat("Outliers found in Rows:\n")
trestbps_indices
trestbps_outliers <- copy_data$trestbps[copy_data$trestbps <
      lower_bound_trestbps | copy_data$trestbps > upper_bound_trestbps]
if(length(trestbps_outliers) == 0) {
  cat("No outliers found in trestbps column.\n")
} else {
  cat("Outliers found in trestbps column:\n")
  trestbps_outliers
}

median_trestbps <- median(copy_data$trestbps)
cat("Median Calculated:", median_trestbps , "\n")
copy_data$trestbps[copy_data$trestbps < lower_bound_trestbps |
          copy_data$trestbps > upper_bound_trestbps] <- median_trestbps
trestbps_outliers <- copy_data$trestbps[copy_data$trestbps <
      lower_bound_trestbps | copy_data$trestbps > upper_bound_trestbps]
if(length(trestbps_outliers) == 0) {
  cat("No outliers remain in trestbps column after replacement.\n")
} else {
  cat("Remaining outliers in trestbps column after replacement:\n")
  trestbps_outliers
}
```

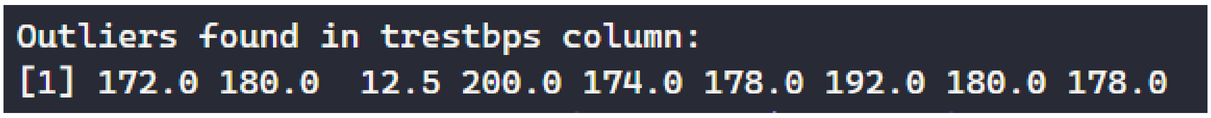
Code 13: Script for Outlier Detection and Handling in the 'trestbps' Column using IQR.

Output:



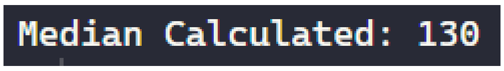
```
Outliers found in Rows:
> trestbps_indices
[1]  15  84 112 127 173 184 189 202 214
```

Output 13.1: Rows with Outliers in 'trestbps' Column



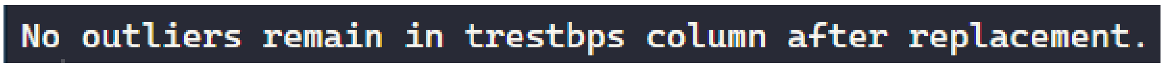
```
Outliers found in trestbps column:
[1] 172.0 180.0  12.5 200.0 174.0 178.0 192.0 180.0 178.0
```

Output 13.2: Outlier Values in 'trestbps' Column



```
Median Calculated: 130
```

Output 13.3: Calculated Median Value for 'trestbps' Outlier Replacement



```
No outliers remain in trestbps column after replacement.
```

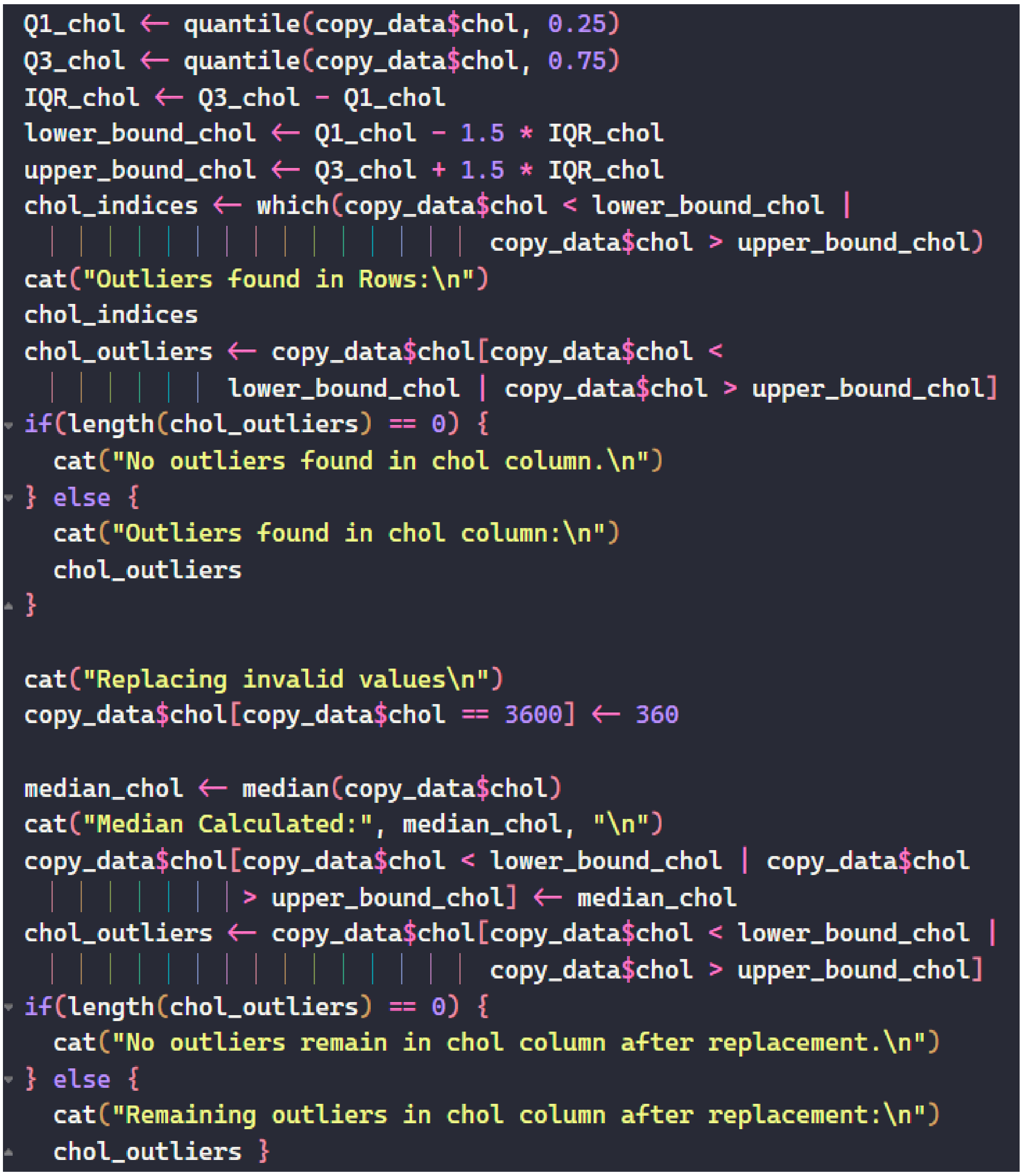
Output 13.4: Report on Remaining Outliers in 'trestbps' After Replacement

Description:

The trestbps (resting blood pressure) column is analyzed for outliers using the Interquartile Range (IQR) method. Values falling outside the lower and upper bounds are identified as outliers and displayed. If present, these outliers are replaced with the median value of the column. A final check confirms whether any outliers remain. In this case, no outliers were found in the trestbps column.

12 Outlier Detection in chol Column

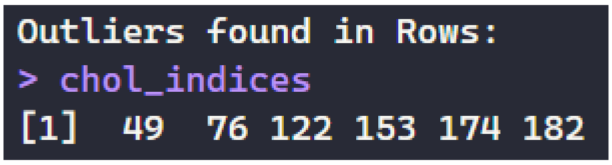
Code:



```r
Q1_chol <- quantile(copy_data$chol, 0.25)
Q3_chol <- quantile(copy_data$chol, 0.75)
IQR_chol <- Q3_chol - Q1_chol
lower_bound_chol <- Q1_chol - 1.5 * IQR_chol
upper_bound_chol <- Q3_chol + 1.5 * IQR_chol
chol_indices <- which(copy_data$chol < lower_bound_chol |
                                copy_data$chol > upper_bound_chol)
cat("Outliers found in Rows:\n")
chol_indices
chol_outliers <- copy_data$chol[copy_data$chol <
                     lower_bound_chol | copy_data$chol > upper_bound_chol]
if(length(chol_outliers) == 0) {
  cat("No outliers found in chol column.\n")
} else {
  cat("Outliers found in chol column:\n")
  chol_outliers
}

cat("Replacing invalid values\n")
copy_data$chol[copy_data$chol == 3600] <- 360

median_chol <- median(copy_data$chol)
cat("Median Calculated:", median_chol, "\n")
copy_data$chol[copy_data$chol < lower_bound_chol | copy_data$chol
                > upper_bound_chol] <- median_chol
chol_outliers <- copy_data$chol[copy_data$chol < lower_bound_chol |
                                copy_data$chol > upper_bound_chol]
if(length(chol_outliers) == 0) {
  cat("No outliers remain in chol column after replacement.\n")
} else {
  cat("Remaining outliers in chol column after replacement:\n")
  chol_outliers }
```

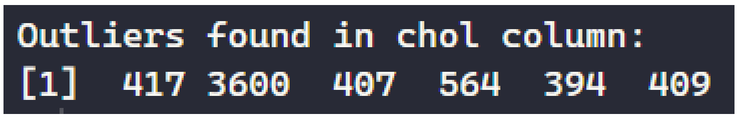
Code 14: R Script for Outlier Detection and Handling in the 'chol' Column using IQR.

Output:



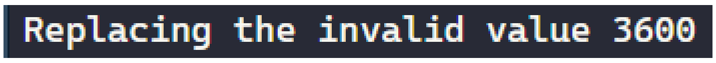
```
Outliers found in Rows:
> chol_indices
[1]  49  76 122 153 174 182
```

Output 14.1: Rows with Outliers in 'chol' Column



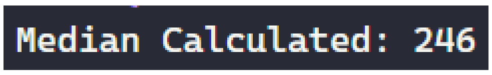
```
Outliers found in chol column:
[1]  417 3600  407  564  394  409
```

Output 14.2: Outlier Values in 'chol' Column



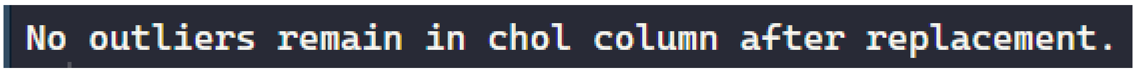
```
Replacing the invalid value 3600
```

Output 14.3: Report on Manual Replacement of Invalid 'chol' Value



```
Median Calculated: 246
```

Output 14.4: Calculated Median Value for 'chol' Outlier Replacement



```
No outliers remain in chol column after replacement.
```

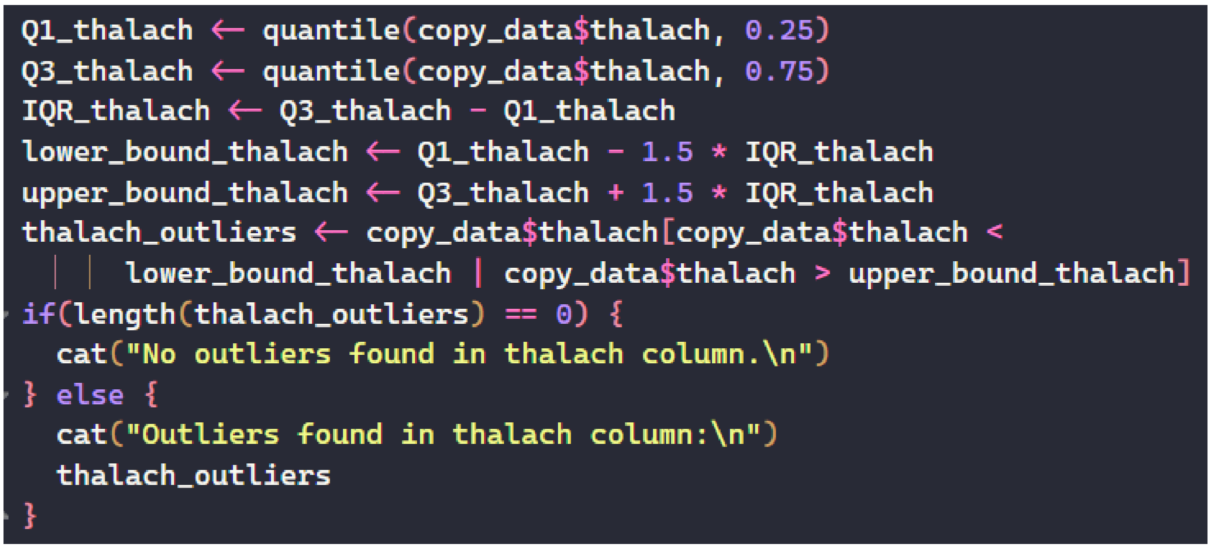
Output 14.5: Report on Remaining Outliers in 'chol' After Replacement

Description:

The chol column was analyzed for outliers using the IQR method. Outliers were identified at specific rows based on values falling outside the calculated bounds. One clearly invalid value, 3600, was manually identified and replaced with 360 before proceeding. Subsequently, all detected outliers were replaced with the median cholesterol value. After this replacement, a final check confirmed that no outliers remained in the chol column.

13 Outlier Detection in thalach Column

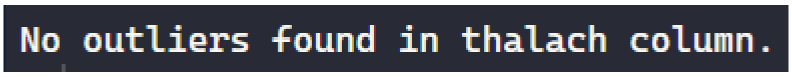
Code:



```
Q1_thalach <- quantile(copy_data$thalach, 0.25)
Q3_thalach <- quantile(copy_data$thalach, 0.75)
IQR_thalach <- Q3_thalach - Q1_thalach
lower_bound_thalach <- Q1_thalach - 1.5 * IQR_thalach
upper_bound_thalach <- Q3_thalach + 1.5 * IQR_thalach
thalach_outliers <- copy_data$thalach[copy_data$thalach <
      lower_bound_thalach | copy_data$thalach > upper_bound_thalach]
if(length(thalach_outliers) == 0) {
  cat("No outliers found in thalach column.\n")
} else {
  cat("Outliers found in thalach column:\n")
  thalach_outliers
}
```

Code 15: R Script for Outlier Detection in the 'thalach' Column using IQR.

Output:



```
No outliers found in thalach column.
```

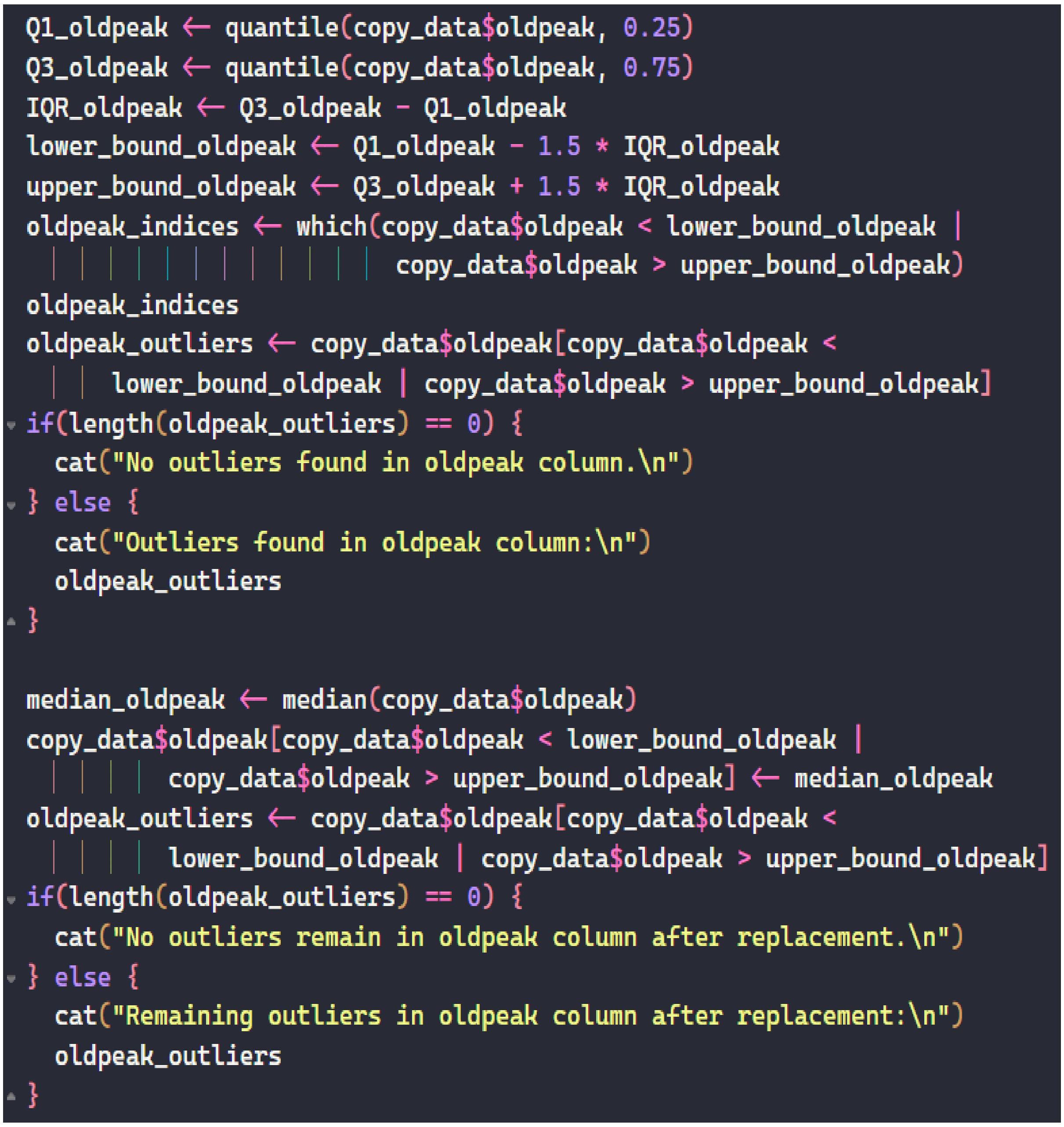
Output 15: Report on Outliers in 'thalach' Column using IQR.

Description:

The thalach column was examined for outliers using the IQR method. After calculating the lower and upper bounds, no values were found to lie outside this range. Therefore, no outliers were detected in the thalach column, and no replacements were necessary.

14 Outlier Detection in oldpeak Column

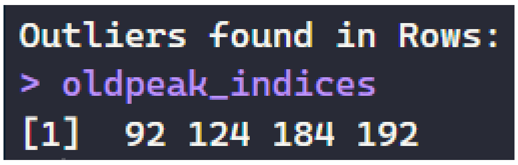
Code:



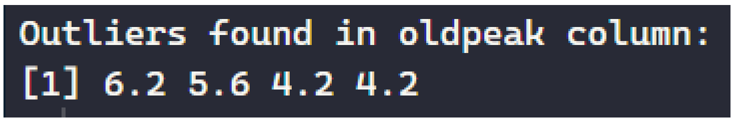
```r
Q1_oldpeak <- quantile(copy_data$oldpeak, 0.25)
Q3_oldpeak <- quantile(copy_data$oldpeak, 0.75)
IQR_oldpeak <- Q3_oldpeak - Q1_oldpeak
lower_bound_oldpeak <- Q1_oldpeak - 1.5 * IQR_oldpeak
upper_bound_oldpeak <- Q3_oldpeak + 1.5 * IQR_oldpeak
oldpeak_indices <- which(copy_data$oldpeak < lower_bound_oldpeak |
                          copy_data$oldpeak > upper_bound_oldpeak)
oldpeak_indices
oldpeak_outliers <- copy_data$oldpeak[copy_data$oldpeak <
      lower_bound_oldpeak | copy_data$oldpeak > upper_bound_oldpeak]
if(length(oldpeak_outliers) == 0) {
  cat("No outliers found in oldpeak column.\n")
} else {
  cat("Outliers found in oldpeak column:\n")
  oldpeak_outliers
}

median_oldpeak <- median(copy_data$oldpeak)
copy_data$oldpeak[copy_data$oldpeak < lower_bound_oldpeak |
          copy_data$oldpeak > upper_bound_oldpeak] <- median_oldpeak
oldpeak_outliers <- copy_data$oldpeak[copy_data$oldpeak <
          lower_bound_oldpeak | copy_data$oldpeak > upper_bound_oldpeak]
if(length(oldpeak_outliers) == 0) {
  cat("No outliers remain in oldpeak column after replacement.\n")
} else {
  cat("Remaining outliers in oldpeak column after replacement:\n")
  oldpeak_outliers
}
```

Code 16: Script for Outlier Detection and Handling in the 'oldpeak' Column using IQR.

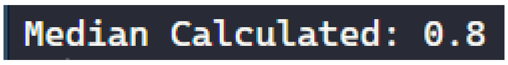
Output:


```
Outliers found in Rows:
> oldpeak_indices
[1]  92 124 184 192
```

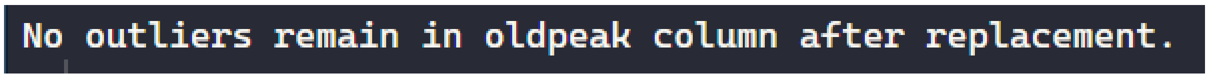
Output 16.1: Rows with Outliers in 'oldpeak' Column


```
Outliers found in oldpeak column:
[1] 6.2 5.6 4.2 4.2
```

Output 16.2: Outlier Values in 'oldpeak' Column


```
Median Calculated: 0.8
```

Output 16.3: Calculated Median Value for 'oldpeak' Outlier Replacement


```
No outliers remain in oldpeak column after replacement.
```

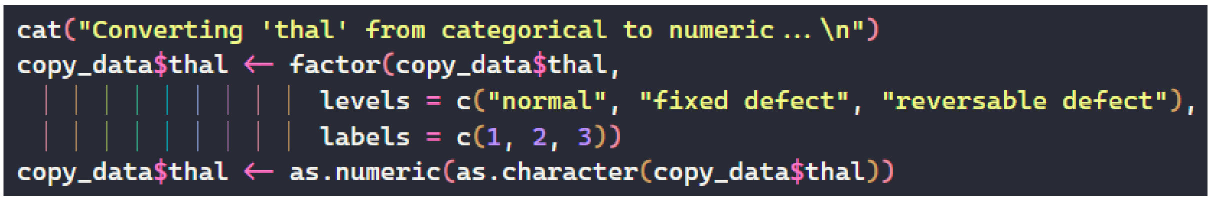
Output 16.4: Report on Remaining Outliers in 'oldpeak' After Replacement

Description:

The oldpeak column was analyzed for outliers using the IQR method. Several outliers were identified. To address this, the median value of the oldpeak column was calculated and used to replace all outlier values. After replacement, a final check confirmed that no outliers remained in the column.

15 Transformation of the thal Column

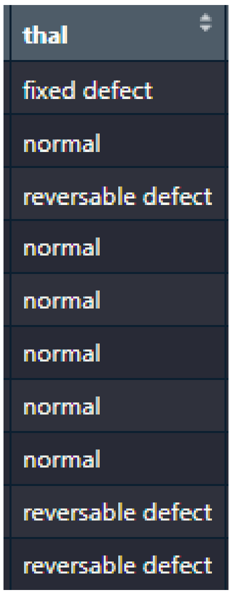
Code:



```r
cat("Converting 'thal' from categorical to numeric...\n")
copy_data$thal <- factor(copy_data$thal,
                         levels = c("normal", "fixed defect", "reversable defect"),
                         labels = c(1, 2, 3))
copy_data$thal <- as.numeric(as.character(copy_data$thal))
```

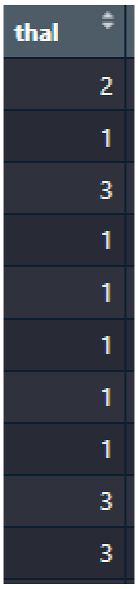
Code 17: R Script for Transforming the 'thal' Column to Numerical Representation

Output:



thal
fixed defect
normal
reversable defect
normal
normal
normal
normal
normal
reversable defect
reversable defect

Output 17.1: Display of the Transformed 'thal' Column (Before)



thal
2
1
3
1
1
1
1
1
3
3

Output 17.2: Display of the Transformed 'thal' Column (After)

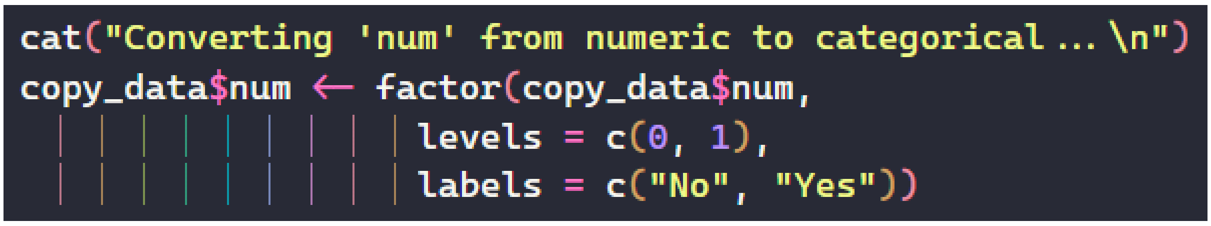
Description:

The thal column, which originally contained textual categories such as *"normal"*, *"fixed defect"*, and *"reversable defect"*, was converted into numerical format. Each category was assigned a corresponding numeric label as follows:

CATEGORIES	MAPPED TO
"normal"	**1**
"fixed defect"	**2**
"reversable defect"	**3**

16 Transformation of the num Column

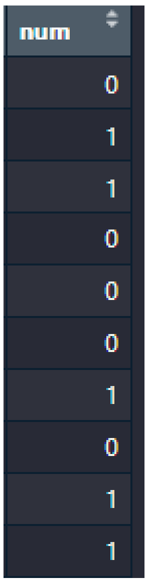
Code:



```r
cat("Converting 'num' from numeric to categorical...\n")
copy_data$num <- factor(copy_data$num,
                        levels = c(0, 1),
                        labels = c("No", "Yes"))
```

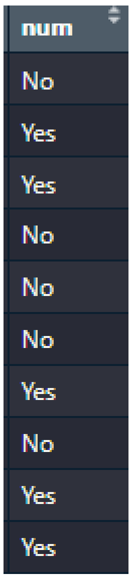
Code 18: R Script for Transforming the 'num' Column to Categorical 'diagnosis'

Output:



```
num
  0
  1
  1
  0
  0
  0
  1
  0
  1
  1
```

Output 18.1: Display of the Transformed 'diagnosis' Column (Before)



```
num
No
Yes
Yes
No
No
No
Yes
No
Yes
Yes
```

Output 18.2: Display of the Transformed 'diagnosis' Column (After)

Description:

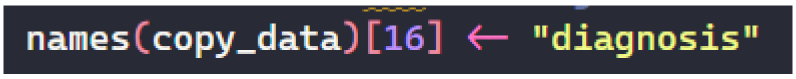
The num column, representing the presence or absence of heart disease, was originally numeric with values 0 and 1. To improve clarity and readability, it was transformed into a categorical variable with descriptive labels:

0 → "No" (indicating no heart disease)   1 → "Yes" (indicating presence of heart disease)

NUMBER	MAPPED TO
0	**“No”**
1	**“Yes”**

17. Renaming the num Column to diagnosis

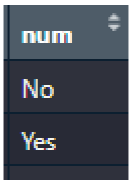
Code:



```
names(copy_data)[16] <- "diagnosis"
```

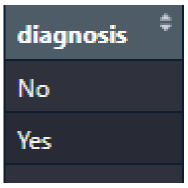
Code 19: R Script for Transforming the 'num' Column to Categorical 'diagnosis'

Output:



num
No
Yes

Output 19.1: Column Renaming (Before)



diagnosis
No
Yes

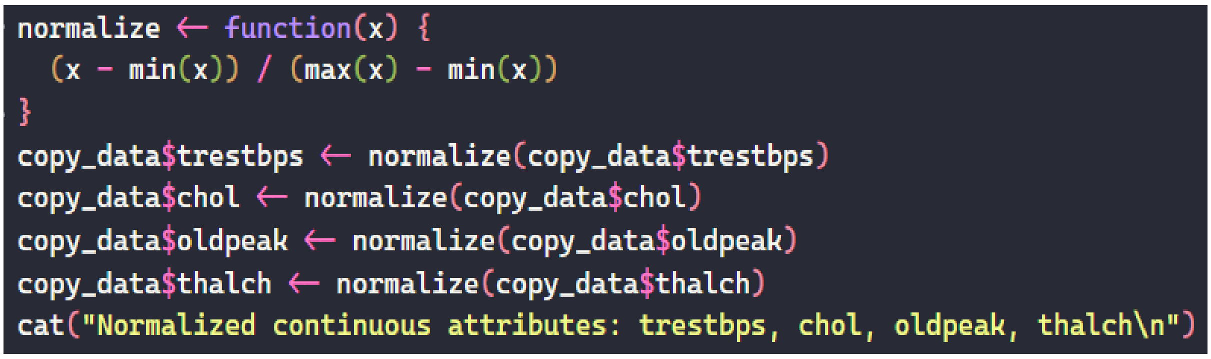
Output 19.2: Column Renaming (After)

Description:

To enhance the clarity of the dataset, the column previously named num was renamed to diagnosis. This new label more accurately reflects the column's purpose—indicating whether a patient is diagnosed with heart disease ("Yes") or not ("No").

18. Normalization of Continuous Attributes

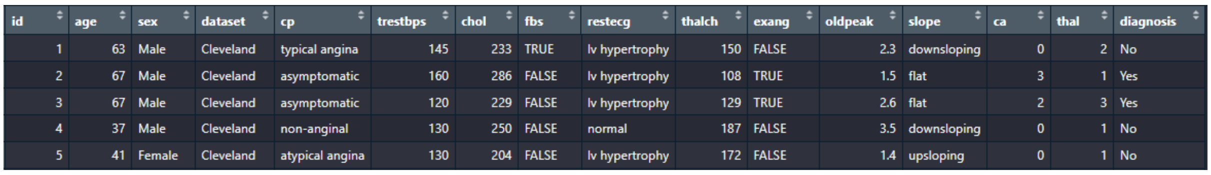
Code:



```r
normalize <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
copy_data$trestbps <- normalize(copy_data$trestbps)
copy_data$chol <- normalize(copy_data$chol)
copy_data$oldpeak <- normalize(copy_data$oldpeak)
copy_data$thalch <- normalize(copy_data$thalch)
cat("Normalized continuous attributes: trestbps, chol, oldpeak, thalch\n")
```

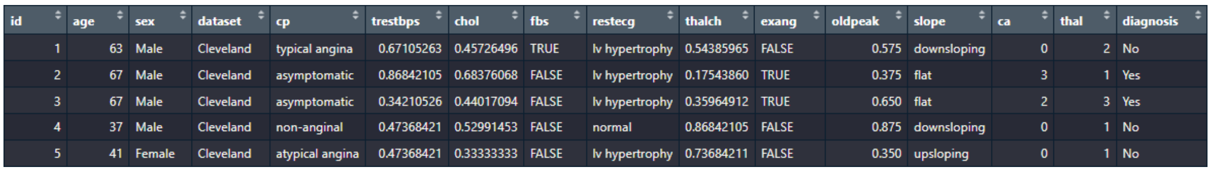
Code 20: R Script for Normalizing Continuous Attributes

Output:



id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	diagnosis
1	63	Male	Cleveland	typical angina	145	233	TRUE	lv hypertrophy	150	FALSE	2.3	downsloping	0	2	No
2	67	Male	Cleveland	asymptomatic	160	286	FALSE	lv hypertrophy	108	TRUE	1.5	flat	3	1	Yes
3	67	Male	Cleveland	asymptomatic	120	229	FALSE	lv hypertrophy	129	TRUE	2.6	flat	2	3	Yes
4	37	Male	Cleveland	non-anginal	130	250	FALSE	normal	187	FALSE	3.5	downsloping	0	1	No
5	41	Female	Cleveland	atypical angina	130	204	FALSE	lv hypertrophy	172	FALSE	1.4	upsloping	0	1	No

Output 20.1: Display of Normalized Continuous Attributes (Before)



id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	diagnosis
1	63	Male	Cleveland	typical angina	0.67105263	0.45726496	TRUE	lv hypertrophy	0.54385965	FALSE	0.575	downsloping	0	2	No
2	67	Male	Cleveland	asymptomatic	0.86842105	0.68376068	FALSE	lv hypertrophy	0.17543860	TRUE	0.375	flat	3	1	Yes
3	67	Male	Cleveland	asymptomatic	0.34210526	0.44017094	FALSE	lv hypertrophy	0.35964912	TRUE	0.650	flat	2	3	Yes
4	37	Male	Cleveland	non-anginal	0.47368421	0.52991453	FALSE	normal	0.86842105	FALSE	0.875	downsloping	0	1	No
5	41	Female	Cleveland	atypical angina	0.47368421	0.33333333	FALSE	lv hypertrophy	0.73684211	FALSE	0.350	upsloping	0	1	No

Output 20.2: Display of Normalized Continuous Attributes (After)

Description:

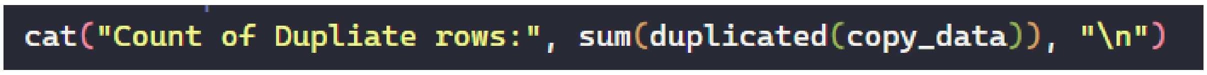
To bring continuous attributes to a common scale, the normalize() function was applied. This function rescales the values of each selected column to a range between 0 and 1.

The columns normalized include:

- trestbps
- chol
- oldpeak
- thalch

19. Counting Duplicate Rows

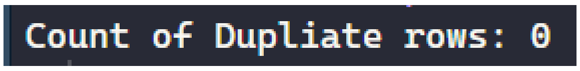
Code:



```r
cat("Count of Dupliate rows:", sum(duplicated(copy_data)), "\n")
```

Code 21: R Script for Counting Duplicate Rows in the Dataset

Output:



```
Count of Dupliate rows: 0
```

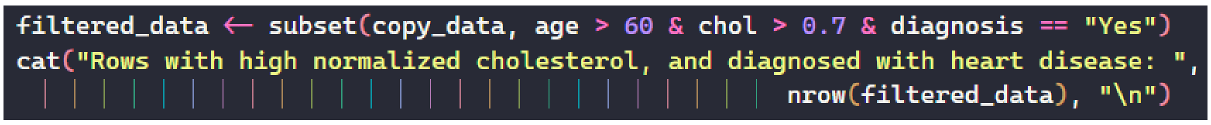
Output 21: Total Count of Duplicate Rows in the Dataset

Description:

To ensure the dataset does not contain redundant information, the code checks for duplicate rows using the duplicated() function. It then calculates the total count of duplicate entries in the dataset.

20. Filtering Data

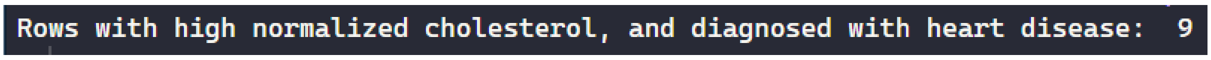
Code:



```r
filtered_data <- subset(copy_data, age > 60 & chol > 0.7 & diagnosis == "Yes")
cat("Rows with high normalized cholesterol, and diagnosed with heart disease: ",
                                                               nrow(filtered_data), "\n")
```

Code 22: R Script for Filtering the Dataset Based on Specific Criteria

Output:



```
Rows with high normalized cholesterol, and diagnosed with heart disease:  9
```

Output 22: Display of the Filtered Subset of the Dataset

Description:

The dataset is filtered based on specific conditions to narrow down the data to rows that meet criteria. In this case, the filtering conditions are:

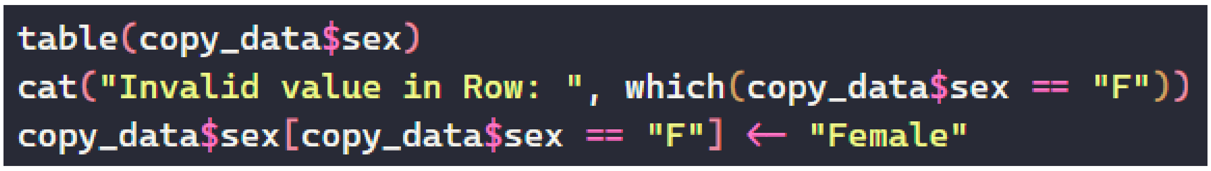
1. Age greater than 60.

2. Normalized cholesterol levels are greater than 0.7.

3. Diagnosis equals "Yes" (indicating a positive diagnosis for heart disease).

21. Fixing Invalid Data

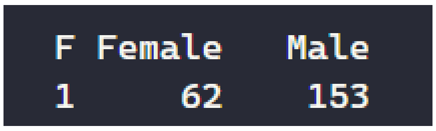
Code:



```
table(copy_data$sex)
cat("Invalid value in Row: ", which(copy_data$sex == "F"))
copy_data$sex[copy_data$sex == "F"] ← "Female"
```

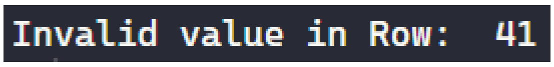
Code 23: Script for Identifying and Correcting Invalid Data in the 'sex' Column.

Output:



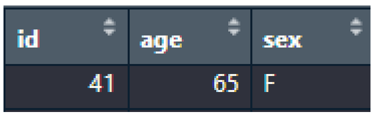
```
F Female    Male
1     62     153
```

Output 23.1: Table Showing Value Counts in the 'sex' Column.



```
Invalid value in Row:  41
```

Output 23.2: Row with Invalid Value



id	age	sex
41	65	F

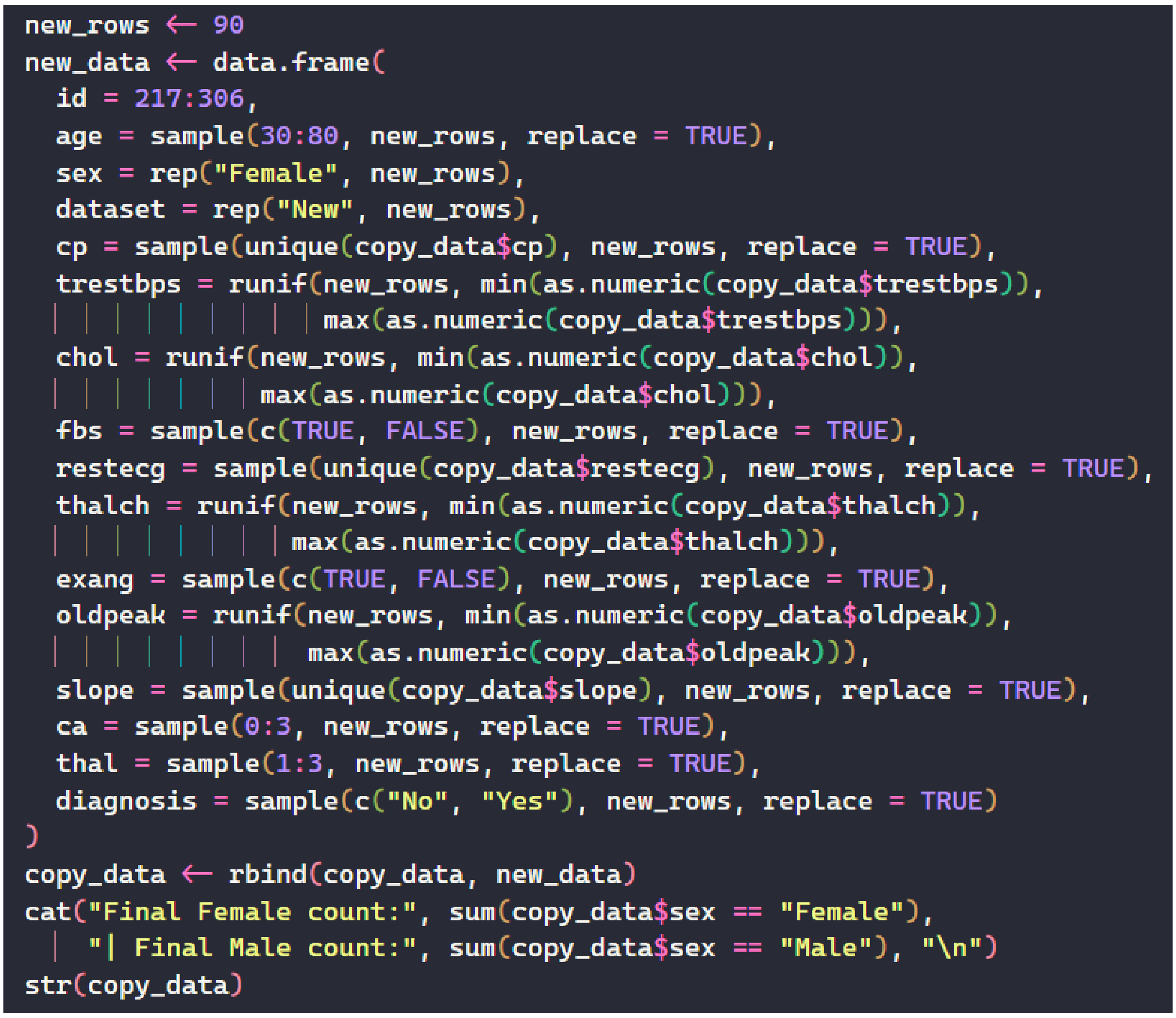
Output 23.3: Viewing the “F” in the Dataset

Description:

In this step, the dataset is checked for invalid entries in the sex column. Specifically, it looks for the invalid value "F" (which might have been mistakenly entered instead of "Female"). The table() function is used to view the distribution of values in the sex column. The which() function is used to locate the row index where the invalid entry "F" appears. The invalid value "F" is then replaced with the correct value "Female" using indexing.

22. Balancing the Dataset

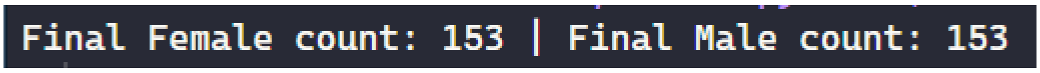
Code:



```r
new_rows <- 90
new_data <- data.frame(
  id = 217:306,
  age = sample(30:80, new_rows, replace = TRUE),
  sex = rep("Female", new_rows),
  dataset = rep("New", new_rows),
  cp = sample(unique(copy_data$cp), new_rows, replace = TRUE),
  trestbps = runif(new_rows, min(as.numeric(copy_data$trestbps)),
                   max(as.numeric(copy_data$trestbps))),
  chol = runif(new_rows, min(as.numeric(copy_data$chol)),
               max(as.numeric(copy_data$chol))),
  fbs = sample(c(TRUE, FALSE), new_rows, replace = TRUE),
  restecg = sample(unique(copy_data$restecg), new_rows, replace = TRUE),
  thalch = runif(new_rows, min(as.numeric(copy_data$thalch)),
                 max(as.numeric(copy_data$thalch))),
  exang = sample(c(TRUE, FALSE), new_rows, replace = TRUE),
  oldpeak = runif(new_rows, min(as.numeric(copy_data$oldpeak)),
                  max(as.numeric(copy_data$oldpeak))),
  slope = sample(unique(copy_data$slope), new_rows, replace = TRUE),
  ca = sample(0:3, new_rows, replace = TRUE),
  thal = sample(1:3, new_rows, replace = TRUE),
  diagnosis = sample(c("No", "Yes"), new_rows, replace = TRUE)
)
copy_data <- rbind(copy_data, new_data)
cat("Final Female count:", sum(copy_data$sex == "Female"),
    "| Final Male count:", sum(copy_data$sex == "Male"), "\n")
str(copy_data)
```

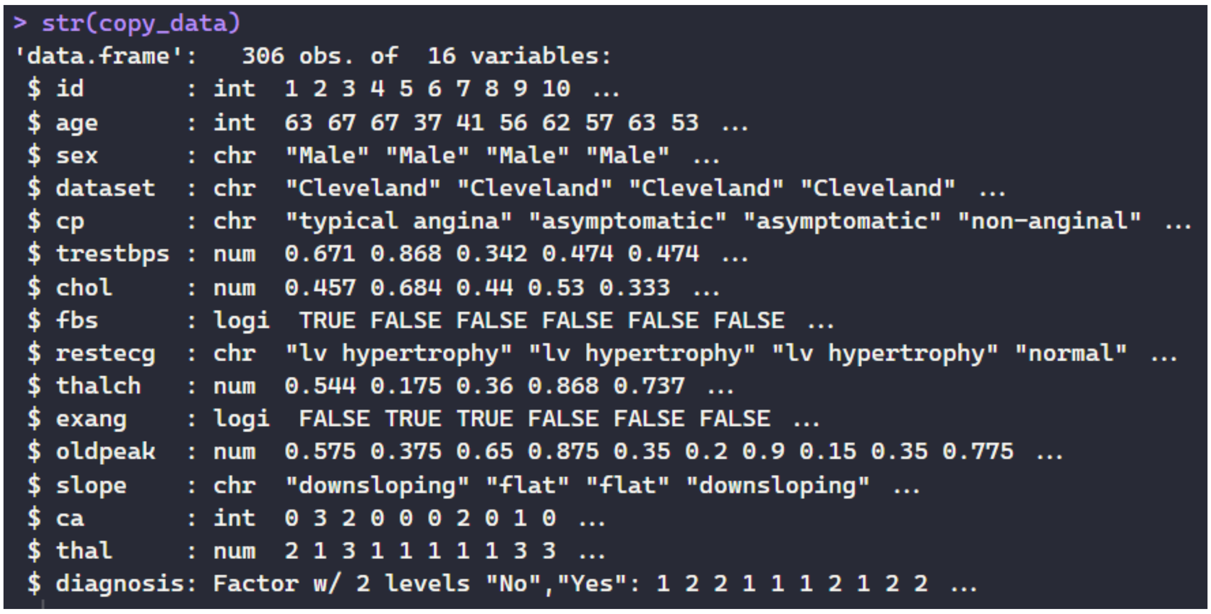
Code 24: R Script for Balancing the Dataset by Adding New Female Records

Output:



```
Final Female count: 153 | Final Male count: 153
```

Output 24.1: Value Counts in the 'sex' Column After Balancing the Dataset



```
> str(copy_data)
'data.frame':	306 obs. of  16 variables:
 $ id       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age      : int  63 67 67 37 41 56 62 57 63 53 ...
 $ sex      : chr  "Male" "Male" "Male" "Male" ...
 $ dataset  : chr  "Cleveland" "Cleveland" "Cleveland" "Cleveland" ...
 $ cp       : chr  "typical angina" "asymptomatic" "asymptomatic" "non-anginal" ...
 $ trestbps : num  0.671 0.868 0.342 0.474 0.474 ...
 $ chol     : num  0.457 0.684 0.44 0.53 0.333 ...
 $ fbs      : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
 $ restecg  : chr  "lv hypertrophy" "lv hypertrophy" "lv hypertrophy" "normal" ...
 $ thalch   : num  0.544 0.175 0.36 0.868 0.737 ...
 $ exang    : logi  FALSE TRUE TRUE FALSE FALSE FALSE ...
 $ oldpeak  : num  0.575 0.375 0.65 0.875 0.35 0.2 0.9 0.15 0.35 0.775 ...
 $ slope    : chr  "downsloping" "flat" "flat" "downsloping" ...
 $ ca       : int  0 3 2 0 0 0 2 0 1 0 ...
 $ thal     : num  2 1 3 1 1 1 1 1 3 3 ...
 $ diagnosis: Factor w/ 2 levels "No","Yes": 1 2 2 1 1 1 2 1 2 2 ...
```

Output 24.2: Value Counts in the 'sex' Column After Balancing the Dataset using str()

Description:

To address the gender imbalance in the dataset, where the number of female entries was significantly lower than that of male entries, we added 90 new female records to ensure more balanced data. The newly generated entries were designed by sampling values from the existing data, ensuring they reflected the distribution of key features. These additional female records were added with randomly generated values for continuous and categorical attributes, maintaining the same structure and attributes as the original dataset.

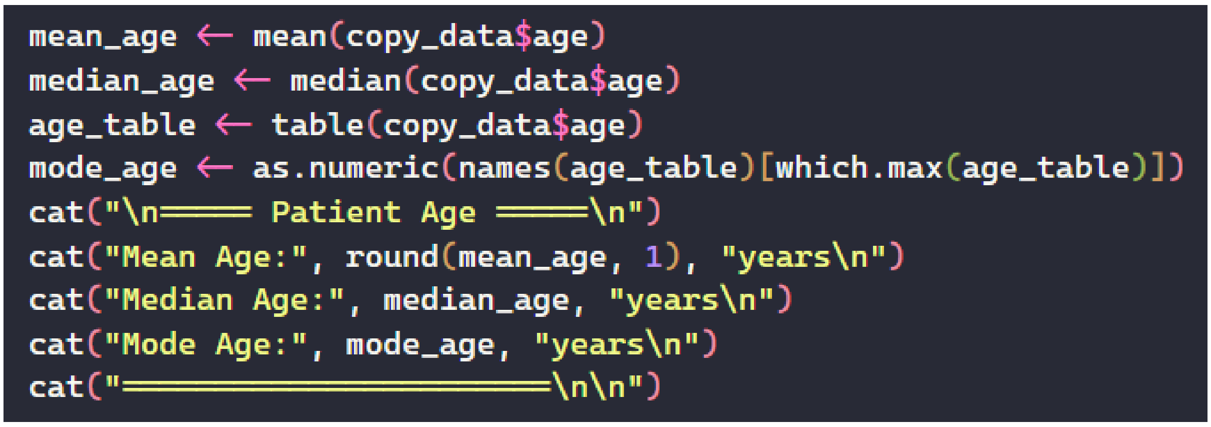
Here is a breakdown of the process:

1. New Data Generation: The new rows were created with:
   - Random ages between 30 and 80.
   - A constant "Female" value for the sex column to maintain the gender balance.
   - Randomized values for attributes such as cp, trestbps, chol, restecg, and more, ensuring the new records mirrored the distribution of existing values.
2. Dataset Update: After generating the new rows, they were appended to the original dataset using rbind().
3. Result Verification: The final count of male and female entries was verified to confirm the balance:

23. Analysis of Central Tendencies and Spread of Patient Data

23.1. age Column

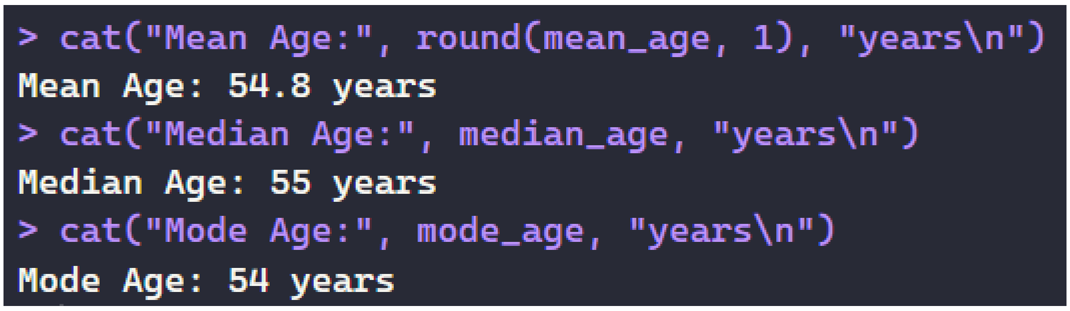
Code:



```r
mean_age <- mean(copy_data$age)
median_age <- median(copy_data$age)
age_table <- table(copy_data$age)
mode_age <- as.numeric(names(age_table)[which.max(age_table)])
cat("\n===== Patient Age =====\n")
cat("Mean Age:", round(mean_age, 1), "years\n")
cat("Median Age:", median_age, "years\n")
cat("Mode Age:", mode_age, "years\n")
cat("=======================\n\n")
```

Code 25: R Script for Calculating Central Tendency of the 'age' Column.

Output:



```
> cat("Mean Age:", round(mean_age, 1), "years\n")
Mean Age: 54.8 years
> cat("Median Age:", median_age, "years\n")
Median Age: 55 years
> cat("Mode Age:", mode_age, "years\n")
Mode Age: 54 years
```

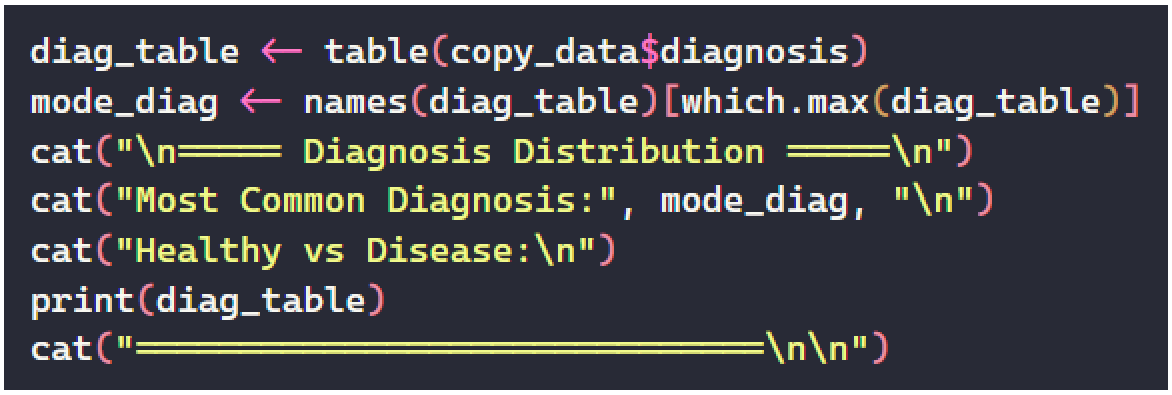
Output 25: Central Tendency Measures of the 'age' Column (Mean, Median, Mode)

Description:

The mean, median, and mode are almost identical, which indicates the patient age distribution is symmetric and not heavily skewed.
Most patients are middle-aged adults, around 54–55 years, showing the typical age group affected in the dataset.

23.2. diagnosis Column

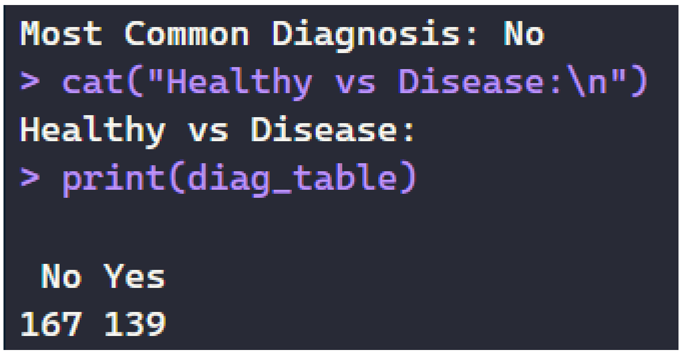
Code:



```r
diag_table <- table(copy_data$diagnosis)
mode_diag <- names(diag_table)[which.max(diag_table)]
cat("\n===== Diagnosis Distribution =====\n")
cat("Most Common Diagnosis:", mode_diag, "\n")
cat("Healthy vs Disease:\n")
print(diag_table)
cat("==============================\n\n")
```

Code 26: R Script for Analyzing the Distribution of the 'diagnosis' Variable.

Output:



```
Most Common Diagnosis: No
> cat("Healthy vs Disease:\n")
Healthy vs Disease:
> print(diag_table)

 No Yes
167 139
```

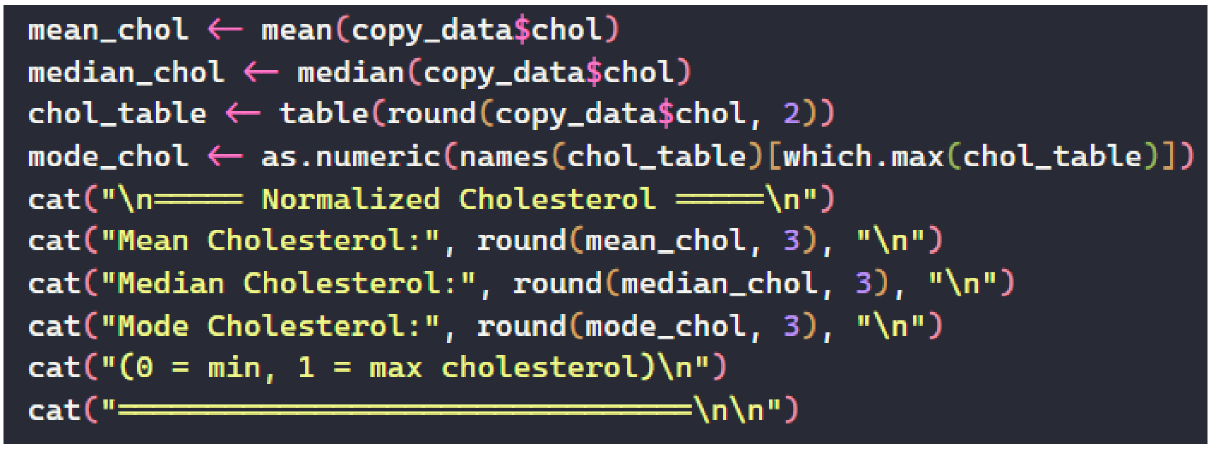
Output 26: Frequency Distribution of the 'diagnosis' Variable ("No", "Yes")

Description:

There are slightly more healthy patients than diseased patients, but the difference is not very large.
This suggests the dataset is balanced between patients with and without heart disease, which is important for fair model training in machine learning.

23.3. chol Column

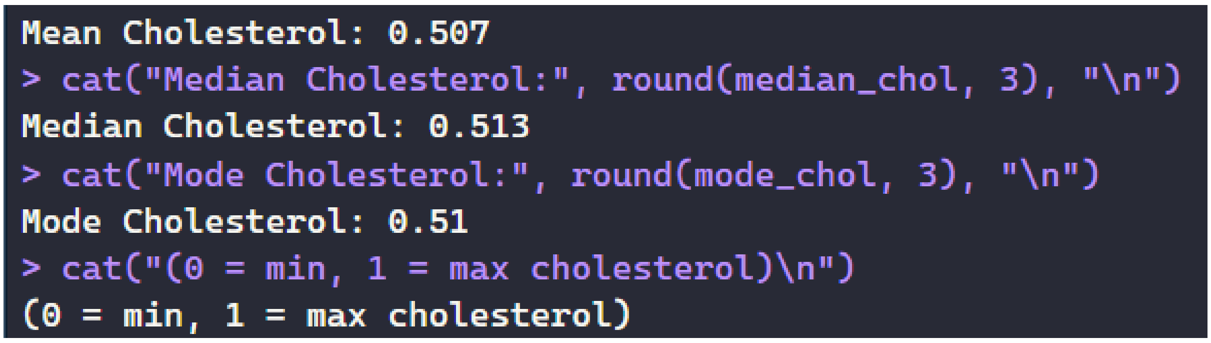
Code:



```r
mean_chol ← mean(copy_data$chol)
median_chol ← median(copy_data$chol)
chol_table ← table(round(copy_data$chol, 2))
mode_chol ← as.numeric(names(chol_table)[which.max(chol_table)])
cat("\n===== Normalized Cholesterol =====\n")
cat("Mean Cholesterol:", round(mean_chol, 3), "\n")
cat("Median Cholesterol:", round(median_chol, 3), "\n")
cat("Mode Cholesterol:", round(mode_chol, 3), "\n")
cat("(0 = min, 1 = max cholesterol)\n")
cat("================================\n\n")
```

Code 27: R Script for Calculating Central Tendency of the Normalized 'chol' Column.

Output:



```
Mean Cholesterol: 0.507
> cat("Median Cholesterol:", round(median_chol, 3), "\n")
Median Cholesterol: 0.513
> cat("Mode Cholesterol:", round(mode_chol, 3), "\n")
Mode Cholesterol: 0.51
> cat("(0 = min, 1 = max cholesterol)\n")
(0 = min, 1 = max cholesterol)
```

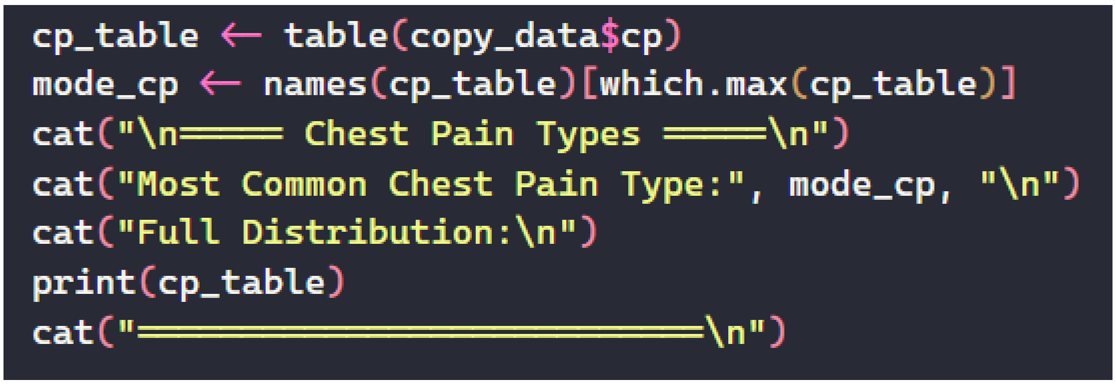
Output 27: Central Tendency Measures of the Normalized 'chol' Column (Mean, Median, Mode)

Description:

The cholesterol values are evenly spread around the center after normalization. Since the mean and median are close, the data is symmetrical. Most patients have moderate cholesterol levels, not extremely high or low after normalization.

23.4. cp Column

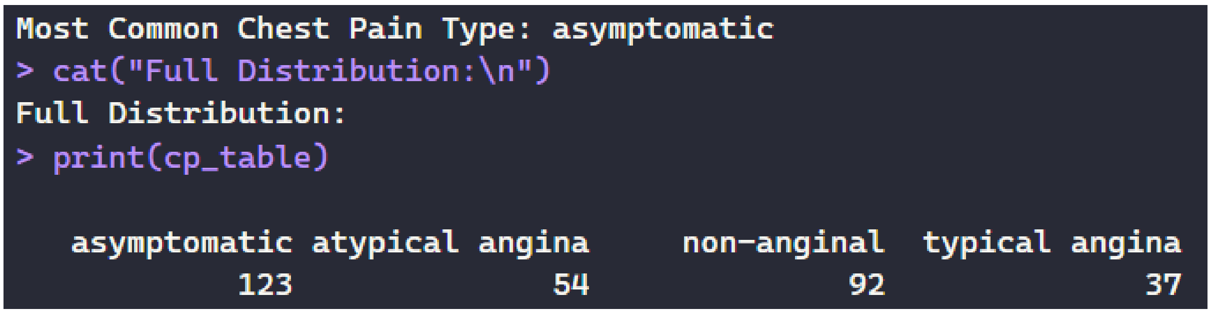
Code:



```r
cp_table <- table(copy_data$cp)
mode_cp <- names(cp_table)[which.max(cp_table)]
cat("\n===== Chest Pain Types =====\n")
cat("Most Common Chest Pain Type:", mode_cp, "\n")
cat("Full Distribution:\n")
print(cp_table)
cat("===========================\n")
```

Code 28: R Script for Analyzing the Distribution of the 'cp' Variable.

Output:



```
Most Common Chest Pain Type: asymptomatic
> cat("Full Distribution:\n")
Full Distribution:
> print(cp_table)

   asymptomatic atypical angina     non-anginal  typical angina 
            123              54              92              37 
```

Output 28: Frequency Distribution of the 'cp' Variable (Chest Pain Types)

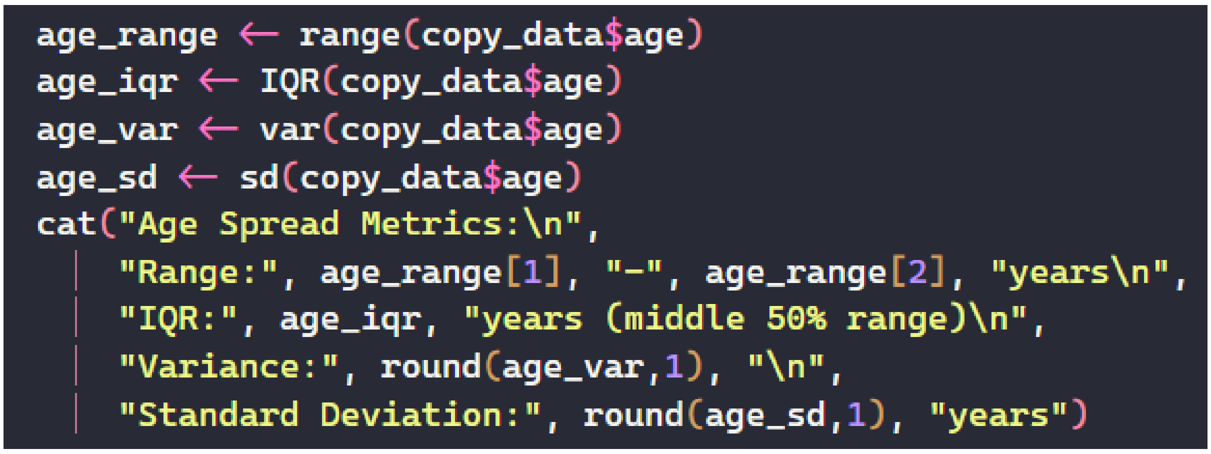
Description:

Most patients have asymptomatic chest pain (no noticeable symptoms). This is concerning because asymptomatic patients may have heart problems without feeling typical chest pain, making early detection more difficult. It highlights the importance of diagnostic testing, not just symptom-checking.

24. Spread Analysis for the Dataset

24.1. age Column

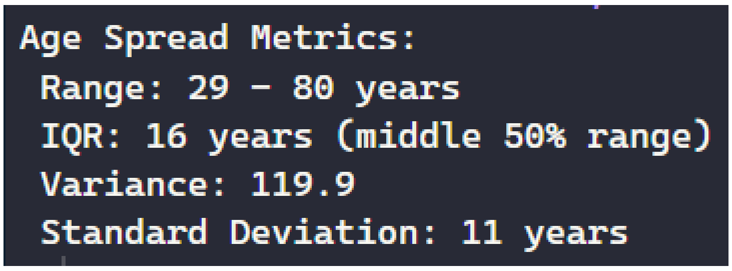
Code:



```r
age_range ← range(copy_data$age)
age_iqr ← IQR(copy_data$age)
age_var ← var(copy_data$age)
age_sd ← sd(copy_data$age)
cat("Age Spread Metrics:\n",
    "Range:", age_range[1], "-", age_range[2], "years\n",
    "IQR:", age_iqr, "years (middle 50% range)\n",
    "Variance:", round(age_var,1), "\n",
    "Standard Deviation:", round(age_sd,1), "years")
```

Code 29: R Script for Calculating Spread of the 'age' Column.

Output:



```
Age Spread Metrics:
 Range: 29 - 80 years
 IQR: 16 years (middle 50% range)
 Variance: 119.9
 Standard Deviation: 11 years
```

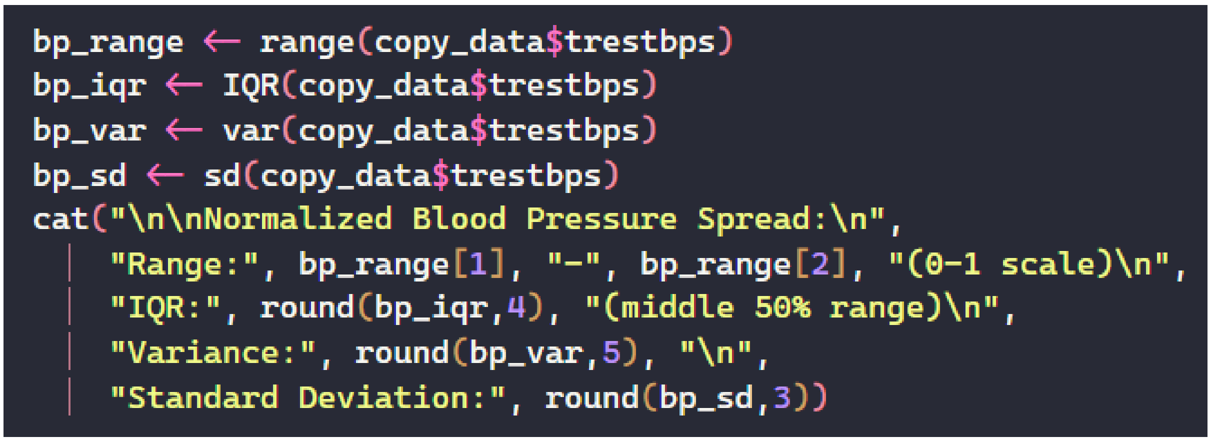
Output 29: Measures of Spread for the 'age' Column.

Description:

Patients' ages are widely distributed, from young adults (29) to elderly (80).
An IQR of 16 means the middle 50% of patients fall within a 16-year band around the median (roughly ages 47–63).
The standard deviation of 11 years indicates a moderate spread around the mean — age variability is present but not extreme.

24.2. trestbps Column

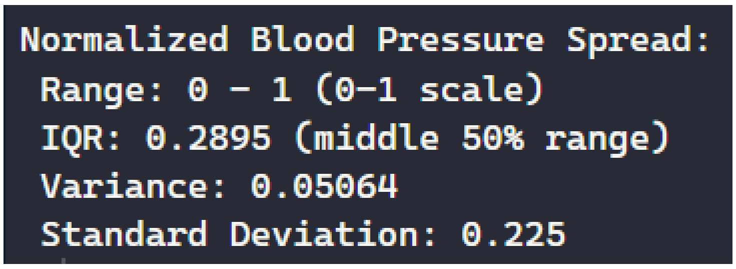
Code:



```
bp_range <- range(copy_data$trestbps)
bp_iqr <- IQR(copy_data$trestbps)
bp_var <- var(copy_data$trestbps)
bp_sd <- sd(copy_data$trestbps)
cat("\n\nNormalized Blood Pressure Spread:\n",
    "Range:", bp_range[1], "-", bp_range[2], "(0-1 scale)\n",
    "IQR:", round(bp_iqr,4), "(middle 50% range)\n",
    "Variance:", round(bp_var,5), "\n",
    "Standard Deviation:", round(bp_sd,3))
```

Code 30: R Script for Calculating Spread of the Normalized 'trestbps' Column.

Output:



```
Normalized Blood Pressure Spread:
 Range: 0 - 1 (0-1 scale)
 IQR: 0.2895 (middle 50% range)
 Variance: 0.05064
 Standard Deviation: 0.225
```

Output 30: Measures of Spread for the 'trestbps' Column.

Description:

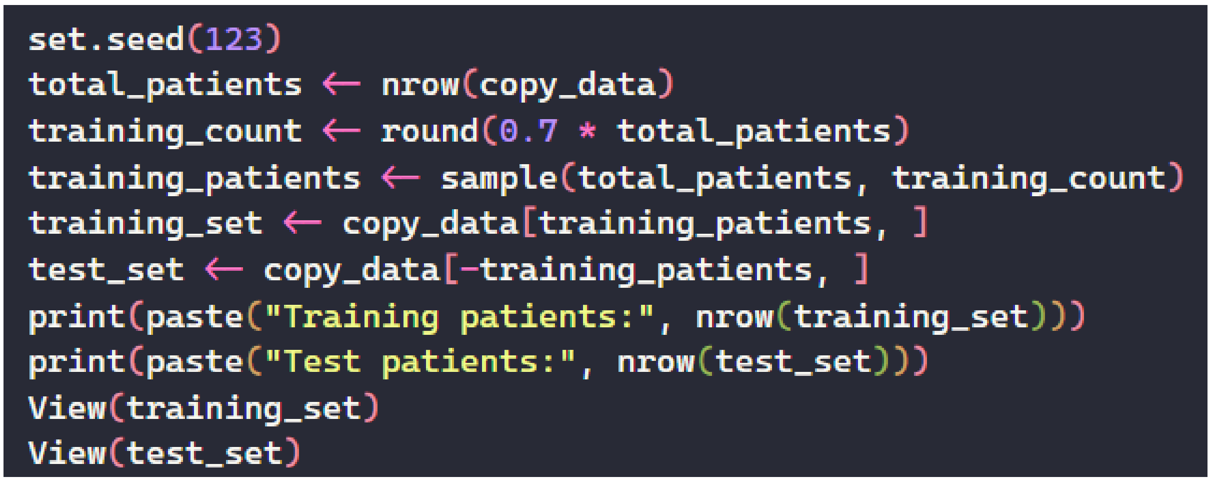
The normalized blood pressure covers the full scale (0–1), but most data points fall within a narrow middle range (IQR = 0.29).
A standard deviation of 0.225 shows that blood pressure values are relatively close to the center, meaning outliers (very high/very low pressures) are less common after normalization.

Final Overall Observation:

The age and cholesterol distributions are symmetric, which is good for modeling assumptions (normality). Most patients are middle-aged and asymptomatic, suggesting hidden risks of heart disease. The dataset is balanced between healthy and diseased cases, which supports fair predictive modeling. Variability in blood pressure and age exists but is moderate, with no extreme outliers dominating.

25. Data Splitting: Training and Testing Sets

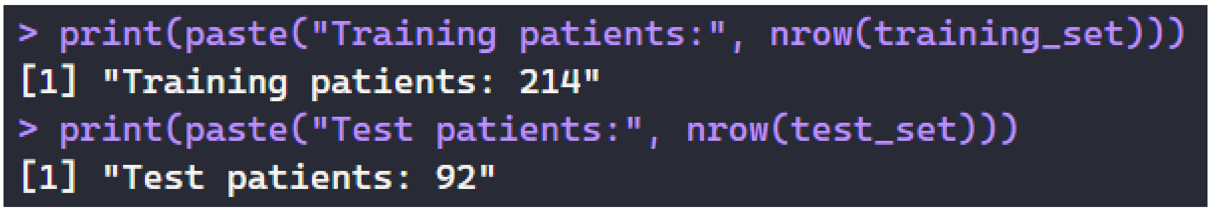
Code:



```r
set.seed(123)
total_patients <- nrow(copy_data)
training_count <- round(0.7 * total_patients)
training_patients <- sample(total_patients, training_count)
training_set <- copy_data[training_patients, ]
test_set <- copy_data[-training_patients, ]
print(paste("Training patients:", nrow(training_set)))
print(paste("Test patients:", nrow(test_set)))
View(training_set)
View(test_set)
```

Code 31: R Script for Splitting the Dataset into Training and Testing Sets

Output:



```
> print(paste("Training patients:", nrow(training_set)))
[1] "Training patients: 214"
> print(paste("Test patients:", nrow(test_set)))
[1] "Test patients: 92"
```

Output 31: Number of Records in the Training and Testing Datasets

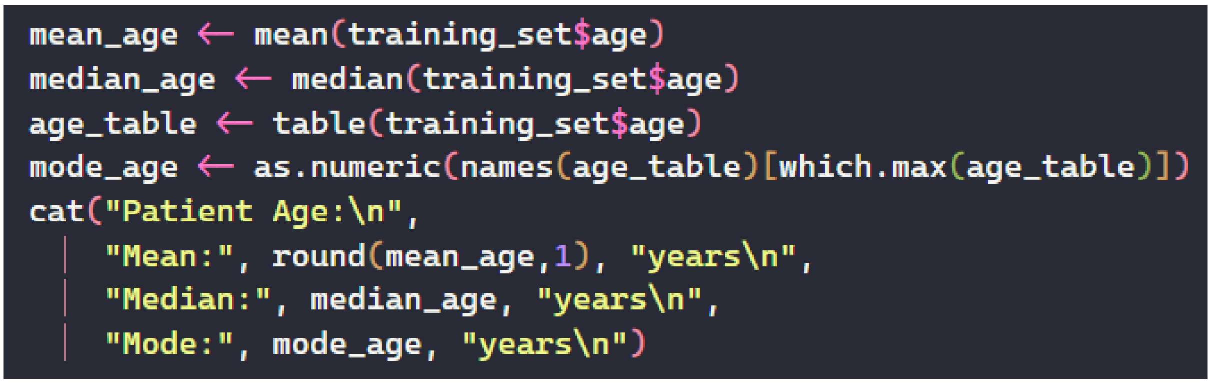
Description:

The dataset was divided into a training set and a testing set using a 70:30 ratio. A fixed random seed was used to maintain reproducibility. Out of 306 total patients, 214 were allocated to the training set to build the model, and 92 were kept in the testing set to evaluate its performance. The split was confirmed by checking the number of patients in each set.

26. **Central Tendency Analysis for the Training Set**

26.1. age Column

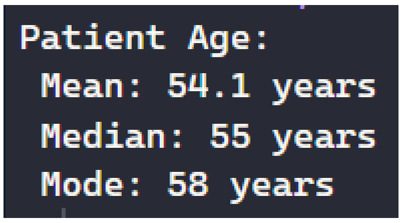
Code:



```r
mean_age <- mean(training_set$age)
median_age <- median(training_set$age)
age_table <- table(training_set$age)
mode_age <- as.numeric(names(age_table)[which.max(age_table)])
cat("Patient Age:\n",
    "Mean:", round(mean_age,1), "years\n",
    "Median:", median_age, "years\n",
    "Mode:", mode_age, "years\n")
```

Code 32: R Script for Calculating Central Tendency of 'age' in the Training Set

Output:



```
Patient Age:
 Mean: 54.1 years
 Median: 55 years
 Mode: 58 years
```

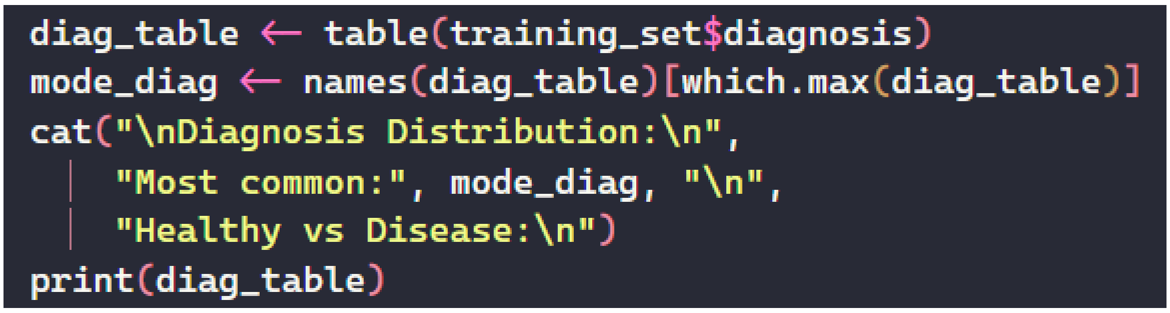
Output 32: Central Tendency Measures of 'age' in the Training Set

Description:

The average age of patients is 54.1 years, with a median of 55 years and a mode of 58 years. This indicates that most patients are middle-aged, and there is a slight skew towards older individuals. The distribution seems balanced around the mid-50s, suggesting a typical age range for the dataset.

26.2. diagnosis Column

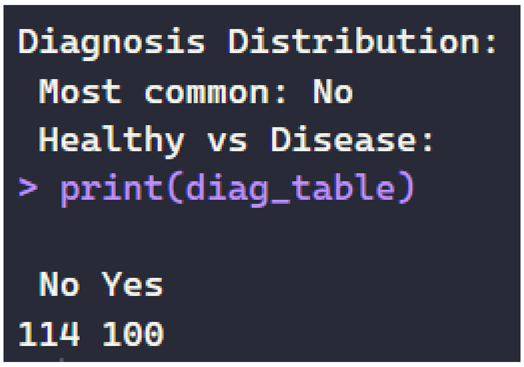
Code:



```r
diag_table <- table(training_set$diagnosis)
mode_diag <- names(diag_table)[which.max(diag_table)]
cat("\nDiagnosis Distribution:\n",
    "Most common:", mode_diag, "\n",
    "Healthy vs Disease:\n")
print(diag_table)
```

Code 33: R Script for Analyzing 'diagnosis' Distribution in the Training Set

Output:



```
Diagnosis Distribution:
 Most common: No
 Healthy vs Disease:
> print(diag_table)

 No Yes
114 100
```

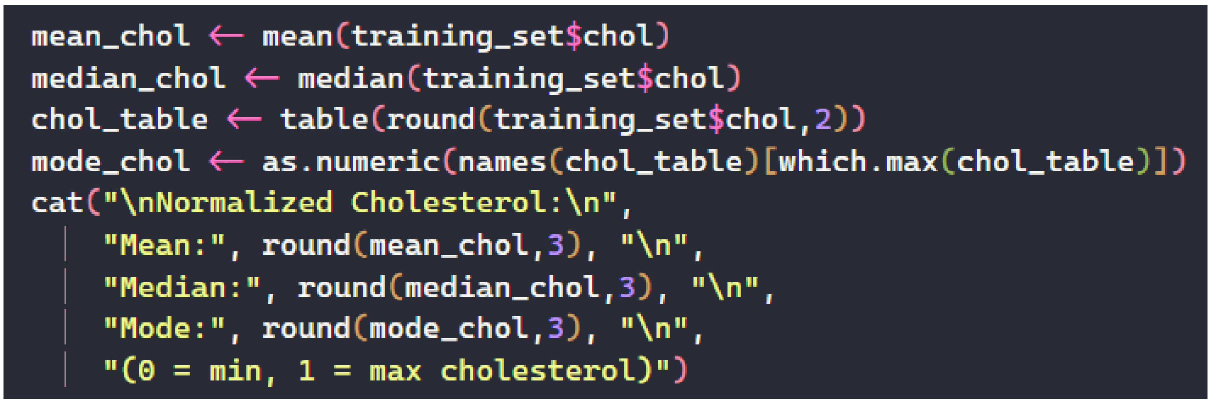
Output 33: Frequency Distribution of 'diagnosis' in the Training Set

Description:

Most patients are labeled as "Healthy" (114 patients), with 100 patients diagnosed with a disease. This suggests that the dataset is balanced, which is ideal for training models to predict heart disease or health status.

26.3. chol Column

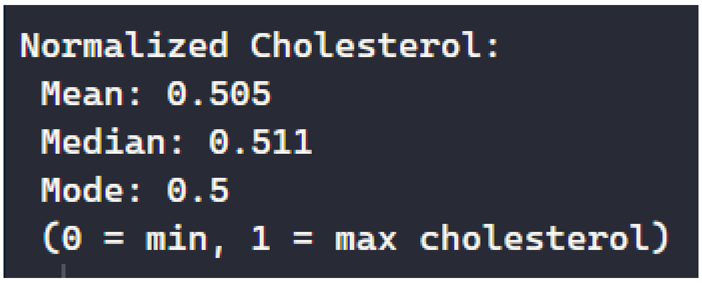
Code:



```r
mean_chol <- mean(training_set$chol)
median_chol <- median(training_set$chol)
chol_table <- table(round(training_set$chol,2))
mode_chol <- as.numeric(names(chol_table)[which.max(chol_table)])
cat("\nNormalized Cholesterol:\n",
    "Mean:", round(mean_chol,3), "\n",
    "Median:", round(median_chol,3), "\n",
    "Mode:", round(mode_chol,3), "\n",
    "(0 = min, 1 = max cholesterol)")
```

Code 34: R Script for Calculating Central Tendency of 'chol' in Training Set

Output:



```
Normalized Cholesterol:
 Mean: 0.505
 Median: 0.511
 Mode: 0.5
 (0 = min, 1 = max cholesterol)
```

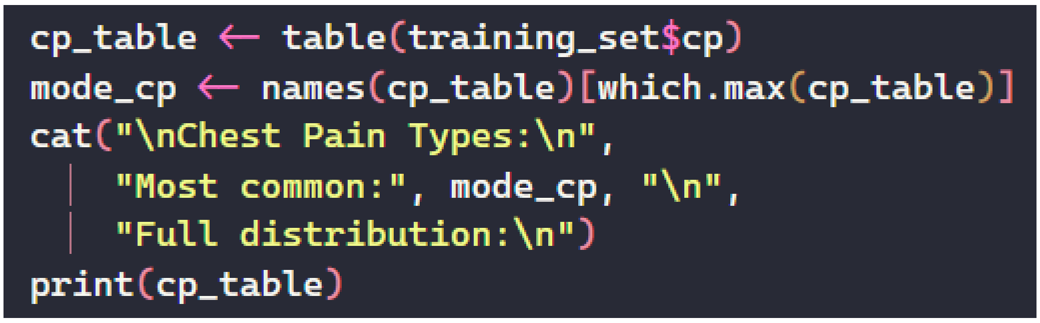
Output 34: Central Tendency Measures of 'chol' in Training Set

Description:

The average normalized cholesterol value is 0.505, with a median of 0.511 and a mode of 0.5. These values indicate a uniform distribution of cholesterol levels across the patients, and the data is normalized on a 0-1 scale, which is good for standardization in predictive modeling.

26.4. cp Column

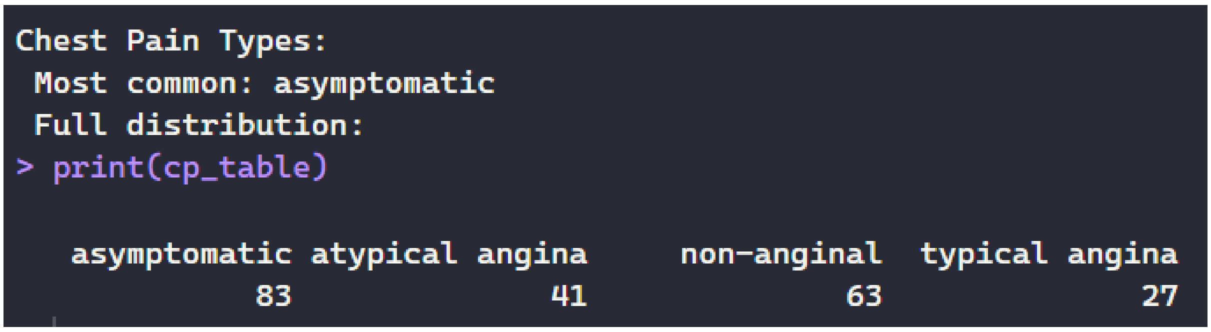
Code:



```r
cp_table <- table(training_set$cp)
mode_cp <- names(cp_table)[which.max(cp_table)]
cat("\nChest Pain Types:\n",
    "Most common:", mode_cp, "\n",
    "Full distribution:\n")
print(cp_table)
```

Code 35: Script for Analyzing 'cp' Distribution in the Training Set

Output:



```
Chest Pain Types:
 Most common: asymptomatic
 Full distribution:
> print(cp_table)

   asymptomatic atypical angina     non-anginal  typical angina 
             83              41              63              27 
```

Output 35: Frequency Distribution of 'cp' in the Training Set

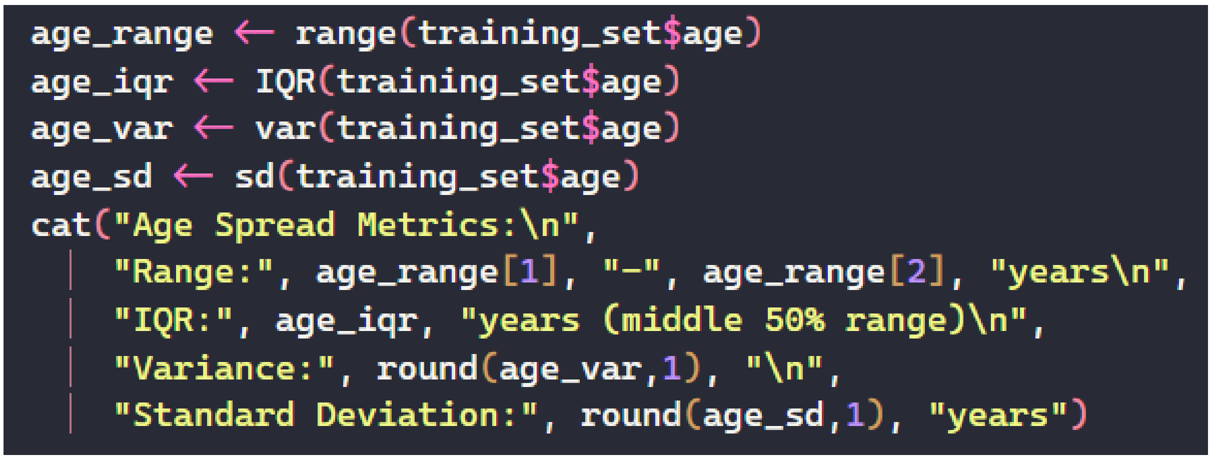
Description:

The most common type of chest pain reported is "asymptomatic" (83 cases), followed by "non-anginal" (63 cases), and "atypical angina" (41 cases). Only 27 patients reported "typical angina." This distribution suggests that many patients are asymptomatic or have non-specific symptoms, which might indicate hidden cardiovascular risk.

27. Spread Analysis for the Training Set

27.1.age Column

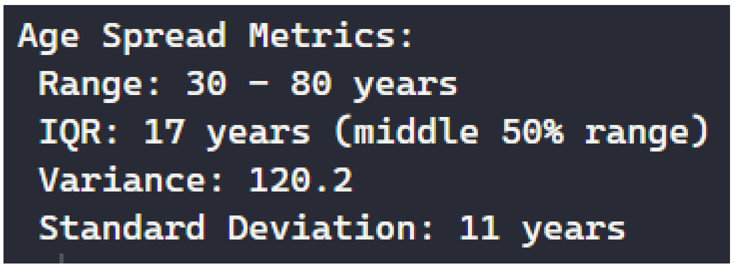
Code:



```r
age_range <- range(training_set$age)
age_iqr <- IQR(training_set$age)
age_var <- var(training_set$age)
age_sd <- sd(training_set$age)
cat("Age Spread Metrics:\n",
    "Range:", age_range[1], "-", age_range[2], "years\n",
    "IQR:", age_iqr, "years (middle 50% range)\n",
    "Variance:", round(age_var,1), "\n",
    "Standard Deviation:", round(age_sd,1), "years")
```

Code 36: R Script for Calculating Spread of 'age' in the Training Set

Output:



```
Age Spread Metrics:
 Range: 30 - 80 years
 IQR: 17 years (middle 50% range)
 Variance: 120.2
 Standard Deviation: 11 years
```

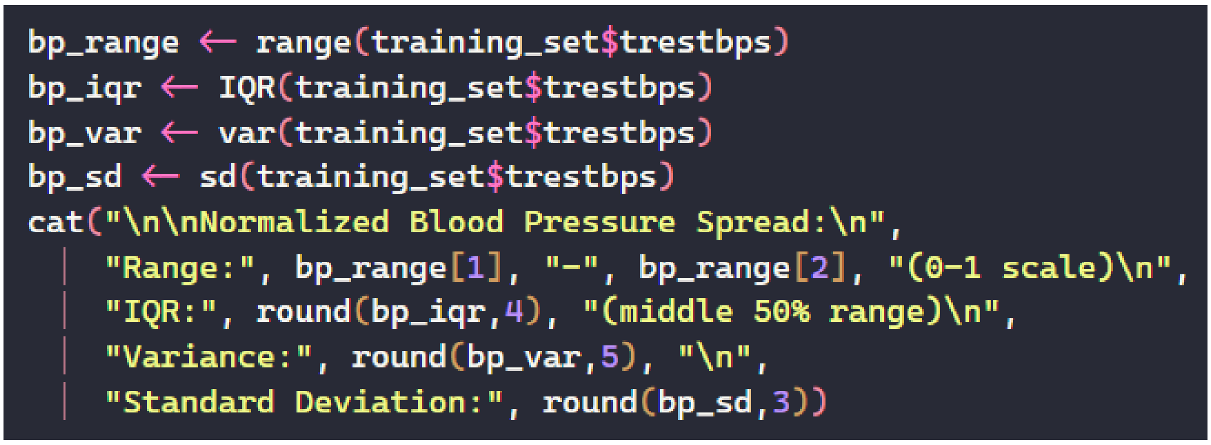
Output 36: Measures of Spread for 'age' in the Training Set

Description:

The age range is from 30 to 80 years, with an interquartile range (IQR) of 17 years. This suggests a broad age range, but with moderate clustering in the middle. The standard deviation of 11 years implies there is moderate variability in age among patients, which is typical for health datasets.

27.2. trestbps Column

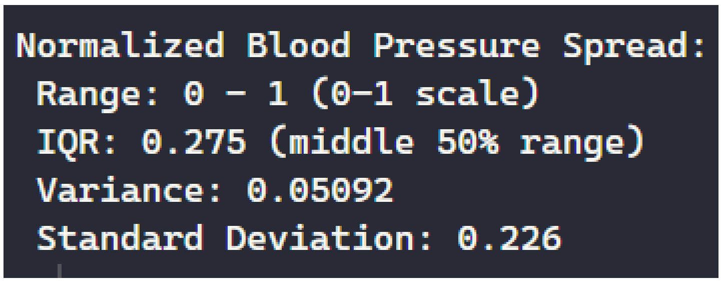
Code:



```r
bp_range ← range(training_set$trestbps)
bp_iqr ← IQR(training_set$trestbps)
bp_var ← var(training_set$trestbps)
bp_sd ← sd(training_set$trestbps)
cat("\n\nNormalized Blood Pressure Spread:\n",
    "Range:", bp_range[1], "-", bp_range[2], "(0-1 scale)\n",
    "IQR:", round(bp_iqr,4), "(middle 50% range)\n",
    "Variance:", round(bp_var,5), "\n",
    "Standard Deviation:", round(bp_sd,3))
```

Code 37: Script for Calculating Spread of Normalized 'trestbps' in Training Set

Output:



```
Normalized Blood Pressure Spread:
 Range: 0 - 1 (0-1 scale)
 IQR: 0.275 (middle 50% range)
 Variance: 0.05092
 Standard Deviation: 0.226
```

Output 37: Measures of Spread for Normalized 'trestbps' in Training Set

Description:

The normalized blood pressure ranges from 0 to 1, with an IQR of 0.275. The variance of 0.05092 and standard deviation of 0.226 indicate moderate variability in blood pressure among the patients. While the blood pressure values are consistent, some degree of variation exists, which is important for capturing different health conditions in the model.

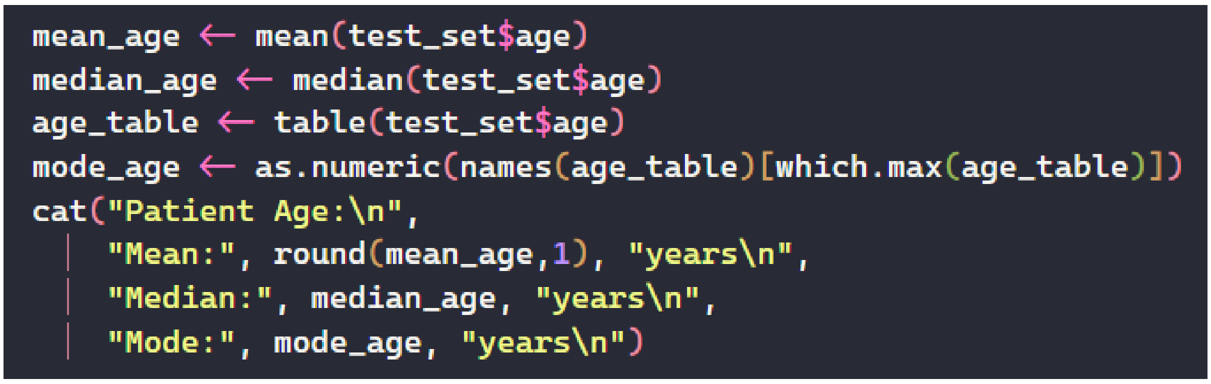
Overall Interpretation

The training dataset shows that most patients are middle-aged (54-55 years), with a balanced distribution between healthy and diseased cases, which is ideal for model training. Cholesterol levels are moderate and symmetrically distributed, while many patients experience asymptomatic chest pain, highlighting the importance of diagnostic testing. The age range is 30-80 years, with moderate variability, and blood pressure values are concentrated around the middle, indicating consistency. Overall, the dataset is well-suited for training models due to its balanced, moderate variability and representative features.

28. Central Tendency Analysis for the Testing Set

28.1. age Column

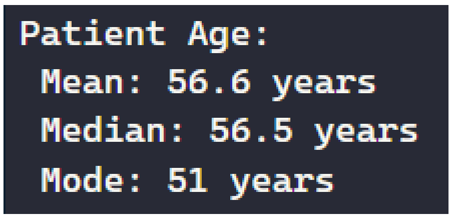
Code:



```r
mean_age <- mean(test_set$age)
median_age <- median(test_set$age)
age_table <- table(test_set$age)
mode_age <- as.numeric(names(age_table)[which.max(age_table)])
cat("Patient Age:\n",
    "Mean:", round(mean_age,1), "years\n",
    "Median:", median_age, "years\n",
    "Mode:", mode_age, "years\n")
```

Code 38: R Script for Calculating Central Tendency of 'age' in the Testing Set

Output:



```
Patient Age:
 Mean: 56.6 years
 Median: 56.5 years
 Mode: 51 years
```

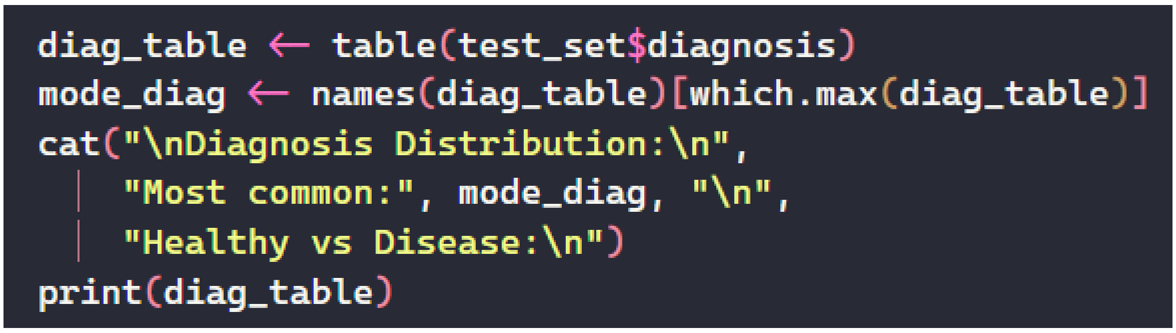
Output 38: Central Tendency Measures of 'age' in the Testing Set

Description:

The mean (56.6 years) and median (56.5 years) of the patient's age are identical, indicating a symmetric distribution. The mode is 51 years, suggesting that this age group is slightly more common. The age distribution shows that most patients in the dataset are middle-aged adults, which aligns with typical patterns seen in heart disease.

28.2. diagnosis Column

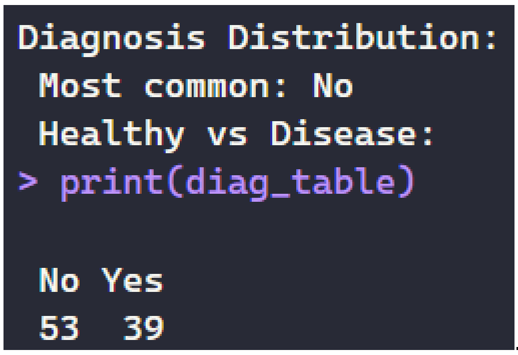
Code:



```r
diag_table <- table(test_set$diagnosis)
mode_diag <- names(diag_table)[which.max(diag_table)]
cat("\nDiagnosis Distribution:\n",
    "Most common:", mode_diag, "\n",
    "Healthy vs Disease:\n")
print(diag_table)
```

Code 39: Script for Analyzing 'diagnosis' Distribution in the Testing Set

Output:



```
Diagnosis Distribution:
 Most common: No
 Healthy vs Disease:
> print(diag_table)

 No Yes
 53  39
```

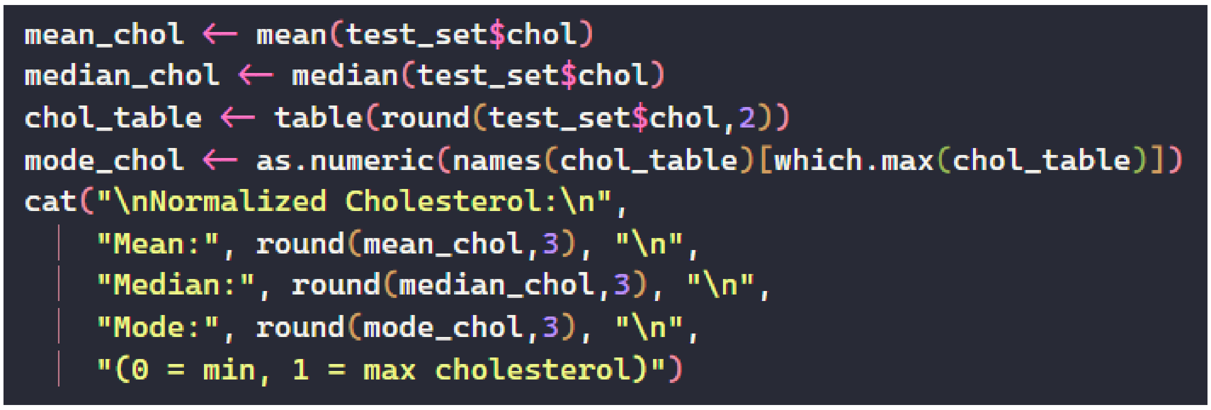
Output 39: Frequency Distribution of 'diagnosis' in the Testing Set

Description:

Most of the patients are healthy (53 healthy vs. 39 diseased), but there is still a considerable portion of diseased patients. This balance between healthy and diseased cases is beneficial for training a model, as it ensures that both classes are adequately represented, promoting fairness in predictive modeling.

28.3. chol Column

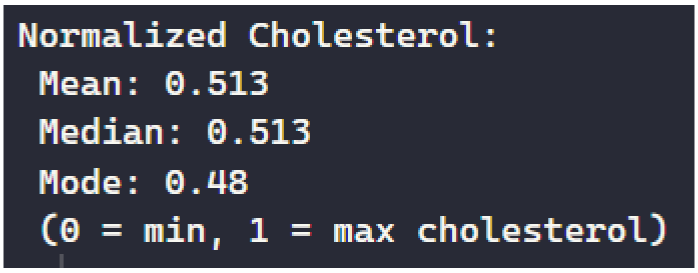
Code:



```r
mean_chol <- mean(test_set$chol)
median_chol <- median(test_set$chol)
chol_table <- table(round(test_set$chol,2))
mode_chol <- as.numeric(names(chol_table)[which.max(chol_table)])
cat("\nNormalized Cholesterol:\n",
    "Mean:", round(mean_chol,3), "\n",
    "Median:", round(median_chol,3), "\n",
    "Mode:", round(mode_chol,3), "\n",
    "(0 = min, 1 = max cholesterol)")
```

Code 40: R Script for Calculating Central Tendency of 'chol' in Testing Set

Output:



```
Normalized Cholesterol:
 Mean:  0.513
 Median:  0.513
 Mode:  0.48
 (0 = min, 1 = max cholesterol)
```

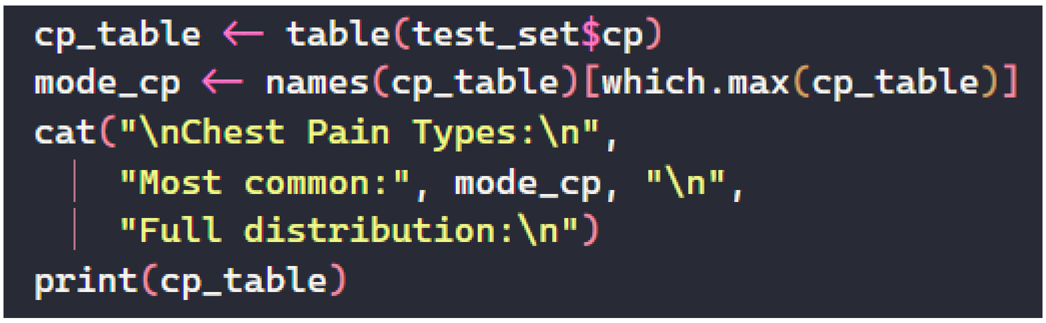
Output 40: Central Tendency Measures of 'chol' in Testing Set

Description:

The mean and median cholesterol levels are both 0.513, and the mode is 0.48, indicating that most patients have cholesterol levels near the center of the range. This suggests that the dataset contains mostly patients with moderate cholesterol levels, and the distribution is symmetrical, which is good for model training as it prevents skewing due to extreme values.

28.4. cp Column

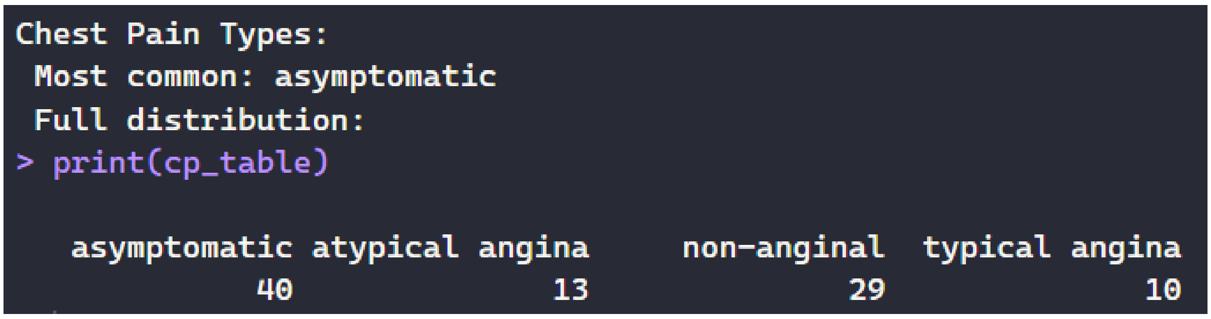
Code:



```r
cp_table <- table(test_set$cp)
mode_cp <- names(cp_table)[which.max(cp_table)]
cat("\nChest Pain Types:\n",
    "Most common:", mode_cp, "\n",
    "Full distribution:\n")
print(cp_table)
```

Code 41: R Script for Analyzing 'cp' Distribution in the Testing Set

Output:



```
Chest Pain Types:
 Most common: asymptomatic
 Full distribution:
> print(cp_table)

   asymptomatic atypical angina     non-anginal  typical angina 
             40              13              29              10 
```

Output 41: Frequency Distribution of 'cp' in the Testing Set

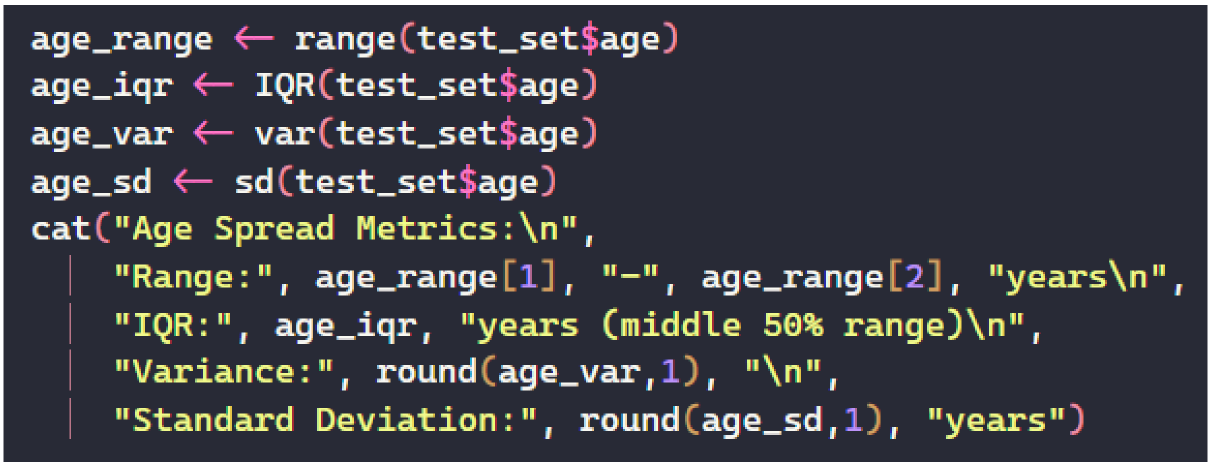
Description:

The most common chest pain type is "asymptomatic" (40 patients), followed by "non-anginal" (29), "atypical angina" (13), and "typical angina" (10). The predominance of asymptomatic patients is concerning because it means that many patients may have heart disease without showing obvious symptoms. This highlights the importance of using diagnostic tests beyond just symptom observation for accurate heart disease detection.

29. Spread Analysis for the Testing Set

29.1. age Column

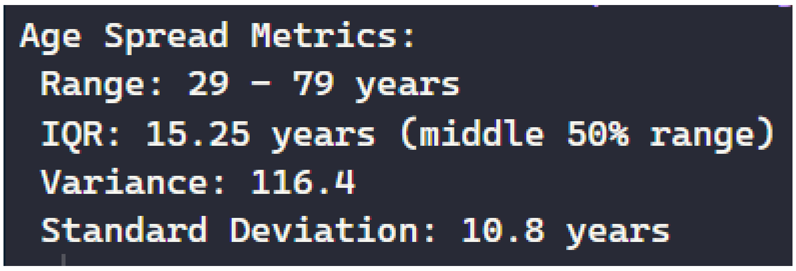
Code:



```r
age_range ← range(test_set$age)
age_iqr ← IQR(test_set$age)
age_var ← var(test_set$age)
age_sd ← sd(test_set$age)
cat("Age Spread Metrics:\n",
    "Range:", age_range[1], "-", age_range[2], "years\n",
    "IQR:", age_iqr, "years (middle 50% range)\n",
    "Variance:", round(age_var,1), "\n",
    "Standard Deviation:", round(age_sd,1), "years")
```

Code 42: R Script for Calculating Spread of 'age' in the Testing Set

Output:



```
Age Spread Metrics:
 Range: 29 - 79 years
 IQR: 15.25 years (middle 50% range)
 Variance: 116.4
 Standard Deviation: 10.8 years
```

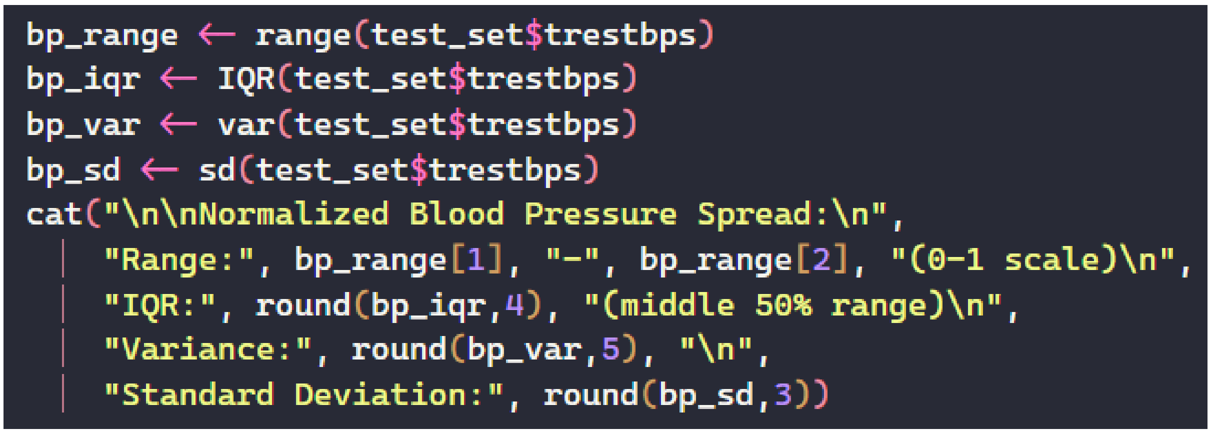
Output 42: Measures of Spread for 'age' in the Testing Set

Description:

The range of ages spans from 29 to 79 years, and the interquartile range (IQR) is 15.25 years, meaning that the middle 50% of patients are concentrated in a 15-year age span (between 47 and 63 years). The standard deviation of 10.8 years suggests moderate variation in age.

29.2. trestbps Column

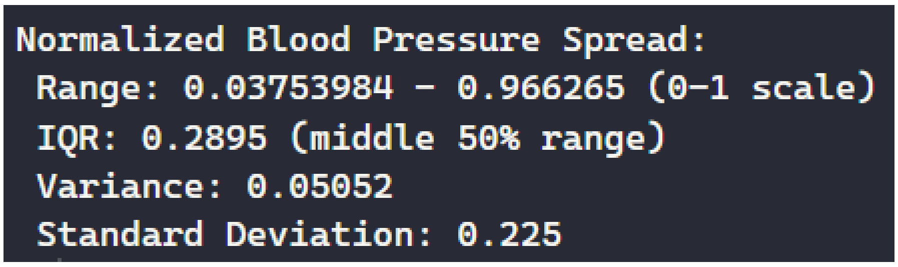
Code:



```r
bp_range ← range(test_set$trestbps)
bp_iqr ← IQR(test_set$trestbps)
bp_var ← var(test_set$trestbps)
bp_sd ← sd(test_set$trestbps)
cat("\n\nNormalized Blood Pressure Spread:\n",
    "Range:", bp_range[1], "-", bp_range[2], "(0-1 scale)\n",
    "IQR:", round(bp_iqr,4), "(middle 50% range)\n",
    "Variance:", round(bp_var,5), "\n",
    "Standard Deviation:", round(bp_sd,3))
```

Code 43: R Script for Calculating Spread of 'trestbps' in Testing Set

Output:



```
Normalized Blood Pressure Spread:
 Range: 0.03753984 - 0.966265 (0-1 scale)
 IQR: 0.2895 (middle 50% range)
 Variance: 0.05052
 Standard Deviation: 0.225
```

Output 43: Measures of Spread for Normalized 'trestbps' in Testing Set

Description:

The range of normalized blood pressure values is between 0.038 and 0.966, with an IQR of 0.2895. The standard deviation of 0.225 shows that blood pressure values are moderately spread around the mean, with few extreme outliers. This indicates that while blood pressure varies, it does not show extreme fluctuations that could skew results.

OVERALL:

The dataset is balanced, with moderate variability in both age and blood pressure. The most common age group is middle-aged, and cholesterol levels are moderate among most patients. There is a notable proportion of asymptomatic patients, suggesting that some individuals might be unaware of their heart disease risks. The data shows no extreme outliers.

30. Conclusion

At first, we familiarized ourselves with the dataset, analyzing its structure and main features, and making a basic assessment of whether there were any missing values. Later, the focus was shifted to the data cleaning stage. We attempted to resolve all missing values through imputation—applying the mean to numerical features such as ‘age,’ and the mode to categorical features like ‘ca’ and ‘thal.’

In addition, we dealt with outliers in continuous variables (‘age,’ ‘trestbps,’ ‘chol,’ ‘oldpeak’) based on the IQR method. Moreover, we aimed to fix all erroneous values like those in the ‘sex’ column and searched for and eliminated duplicate rows. After these operations, the data was subjected to transformation.

For analytical convenience, categorical features including ‘thal’ and ‘num’ were turned into numbers, and the ‘num’ column was modified into ‘diagnosis,’ which is more informative. To ensure that features with wider ranges did not dominate future analyses, continuous variables (‘trestbps,’ ‘chol,’ ‘oldpeak,’ ‘thalach’) were normalized to a common scale. Additional records of females were also added to eliminate unwanted biases.

Lastly, we carried out exploratory data analysis (EDA). This was done by looking at central tendencies (mean, median, mode) and their spread (range, IQR, standard deviation) for 'age', 'diagnosis', 'chol', 'cp', 'trestbps' on a full dataset as well as on the training and testing datasets after splitting them. This helped us grasp feature distribution as well as to confirm representativeness of the training and testing sets relative to the overall data.

To sum up, the algorithm stepped through various stages including cleaning, exploring, and transforming the heart disease dataset while providing a clear set of insights into patient data which could later be utilized with machine learning techniques.