CLUSTERING HEART DISEASE PATIENT DATA

&

REDUCING TRAFFIC MORTALITY IN THE USA

PROJECTS

Submitted in partial fulfilment of the requirement for the award of the degree of

MASTER'S IN DATA SCIENCE

In course

DATA MINING (CSCI 415)

by

MEHREET SINGH BAJAJ (1274698)
RISHABH BADJATYA (1270662)
MAITRY JARIWALA (1275288)
MOHAMMED SAFWAN ASLAM KAZI (1270795)

New York Institute of Technology

DECLARATION

We hereby declare that the projects entitled "**CLUSTERING HEART DISEASE PATIENT DATA & REDUCING TRAFFIC MORTALITY IN THE USA"** submitted as part of the partial course requirements for the course ***DATA MINING (project part – 3)*** , for the award of the degree of Masters in Data Science at New York Institute of Technology during the 2nd semester, has been carried out by us as a group . We declare that the project has not formed the basis for the award of any degree, associate ship, fellowship or any other similar titles elsewhere.

Further, we declare that we will not share, re-submit or publish the code, idea, framework and/or any publication that may arise out of this work for academic or profit purposes without obtaining the prior written consent of the Course Faculty Mentor and Course Instructor.

Signature of the Students:

MEHREET SINGH BAJAJ (1274698)

MAITRY JARIWALA (1275288)

RISHABH BADJATYA (1270662)

MOHAMMED SAFWAN ASLAM KAZI **(1270795)**

Place: New York Institute of Technology

Date: 12.07.2020

CERTIFICATE

This is to certify that the projects entitled " **CLUSTERING HEART DISEASE PATIENT DATA & REDUCING TRAFFIC MORTALITY IN THE USA** " is a record of bonafide work carried out as part of the course ***DATA MINING (project part 3)***, under my guidance by ***Mehreet Singh Bajaj (1274698)*** , **Maitry Jariwala(1275288)**, **Rishabh Badjatya(1270662), Mohammed Safwan Aslam Kazi 1270795** during the academic semester ***2nd***, in partial fulfilment of the requirements for the award of the Degree of Masters in Data Science, at New York Institute of Technology during academic year 2020.

HUANYING (HELEN) GU
Project guide
New York Institute of Technology

ACKNOWLEDGEMENT

We as a group wish to express our deep sense of gratitude to our guide HUANYING (HELEN) GU, Professor, Computer and Data Science department, New York Institute of Technology for her guidance and useful suggestions, for encouraging us to work on this project, using the techniques taught in the class and implementing them for real case scenarios, without which it would've been impossible to complete this project, that too in time. We are really thankful to her for believing in me and my project idea and helping us in shaping it up even better than we thought we could.

We feel blessed to have each other as a group of friends with same thinking and different ideologies which helped us in shaping the project perfectly in a fun and learning way. It was due to the intellectual discussion that we had with each other all the time that always motivated us and inspired us to do something, that we collaborated together and come up with the willingness to do these projects.

MEHREET SINGH BAJAJ (1274698)

MAITRY JARIWALA (1275288)

RISHABH BADJATYA (1270662)

MOHAMMED SAFWAN ASLAM KAZI (1270795)

Master's in data science

DATA MINING (2nd semester)

ABSTRACT

For part 3 of project assignment, we have done 2 projects as follows:

PROJECT 1: CLUSTERING HEART DISEASE PATIENT DATA

There are many industries where understanding how things group together is beneficial. For example, retailers want to understand the similarities among their customers to direct advertisement campaigns, and botanists classify plants based on their shared similar characteristics. One way to group objects is to use clustering algorithms. We are going to explore the usefulness of unsupervised clustering algorithms to help doctors understand which treatments might work with their patients.

We are going to cluster anonymized data of patients who have been diagnosed with heart disease. Patients with similar characteristics might respond to the same treatments, and doctors could benefit from learning about the treatment outcomes of patients like those they are treating. The data we are analyzing comes from the V.A. Medical Center in Long Beach, CA. Before running any analysis, it is essential to get an idea of what the data look like. The clustering algorithms we will use require numeric data—we'll check that all the data are numeric. In this project, we will be brushing up o brushing up on our base R skills.

PROJECT 2: **REDUCING TRAFFIC MORTALITY IN THE USA**

Road traffic injuries in the United States of America (USA) kills roughly around 154,089 people each year, representing 12% of the road traffic deaths worldwide. The data rate of fatal road accidents was steadily decreasing since 80s' but for the past ten years there was a stagnation in this reduction. As per the records there are almost 500-800 drivers per billion miles. Due to the increased number of vehicles used in the states there has been an increase in traffic related fatalities as well in the past ten years and it is increasing rapidly day by day. If we consider number of million miles driven with respect to number of driver fatal rate annually this roughly sums up to 2.95 million miles driven with respect to 100K number of driver fatal rate annually. These statistics demonstrates that road traffic death rate for the region as a whole might be 29 drivers per 100K population. Usually road traffic death rate also includes death rates of pedestrians, cyclists and motorcyclists as well and not just the death rates of drivers. As per the Pan American Health Organization (PAHO) there is a statistical pie distribution which shows that in these death rates car occupants, motorcyclists, and pedestrians accounts for 34%, 23% and 22% respectively, while cyclists represent 3% and remaining 18% of the deaths are from other categories which are not specified.

By looking at the above-mentioned demographics of traffic accident victims in the United States we found that there is a lot variation between states. Now we want to understand if there are patterns in this variation in order to derive suggestions for a policy action plan. In particular, instead of implementing a costly nation-wide plan we want to focus on groups of states with similar profiles. To accomplish these tasks, we will make use of Data Wrangling, Data Visualization, Machine Learning, Dimensionality Reduction and Unsupervised Learning. The data given to us was originally collected by the National Highway Traffic Safety Administration and the National Association of Insurance Commissioners.

TABLE OF CONTENTS

Page No.

PROJECT 2: REDUCING TRAFFIC MORTALITY IN USA

INTRODUCTION

1.1 Basic Introduction

In today's fast-growing world in which we are surrounded by problems which can be solved easily using the knowledge of computing science, we discovered one of the common problems which has not been paid good attention to from a long time and it really needs an efficient solution. Heart disease is one of the common health conditions which an individual faces at some stage of their life. In this modern automated self-driven world, when the highly qualified doctors sometimes cannot predict the right treatment at right time with patients having complex and particular heart disease, the algorithms and machine learning techniques comes handy. Data mining is the solution to this problem as it extracts the hidden information from the very complex medical data set. By predicting the disease and treating it right on time can save so any patient's life. Good data-driven systems for predicting heart disease can improve the entire research and prevention process, making sure that more people can live healthy lives.

1.2 Scope of Work

In the United States, the CDC (Centers for Disease Control and Prevention) is an efficient resource for knowledge about heart disease. As per them:

- In the United States, about 610,000 people die every year at the cause of heart disease– that's 1 in every 4 deaths.
- Heart disease is known as the main factor leading to death of people. In 2009 more than half of the men suffered with death all caused some heart disease.
- 370,000 people die annually because of most common type of heart disease known as Coronary heart disease (CHD)
- 735,000 Americans have a heart attack every year. Of these, first heart attack is almost in 525,000 and people who have already had a heart attack are around 210,000.
- Heart disease is the leading cause of death for people of most ethnicities in the United States, including African Americans, Hispanics, and whites.

So, taking all these into consideration, we will design a clustering-based model which will group the patients with same kind of heart disease in a cluster in an accurate manner.

Anonymized data of patients who have been diagnosed with cardiovascular disease will be clustered. Patients with matching characteristics might react to the similar treatments, and doctors could have a plus point from learning about outcomes of treatment of patients like those they are treating. The data which is analyzed is extracted from the V.A. Medical Center in Long Beach, CA. Before this we will also perform Exploratory data analysis [EDA] which will help us to understand the characteristics of the patients in the data. Doing this life of many patients can be saved on time as doctor will make a decision based on our model which will check the patient's symptoms and will direct to the possibility of having a heart attack or what particular group the patient belong to so that giving proper treatment and care on time can prove to be a lifesaving technique.

REQUIREMENT ANALYSIS

2.1 Functional Requirements

Providing the details of the project, all requirements and specifications will be made. Some of the functionalities we are going to use in our project are the basic principles of Data science along with R. Clustering data grouping techniques from Machine Learning and Visualizations using ggplot2. Most of the important libraries of R have been implemented in our code in the best possible way to enhance the quality of our project.

All the work has been done in the r environment of R-studio using base language as R. This provides a better work environments and faster run time than other IDE available out there .

2.2Dataset

This database has 76 attributes, and 14 attributes are considered to be most relevant for our target in the dataset.
The "target" variable means the presence of heart disease in the patient. It is integer valued 0 and 1. 0 meaning no disease and 1 means presence of a disease
The names and social security numbers of the patients were recently removed from the database, replaced with dummy values.
The data we are analyzing comes from the V.A. Medical Center in Long Beach
To download the data, visit here.

The patient_id column is a unique and random identifier. The remaining 14 features are described in the section below.

Only 14 attributes used:

1. **Age**: Age of subject
2. **Sex**: Gender of subject:
   0 = female 1 = male
3. **Chest-pain type**: Type of chest-pain experienced by the individual:
   1 = typical angina
   2 = atypical angina
   3 = non-angina pain
   4 = asymptomatic angina
4. **Resting Blood Pressure**: Resting blood pressure in mm Hg
5. **Serum Cholesterol**: Serum cholesterol in mg/dl
6. **Fasting Blood Sugar**: Fasting blood sugar level relative to 120 mg/dl: 0 = fasting blood sugar <= 120 mg/dl
   1 = fasting blood sugar > 120 mg/dl
7. **Resting ECG**: Resting electrocardiographic results
   0 = normal

1 = ST-T wave abnormality
2 = left ventricle hypertrophy

8. **Max Heart Rate Achieved**: Max heart rate of subject
9. **Exercise Induced Angina**:
   0 = no 1 = yes
10. **ST Depression Induced by Exercise Relative to Rest**: ST Depression of subject
11. **Peak Exercise ST Segment**:
    1 = Up-sloaping
    2 = Flat
    3 = Down-sloaping
12. **Number of Major Vessels (0-3) Visible on Flouroscopy**: Number of visible vessels under flouro
13. **Thal**: Form of thalassemia:
    3 = normal
    6 = fixed defect
    7 = reversible defect
14. **Diagnosis of Heart Disease(target)**: Indicates whether subject is suffering from heart disease or not:
    0 = absence
    1, 2, 3, 4 = heart disease present

BACKGROUND ANALYSIS

3.1 A Look at Data

heart.csv

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
2	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
3	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
4	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
5	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
6	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
7	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
8	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
9	44	1	1	120	263	0	1	173	0	0	2	0	3	1
10	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
11	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
12	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
13	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
14	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
15	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
16	58	0	3	150	283	1	0	162	0	1	2	0	2	1
17	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
18	58	0	2	120	340	0	1	172	0	0	2	0	2	1
19	66	0	3	150	226	0	1	114	0	2.6	0	0	2	1
20	43	1	0	150	247	0	1	171	0	1.5	2	0	2	1
21	69	0	3	140	239	0	1	151	0	1.8	2	2	2	1
22	59	1	0	135	234	0	1	161	0	0.5	1	0	3	1
23	44	1	2	130	233	0	1	179	1	0.4	2	0	2	1
24	42	1	0	140	226	0	1	178	0	0	2	0	2	1
25	61	1	2	150	243	1	1	137	1	1	1	0	2	1
26	40	1	3	140	199	0	1	178	1	1.4	2	0	3	1
27	71	0	1	160	302	0	1	162	0	0.4	2	2	2	1
28	59	1	2	150	212	1	1	157	0	1.6	2	0	2	1
29	51	1	2	110	175	0	1	123	0	0.6	2	0	2	1
30	65	0	2	140	417	1	0	157	0	0.8	2	1	2	1
31	53	1	2	130	197	1	0	152	0	1.2	0	0	2	1
32	41	0	1	105	198	0	1	168	0	0	2	1	2	1
33	65	1	0	120	177	0	1	140	0	0.4	2	0	3	1
34	44	1	1	130	219	0	0	188	0	0	2	0	2	1
35	54	1	2	125	273	0	0	152	0	0.5	0	1	2	1
36	51	1	3	125	213	0	0	125	1	1.4	2	1	2	1
37	46	0	2	142	177	0	0	160	1	1.4	0	0	2	1
38	54	0	2	135	304	1	1	170	0	0	2	0	2	1

3.2 Methodology

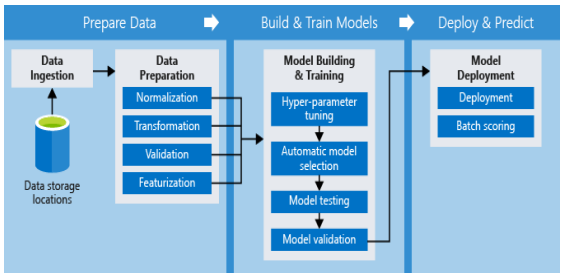


Fig 12 : The methodology diagram

This is the methodology which we will follow:

- **Data storage locations**: The recorded data or the raw data that is stored and will be used as the base for our whole project. Heavier the data is, the more trained our machine will be and more accurate the results will be.

- **Data ingestion**: Importing our set of data from the stored location to our main code which will further be processed. This includes steps like understanding our data, collecting it, describing the data, exploring and verifying the quality of it.

- **Data Preparation**: Consists of steps like selecting the data, cleaning it, constructing, integrating and formatting the data using technique like Normalization, Transformation, Validation and Featurization

- **Model Building & Training:** In this stage we will apply hyper parameter tuning (choose a set of optimal hyperparameters for a learning algorithm. Its value controls the learning process). Model selection (selecting various supervised algorithmic models which gives a better result to the given model), Model Testing, scoring (applying those algorithmic models from historical data set to new dataset)
- **Model Deployment:** scoring (applying those algorithmic models from historical data set to new dataset in order to uncover the practical insights that help solving a problem) and finally deploying the model.

3.3 How to overcome the problem?

Doctors frequently study former cases to learn how to best treat their patients. A patient who has a similar health history or symptoms to a previous patient could benefit from undergoing the same treatment. This project investigates whether doctors might be able to group together patients to target treatments using common unsupervised learning techniques. In this project you will use k-means and hierarchical clustering algorithms.

The dataset for this project contains characteristics of patients diagnosed with heart disease. It can be found here. It is a Classification_Problem, our goal is to predict the binary class target(heart_disease), which represents whether or not a patient has heart disease: 0 represents no heart disease present, 1 represents heart disease present.

APPROACH

This is just a short overview of what strategies have been used to solve the problem. The detailed description will be in the code and explanation section.

4.1 Importing libraries

Firstly, we imported some of the basic important libraries used in R

- tidyverse - data cleaning
- dplyr - data manipulation
- skimr - descriptive statistics
- ggvis - scatterplots
- caret - machine learning models
- ggplot - help in visualization
- corrplot - used for correlation analysis

4.2 Loading into R

Some pre-processing steps included downloading the data and loading it., pre-processed to make it smaller and faster to load. All the CSV (comma separated) files were loaded into R using the read.csv function data frame.

4.3 Tidying the Data

Now, to check the basic statistics of the data and separate the unwanted data from the important data summary() method was used and some variables showing no variance were observed and dumped and all N/A if were present were also thrown out. The data was scaled/Normalized.

4.4 Transforming the Data

To make the data look in a clean and clear way we have used a function mutate() to transform some attribute values presenting the data in a binary or numerical form to more clear character values . This will make our data to be in a more readable human format.

4.5 Data Visualization

Here we are doing the general visualization of our data, since the data is categorical Bar graph will show us some good results. We are using the library ggplot2 here for our visualization purpose

4.6 Correlation

We can check the correlation analysis from different variables, for this we have to use the dataset with numerical values. We will plot correlation graph which will represent how each variable are related to each other.
If to analyze in values, we will also be present the value chart for the same.

4.7 Using various clustering techniques

The techniques which we will be using in our project will be K- means algorithm and the hierarchal clustering.
Now that we have cleaned, transformed, scaled and visualized the data, we can start the clustering process.

k-means algorithm initially selects the cluster centers by randomly selecting points(), different iterations of the algorithm can result in different clusters**.**
If the algorithm is genuinely grouping similar observations (as opposed to clustering noise), then cluster assignments will be somewhat robust between various iterations of the algorithm. With regards to the heart disease data, this would mean that the same patients would be grouped even when the algorithm is initialized at different random points. If patients are not in similar clusters with various algorithm runs, then the clustering method is not picking up on meaningful relationships between patients.

We're going to explore how the patients are grouped with another iteration of the k-means algorithm. We will then be able to compare the resulting groups of patients.

Hierarchal clustering is another alternative to K- Means clustering. This method works well when data have a nested structure. Hierarchical clustering also does not require the number of clusters to be selected before running the algorithm. Clusters can be selected by using the dendrogram.

ANALYSIS OF CODE

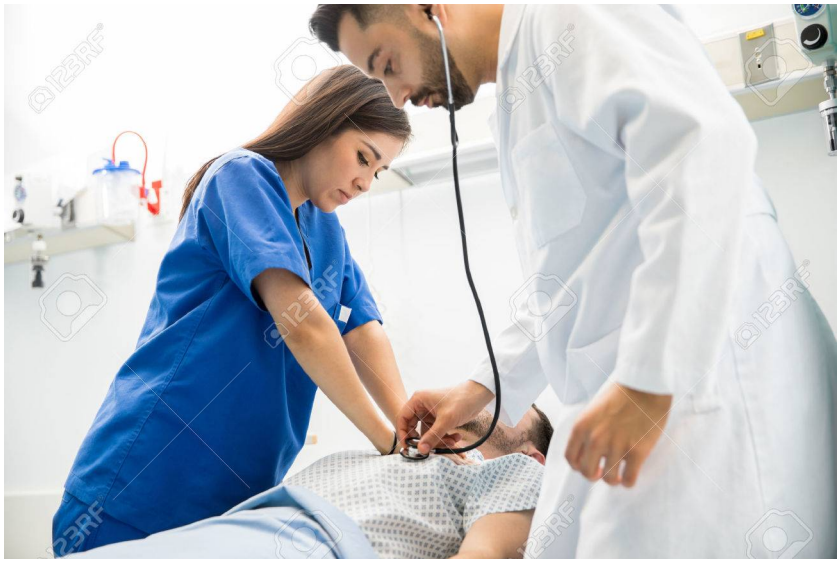
FINAL PROJECT ANALYSIS

Mehreet Singh Bajaj

12/7/2020

1. Targeting treatment for heart disease patients

There are many industries where understanding how things group together is beneficial. For example, retailers want to understand the similarities among their customers to direct advertisement campaigns, and botanists classify plants based on their shared similar characteristics. One way to group objects is to use clustering algorithms. We are going to explore the usefulness of unsupervised clustering algorithms to help doctors understand which treatments might work with their patients.



We are going to cluster anonymized data of patients who have been diagnosed with heart disease. Patients with similar characteristics might respond to the same treatments, and doctors could benefit from learning about the treatment outcomes of patients like those they are treating. The data we are analyzing comes from the V.A. Medical Center in Long Beach, CA. To download the data, visit here link

Before running any analysis, it is essential to get an idea of what the data look like. The clustering algorithms we will use require numeric data—we'll check that all the data are numeric. In this project, we will be brushing up on your base R skills.

```
# Loading the important libraries
library(tidyverse)
```

```
## ── Attaching packages ──────────────────────────────────────────────
──────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
## ✓ tibble  3.0.3     ✓ dplyr   1.0.0
## ✓ tidyr   1.1.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0

## ── Conflicts ─────────────────────────────────────────────────────────
──────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(dplyr)
library(ggplot2)
library(stats)
library(ggfortify)
library(corrplot)

## corrplot 0.84 loaded

#Load the data
heart_disease<- read.csv("/Users/mehreetsinghbajaj/Desktop/Fall 2020/DATA_
MINING/PROJECT/Part 3/Datasets/heart_disease_patient2.csv")

#print the first 10 lines
head(heart_disease,n=10)

##    age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca
thal
## 1   63   1  3      145  233   1       0     150     0     2.3     0  0
1
## 2   37   1  2      130  250   0       1     187     0     3.5     0  0
2
## 3   41   0  1      130  204   0       0     172     0     1.4     2  0
2
## 4   56   1  1      120  236   0       1     178     0     0.8     2  0
2
## 5   57   0  0      120  354   0       1     163     1     0.6     2  0
2
## 6   57   1  0      140  192   0       1     148     0     0.4     1  0
1
## 7   56   0  1      140  294   0       0     153     0     1.3     1  0
2
## 8   44   1  1      120  263   0       1     173     0     0.0     2  0
3
## 9   52   1  2      172  199   1       1     162     0     0.5     2  0
3
## 10  57   1  2      150  168   0       1     174     0     1.6     2  0
2
##    target
## 1       1
```

```
## 2       1
## 3       1
## 4       1
## 5       1
## 6       1
## 7       1
## 8       1
## 9       1
## 10      1
```

```
#getting a glimpse of the data
glimpse(heart_disease)
```

```
## Rows: 303
## Columns: 14
## $ age      <int> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64
, 58, …
## $ sex      <int> 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
1, 0, …
## $ cp       <int> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3,
0, 3, …
## $ trestbps <int> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140,
130, …
## $ chol     <int> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239,
275, …
## $ fbs      <int> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, …
## $ restecg  <int> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
1, 1, …
## $ thalach  <int> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160,
139, …
## $ exang    <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, …
## $ oldpeak  <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2,
0.2, …
## $ slope    <int> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0,
2, 2, …
## $ ca       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, …
## $ thal     <int> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, …
## $ target   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, …
```

```
# To particularly get the no. of rows and columns
ncol(heart_disease)
```

```
## [1] 14
```

```
nrow(heart_disease)
```

```
## [1] 303
```

```
# To check the col/attribute names
colnames(heart_disease)
```

```
##  [1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"
##  [7] "restecg"  "thalach"  "exang"    "oldpeak"  "slope"    "ca"
## [13] "thal"     "target"
```

2. Quantifying patient differences

It is important to conduct some exploratory data analysis (EDA) to familiarize ourselves with the data before clustering. EDA will help us learn more about the variables and make an informed decision about whether we should scale the data. Because k-means and hierarchical clustering measure similarity between points using a distance formula, it can place extra emphasis on certain variables that have a larger scale and thus larger differences between points. Exploratory data analysis helps us to understand the characteristics of the patients in the data. We need to get an idea of the value ranges of the variables and their distributions. This will also be helpful when we evaluate the clusters of patients from the algorithms. Are there more patients of one gender? What might an outlier look like?

```
#Evidence that data should be scaled
summary(heart_disease)
```

```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0
##  1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0
##  Median :55.00   Median :1.0000   Median :1.000   Median :130.0
##  Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0
##       chol            fbs             restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
##  Median :240.0   Median :0.0000   Median :1.0000   Median :153.0
##  Mean   :246.3   Mean   :0.1485   Mean   :0.5281   Mean   :149.6
##  3rd Qu.:274.5   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##      exang           oldpeak         slope             ca
##  Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.80   Median :1.000   Median :0.0000
##  Mean   :0.3267   Mean   :1.04   Mean   :1.399   Mean   :0.7294
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :6.20   Max.   :2.000   Max.   :4.0000
##       thal           target
##  Min.   :0.000   Min.   :0.0000
##  1st Qu.:2.000   1st Qu.:0.0000
##  Median :2.000   Median :1.0000
##  Mean   :2.314   Mean   :0.5446
##  3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :3.000   Max.   :1.0000
```

```r
# Remove id
heart_disease <- heart_disease[ , !(names(heart_disease) %in% c("id"))]

# Scaling data and saving as a data frame
scaled <- scale(heart_disease)

# What do the data look like now?
summary(scaled)
```

```
##       age                sex                cp              trestbps
##  Min.   :-2.79300   Min.   :-1.4660   Min.   :-0.93696   Min.   :-2.145
25
##  1st Qu.:-0.75603   1st Qu.:-1.4660   1st Qu.:-0.93696   1st Qu.:-0.662
77
##  Median : 0.06977   Median : 0.6799   Median : 0.03198   Median :-0.092
59
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.000
00
##  3rd Qu.: 0.73041   3rd Qu.: 0.6799   3rd Qu.: 1.00092   3rd Qu.: 0.477
60
##  Max.   : 2.49212   Max.   : 0.6799   Max.   : 1.96986   Max.   : 3.898
72
##       chol              fbs             restecg           thalach
##  Min.   :-2.3203   Min.   :-0.4169   Min.   :-1.0042   Min.   :-3.4336
##  1st Qu.:-0.6804   1st Qu.:-0.4169   1st Qu.:-1.0042   1st Qu.:-0.7049
##  Median :-0.1209   Median :-0.4169   Median : 0.8975   Median : 0.1464
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5448   3rd Qu.:-0.4169   3rd Qu.: 0.8975   3rd Qu.: 0.7139
##  Max.   : 6.1303   Max.   : 2.3905   Max.   : 2.7991   Max.   : 2.2856
##      exang            oldpeak            slope                ca
##  Min.   :-0.6955   Min.   :-0.8954   Min.   :-2.2708   Min.   :-0.7132
##  1st Qu.:-0.6955   1st Qu.:-0.8954   1st Qu.:-0.6480   1st Qu.:-0.7132
##  Median :-0.6955   Median :-0.2064   Median :-0.6480   Median :-0.7132
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.4331   3rd Qu.: 0.4827   3rd Qu.: 0.9747   3rd Qu.: 0.2646
##  Max.   : 1.4331   Max.   : 4.4445   Max.   : 0.9747   Max.   : 3.1983
##       thal              target
##  Min.   :-3.7786   Min.   :-1.092
##  1st Qu.:-0.5121   1st Qu.:-1.092
##  Median :-0.5121   Median : 0.913
##  Mean   : 0.0000   Mean   : 0.000
##  3rd Qu.: 1.1212   3rd Qu.: 0.913
##  Max.   : 1.1212   Max.   : 0.913
```

3.Data Tranformation

```r
heart_disease2<- heart_disease %>%
  mutate(sex=if_else(sex==1,"Male", "female"),
         fbs=if_else(fbs==1,">120", "<=120"),
         exang=if_else(exang==1,"Yes", "No"),
         cp=if_else(cp==1,"typical angina ",
                    if_else(cp==2," atypical angina ",
```

```
                    if_else(cp==3,"non-anginal pain ","asymptomatic ")),
         restecg=if_else(restecg==0,"Normal",
                         if_else(restecg==1,"Abnormal","Probable or Defini
te")),
          #Performiing coercion
         slope=as.factor(slope),
         ca=as.factor(ca),
         thal=as.factor(thal),
         target=if_else(target==1, "Yes", "No")
        ) %>%
  mutate_if(is.character,as.factor)%>%
  dplyr::select(target,sex,fbs,exang,cp,restecg,slope,ca,thal,everything()
)


#check the clarity of data by observing the first 10 rows of it
head(heart_disease2,n=10)
```

```
##    target    sex   fbs exang               cp  restecg slope ca thal a
ge
## 1     Yes   Male  >120    No non-anginal pain    Normal     0  0    1
63
## 2     Yes   Male <=120    No  atypical angina  Abnormal     0  0    2
37
## 3     Yes female <=120    No   typical angina    Normal     2  0    2
41
## 4     Yes   Male <=120    No   typical angina  Abnormal     2  0    2
56
## 5     Yes female <=120   Yes     asymptomatic  Abnormal     2  0    2
57
## 6     Yes   Male <=120    No     asymptomatic  Abnormal     1  0    1
57
## 7     Yes female <=120    No   typical angina    Normal     1  0    2
56
## 8     Yes   Male <=120    No   typical angina  Abnormal     2  0    3
44
## 9     Yes   Male  >120    No  atypical angina  Abnormal     2  0    3
52
## 10    Yes   Male <=120    No  atypical angina  Abnormal     2  0    2
57
##    trestbps chol thalach oldpeak
## 1       145  233     150     2.3
## 2       130  250     187     3.5
## 3       130  204     172     1.4
## 4       120  236     178     0.8
## 5       120  354     163     0.6
## 6       140  192     148     0.4
## 7       140  294     153     1.3
## 8       120  263     173     0.0
## 9       172  199     162     0.5
## 10      150  168     174     1.6
```

We can see that our data is in much clear and human readable form , our data is now transformed

4.Removing N/A (missing values) and replacing it with the mean

It is important to find the missing values because later on while clustering it will cause issues

```
is.na(heart_disease2)
```

```
##        target   sex   fbs exang    cp restecg slope    ca  thal   age t
restbps
##   [1,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [2,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [3,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [4,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [5,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [6,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [7,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [8,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##   [9,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [10,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [11,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [12,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [13,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [14,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [15,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [16,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [17,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [18,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [19,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [20,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
##  [21,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [22,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [23,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [24,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [25,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [26,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [27,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [28,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [29,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [30,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [31,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [32,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [33,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [34,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [35,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [36,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [37,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [38,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [39,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [40,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [41,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [42,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [43,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [44,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [45,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [46,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
```

```
FALSE
##  [47,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [48,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [49,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [50,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [51,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [52,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [53,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [54,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [55,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [56,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [57,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [58,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [59,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [60,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [61,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [62,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [63,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [64,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [65,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [66,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [67,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [68,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [69,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [70,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [71,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
##  [72,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [73,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [74,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [75,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [76,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [77,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [78,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [79,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [80,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [81,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [82,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [83,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [84,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [85,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [86,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [87,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [88,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [89,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [90,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [91,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [92,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [93,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [94,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [95,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [96,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [97,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
```

```
FALSE
##  [98,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##  [99,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [100,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [101,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [102,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [103,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [104,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [105,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [106,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [107,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [108,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [109,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [110,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [111,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [112,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [113,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [114,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [115,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [116,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [117,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [118,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [119,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [120,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [121,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [122,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## [123,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [124,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [125,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [126,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [127,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [128,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [129,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [130,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [131,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [132,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [133,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [134,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [135,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [136,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [137,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [138,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [139,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [140,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [141,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [142,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [143,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [144,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [145,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [146,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [147,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [148,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
```

```
FALSE
## [149,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [150,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [151,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [152,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [153,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [154,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [155,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [156,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [157,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [158,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [159,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [160,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [161,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [162,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [163,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [164,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [165,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [166,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [167,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [168,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [169,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [170,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [171,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [172,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [173,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## [174,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [175,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [176,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [177,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [178,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [179,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [180,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [181,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [182,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [183,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [184,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [185,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [186,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [187,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [188,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [189,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [190,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [191,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [192,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [193,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [194,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [195,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [196,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [197,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [198,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [199,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
```

```
FALSE
## [200,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [201,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [202,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [203,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [204,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [205,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [206,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [207,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [208,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [209,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [210,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [211,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [212,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [213,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [214,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [215,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [216,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [217,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [218,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [219,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [220,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [221,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [222,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [223,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [224,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## [225,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [226,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [227,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [228,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [229,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [230,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [231,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [232,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [233,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [234,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [235,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [236,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [237,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [238,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [239,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [240,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [241,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [242,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [243,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [244,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [245,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [246,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [247,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [248,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [249,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [250,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
```

```
FALSE
## [251,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [252,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [253,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [254,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [255,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [256,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [257,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [258,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [259,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [260,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [261,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [262,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [263,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [264,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [265,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [266,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [267,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [268,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [269,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [270,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [271,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [272,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [273,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [274,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [275,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## [276,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [277,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [278,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [279,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [280,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [281,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [282,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [283,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [284,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [285,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [286,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [287,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [288,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [289,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [290,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [291,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [292,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [293,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [294,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [295,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [296,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [297,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [298,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [299,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [300,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [301,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
```

```
FALSE
## [302,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
## [303,]  FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE
FALSE
##         chol thalach oldpeak
##   [1,] FALSE   FALSE   FALSE
##   [2,] FALSE   FALSE   FALSE
##   [3,] FALSE   FALSE   FALSE
##   [4,] FALSE   FALSE   FALSE
##   [5,] FALSE   FALSE   FALSE
##   [6,] FALSE   FALSE   FALSE
##   [7,] FALSE   FALSE   FALSE
##   [8,] FALSE   FALSE   FALSE
##   [9,] FALSE   FALSE   FALSE
##  [10,] FALSE   FALSE   FALSE
##  [11,] FALSE   FALSE   FALSE
##  [12,] FALSE   FALSE   FALSE
##  [13,] FALSE   FALSE   FALSE
##  [14,] FALSE   FALSE   FALSE
##  [15,] FALSE   FALSE   FALSE
##  [16,] FALSE   FALSE   FALSE
##  [17,] FALSE   FALSE   FALSE
##  [18,] FALSE   FALSE   FALSE
##  [19,] FALSE   FALSE   FALSE
##  [20,] FALSE   FALSE   FALSE
##  [21,] FALSE   FALSE   FALSE
##  [22,] FALSE   FALSE   FALSE
##  [23,] FALSE   FALSE   FALSE
##  [24,] FALSE   FALSE   FALSE
##  [25,] FALSE   FALSE   FALSE
##  [26,] FALSE   FALSE   FALSE
##  [27,] FALSE   FALSE   FALSE
##  [28,] FALSE   FALSE   FALSE
##  [29,] FALSE   FALSE   FALSE
##  [30,] FALSE   FALSE   FALSE
##  [31,] FALSE   FALSE   FALSE
##  [32,] FALSE   FALSE   FALSE
##  [33,] FALSE   FALSE   FALSE
##  [34,] FALSE   FALSE   FALSE
##  [35,] FALSE   FALSE   FALSE
##  [36,] FALSE   FALSE   FALSE
##  [37,] FALSE   FALSE   FALSE
##  [38,] FALSE   FALSE   FALSE
##  [39,] FALSE   FALSE   FALSE
##  [40,] FALSE   FALSE   FALSE
##  [41,] FALSE   FALSE   FALSE
##  [42,] FALSE   FALSE   FALSE
##  [43,] FALSE   FALSE   FALSE
##  [44,] FALSE   FALSE   FALSE
##  [45,] FALSE   FALSE   FALSE
```

```
##  [46,] FALSE   FALSE   FALSE
##  [47,] FALSE   FALSE   FALSE
##  [48,] FALSE   FALSE   FALSE
##  [49,] FALSE   FALSE   FALSE
##  [50,] FALSE   FALSE   FALSE
##  [51,] FALSE   FALSE   FALSE
##  [52,] FALSE   FALSE   FALSE
##  [53,] FALSE   FALSE   FALSE
##  [54,] FALSE   FALSE   FALSE
##  [55,] FALSE   FALSE   FALSE
##  [56,] FALSE   FALSE   FALSE
##  [57,] FALSE   FALSE   FALSE
##  [58,] FALSE   FALSE   FALSE
##  [59,] FALSE   FALSE   FALSE
##  [60,] FALSE   FALSE   FALSE
##  [61,] FALSE   FALSE   FALSE
##  [62,] FALSE   FALSE   FALSE
##  [63,] FALSE   FALSE   FALSE
##  [64,] FALSE   FALSE   FALSE
##  [65,] FALSE   FALSE   FALSE
##  [66,] FALSE   FALSE   FALSE
##  [67,] FALSE   FALSE   FALSE
##  [68,] FALSE   FALSE   FALSE
##  [69,] FALSE   FALSE   FALSE
##  [70,] FALSE   FALSE   FALSE
##  [71,] FALSE   FALSE   FALSE
##  [72,] FALSE   FALSE   FALSE
##  [73,] FALSE   FALSE   FALSE
##  [74,] FALSE   FALSE   FALSE
##  [75,] FALSE   FALSE   FALSE
##  [76,] FALSE   FALSE   FALSE
##  [77,] FALSE   FALSE   FALSE
##  [78,] FALSE   FALSE   FALSE
##  [79,] FALSE   FALSE   FALSE
##  [80,] FALSE   FALSE   FALSE
##  [81,] FALSE   FALSE   FALSE
##  [82,] FALSE   FALSE   FALSE
##  [83,] FALSE   FALSE   FALSE
##  [84,] FALSE   FALSE   FALSE
##  [85,] FALSE   FALSE   FALSE
##  [86,] FALSE   FALSE   FALSE
##  [87,] FALSE   FALSE   FALSE
##  [88,] FALSE   FALSE   FALSE
##  [89,] FALSE   FALSE   FALSE
##  [90,] FALSE   FALSE   FALSE
##  [91,] FALSE   FALSE   FALSE
##  [92,] FALSE   FALSE   FALSE
##  [93,] FALSE   FALSE   FALSE
##  [94,] FALSE   FALSE   FALSE
##  [95,] FALSE   FALSE   FALSE
##  [96,] FALSE   FALSE   FALSE
```

```
##  [97,] FALSE   FALSE   FALSE
##  [98,] FALSE   FALSE   FALSE
##  [99,] FALSE   FALSE   FALSE
## [100,] FALSE   FALSE   FALSE
## [101,] FALSE   FALSE   FALSE
## [102,] FALSE   FALSE   FALSE
## [103,] FALSE   FALSE   FALSE
## [104,] FALSE   FALSE   FALSE
## [105,] FALSE   FALSE   FALSE
## [106,] FALSE   FALSE   FALSE
## [107,] FALSE   FALSE   FALSE
## [108,] FALSE   FALSE   FALSE
## [109,] FALSE   FALSE   FALSE
## [110,] FALSE   FALSE   FALSE
## [111,] FALSE   FALSE   FALSE
## [112,] FALSE   FALSE   FALSE
## [113,] FALSE   FALSE   FALSE
## [114,] FALSE   FALSE   FALSE
## [115,] FALSE   FALSE   FALSE
## [116,] FALSE   FALSE   FALSE
## [117,] FALSE   FALSE   FALSE
## [118,] FALSE   FALSE   FALSE
## [119,] FALSE   FALSE   FALSE
## [120,] FALSE   FALSE   FALSE
## [121,] FALSE   FALSE   FALSE
## [122,] FALSE   FALSE   FALSE
## [123,] FALSE   FALSE   FALSE
## [124,] FALSE   FALSE   FALSE
## [125,] FALSE   FALSE   FALSE
## [126,] FALSE   FALSE   FALSE
## [127,] FALSE   FALSE   FALSE
## [128,] FALSE   FALSE   FALSE
## [129,] FALSE   FALSE   FALSE
## [130,] FALSE   FALSE   FALSE
## [131,] FALSE   FALSE   FALSE
## [132,] FALSE   FALSE   FALSE
## [133,] FALSE   FALSE   FALSE
## [134,] FALSE   FALSE   FALSE
## [135,] FALSE   FALSE   FALSE
## [136,] FALSE   FALSE   FALSE
## [137,] FALSE   FALSE   FALSE
## [138,] FALSE   FALSE   FALSE
## [139,] FALSE   FALSE   FALSE
## [140,] FALSE   FALSE   FALSE
## [141,] FALSE   FALSE   FALSE
## [142,] FALSE   FALSE   FALSE
## [143,] FALSE   FALSE   FALSE
## [144,] FALSE   FALSE   FALSE
## [145,] FALSE   FALSE   FALSE
## [146,] FALSE   FALSE   FALSE
## [147,] FALSE   FALSE   FALSE
```

```
## [148,] FALSE   FALSE   FALSE
## [149,] FALSE   FALSE   FALSE
## [150,] FALSE   FALSE   FALSE
## [151,] FALSE   FALSE   FALSE
## [152,] FALSE   FALSE   FALSE
## [153,] FALSE   FALSE   FALSE
## [154,] FALSE   FALSE   FALSE
## [155,] FALSE   FALSE   FALSE
## [156,] FALSE   FALSE   FALSE
## [157,] FALSE   FALSE   FALSE
## [158,] FALSE   FALSE   FALSE
## [159,] FALSE   FALSE   FALSE
## [160,] FALSE   FALSE   FALSE
## [161,] FALSE   FALSE   FALSE
## [162,] FALSE   FALSE   FALSE
## [163,] FALSE   FALSE   FALSE
## [164,] FALSE   FALSE   FALSE
## [165,] FALSE   FALSE   FALSE
## [166,] FALSE   FALSE   FALSE
## [167,] FALSE   FALSE   FALSE
## [168,] FALSE   FALSE   FALSE
## [169,] FALSE   FALSE   FALSE
## [170,] FALSE   FALSE   FALSE
## [171,] FALSE   FALSE   FALSE
## [172,] FALSE   FALSE   FALSE
## [173,] FALSE   FALSE   FALSE
## [174,] FALSE   FALSE   FALSE
## [175,] FALSE   FALSE   FALSE
## [176,] FALSE   FALSE   FALSE
## [177,] FALSE   FALSE   FALSE
## [178,] FALSE   FALSE   FALSE
## [179,] FALSE   FALSE   FALSE
## [180,] FALSE   FALSE   FALSE
## [181,] FALSE   FALSE   FALSE
## [182,] FALSE   FALSE   FALSE
## [183,] FALSE   FALSE   FALSE
## [184,] FALSE   FALSE   FALSE
## [185,] FALSE   FALSE   FALSE
## [186,] FALSE   FALSE   FALSE
## [187,] FALSE   FALSE   FALSE
## [188,] FALSE   FALSE   FALSE
## [189,] FALSE   FALSE   FALSE
## [190,] FALSE   FALSE   FALSE
## [191,] FALSE   FALSE   FALSE
## [192,] FALSE   FALSE   FALSE
## [193,] FALSE   FALSE   FALSE
## [194,] FALSE   FALSE   FALSE
## [195,] FALSE   FALSE   FALSE
## [196,] FALSE   FALSE   FALSE
## [197,] FALSE   FALSE   FALSE
## [198,] FALSE   FALSE   FALSE
```

```
## [199,] FALSE   FALSE   FALSE
## [200,] FALSE   FALSE   FALSE
## [201,] FALSE   FALSE   FALSE
## [202,] FALSE   FALSE   FALSE
## [203,] FALSE   FALSE   FALSE
## [204,] FALSE   FALSE   FALSE
## [205,] FALSE   FALSE   FALSE
## [206,] FALSE   FALSE   FALSE
## [207,] FALSE   FALSE   FALSE
## [208,] FALSE   FALSE   FALSE
## [209,] FALSE   FALSE   FALSE
## [210,] FALSE   FALSE   FALSE
## [211,] FALSE   FALSE   FALSE
## [212,] FALSE   FALSE   FALSE
## [213,] FALSE   FALSE   FALSE
## [214,] FALSE   FALSE   FALSE
## [215,] FALSE   FALSE   FALSE
## [216,] FALSE   FALSE   FALSE
## [217,] FALSE   FALSE   FALSE
## [218,] FALSE   FALSE   FALSE
## [219,] FALSE   FALSE   FALSE
## [220,] FALSE   FALSE   FALSE
## [221,] FALSE   FALSE   FALSE
## [222,] FALSE   FALSE   FALSE
## [223,] FALSE   FALSE   FALSE
## [224,] FALSE   FALSE   FALSE
## [225,] FALSE   FALSE   FALSE
## [226,] FALSE   FALSE   FALSE
## [227,] FALSE   FALSE   FALSE
## [228,] FALSE   FALSE   FALSE
## [229,] FALSE   FALSE   FALSE
## [230,] FALSE   FALSE   FALSE
## [231,] FALSE   FALSE   FALSE
## [232,] FALSE   FALSE   FALSE
## [233,] FALSE   FALSE   FALSE
## [234,] FALSE   FALSE   FALSE
## [235,] FALSE   FALSE   FALSE
## [236,] FALSE   FALSE   FALSE
## [237,] FALSE   FALSE   FALSE
## [238,] FALSE   FALSE   FALSE
## [239,] FALSE   FALSE   FALSE
## [240,] FALSE   FALSE   FALSE
## [241,] FALSE   FALSE   FALSE
## [242,] FALSE   FALSE   FALSE
## [243,] FALSE   FALSE   FALSE
## [244,] FALSE   FALSE   FALSE
## [245,] FALSE   FALSE   FALSE
## [246,] FALSE   FALSE   FALSE
## [247,] FALSE   FALSE   FALSE
## [248,] FALSE   FALSE   FALSE
## [249,] FALSE   FALSE   FALSE
```

```
## [250,] FALSE   FALSE   FALSE
## [251,] FALSE   FALSE   FALSE
## [252,] FALSE   FALSE   FALSE
## [253,] FALSE   FALSE   FALSE
## [254,] FALSE   FALSE   FALSE
## [255,] FALSE   FALSE   FALSE
## [256,] FALSE   FALSE   FALSE
## [257,] FALSE   FALSE   FALSE
## [258,] FALSE   FALSE   FALSE
## [259,] FALSE   FALSE   FALSE
## [260,] FALSE   FALSE   FALSE
## [261,] FALSE   FALSE   FALSE
## [262,] FALSE   FALSE   FALSE
## [263,] FALSE   FALSE   FALSE
## [264,] FALSE   FALSE   FALSE
## [265,] FALSE   FALSE   FALSE
## [266,] FALSE   FALSE   FALSE
## [267,] FALSE   FALSE   FALSE
## [268,] FALSE   FALSE   FALSE
## [269,] FALSE   FALSE   FALSE
## [270,] FALSE   FALSE   FALSE
## [271,] FALSE   FALSE   FALSE
## [272,] FALSE   FALSE   FALSE
## [273,] FALSE   FALSE   FALSE
## [274,] FALSE   FALSE   FALSE
## [275,] FALSE   FALSE   FALSE
## [276,] FALSE   FALSE   FALSE
## [277,] FALSE   FALSE   FALSE
## [278,] FALSE   FALSE   FALSE
## [279,] FALSE   FALSE   FALSE
## [280,] FALSE   FALSE   FALSE
## [281,] FALSE   FALSE   FALSE
## [282,] FALSE   FALSE   FALSE
## [283,] FALSE   FALSE   FALSE
## [284,] FALSE   FALSE   FALSE
## [285,] FALSE   FALSE   FALSE
## [286,] FALSE   FALSE   FALSE
## [287,] FALSE   FALSE   FALSE
## [288,] FALSE   FALSE   FALSE
## [289,] FALSE   FALSE   FALSE
## [290,] FALSE   FALSE   FALSE
## [291,] FALSE   FALSE   FALSE
## [292,] FALSE   FALSE   FALSE
## [293,] FALSE   FALSE   FALSE
## [294,] FALSE   FALSE   FALSE
## [295,] FALSE   FALSE   FALSE
## [296,] FALSE   FALSE   FALSE
## [297,] FALSE   FALSE   FALSE
## [298,] FALSE   FALSE   FALSE
## [299,] FALSE   FALSE   FALSE
## [300,] FALSE   FALSE   FALSE
```

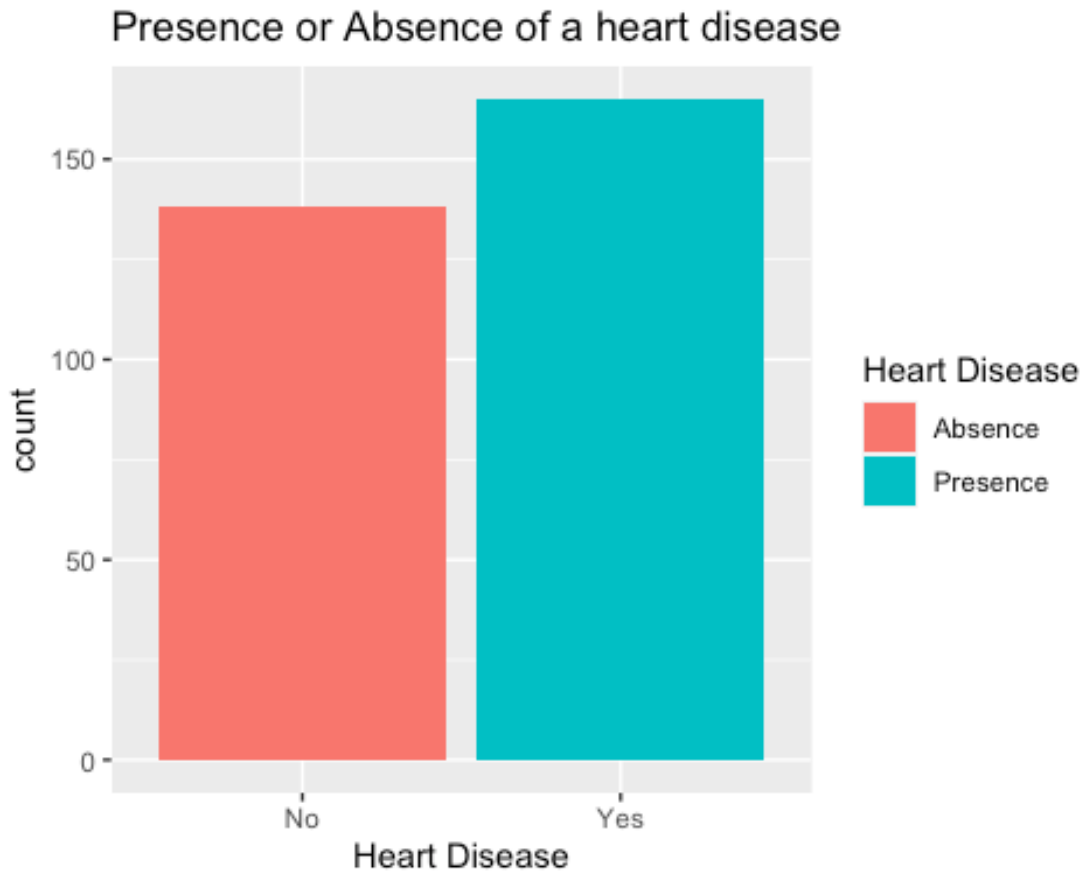
```
## [301,] FALSE    FALSE    FALSE
## [302,] FALSE    FALSE    FALSE
## [303,] FALSE    FALSE    FALSE

which(is.na(heart_disease2))

## integer(0)
```

As it shows no N/A value no need to remove or replace

5.Data Visualization

Now lets do some visualization , let us visualize how much people have the heart_disease

```
#To visualize and find how many people are infected with the heart_disease
, that is finding Target is yes or no
ggplot(heart_disease2,aes(target,fill=target))+
  geom_bar() +
  xlab("Heart Disease")+
  ylab("count")+
  ggtitle("Presence or Absence of a heart disease")+
  scale_fill_discrete(name='Heart Disease', labels=c("Absence","Presence")
)
```

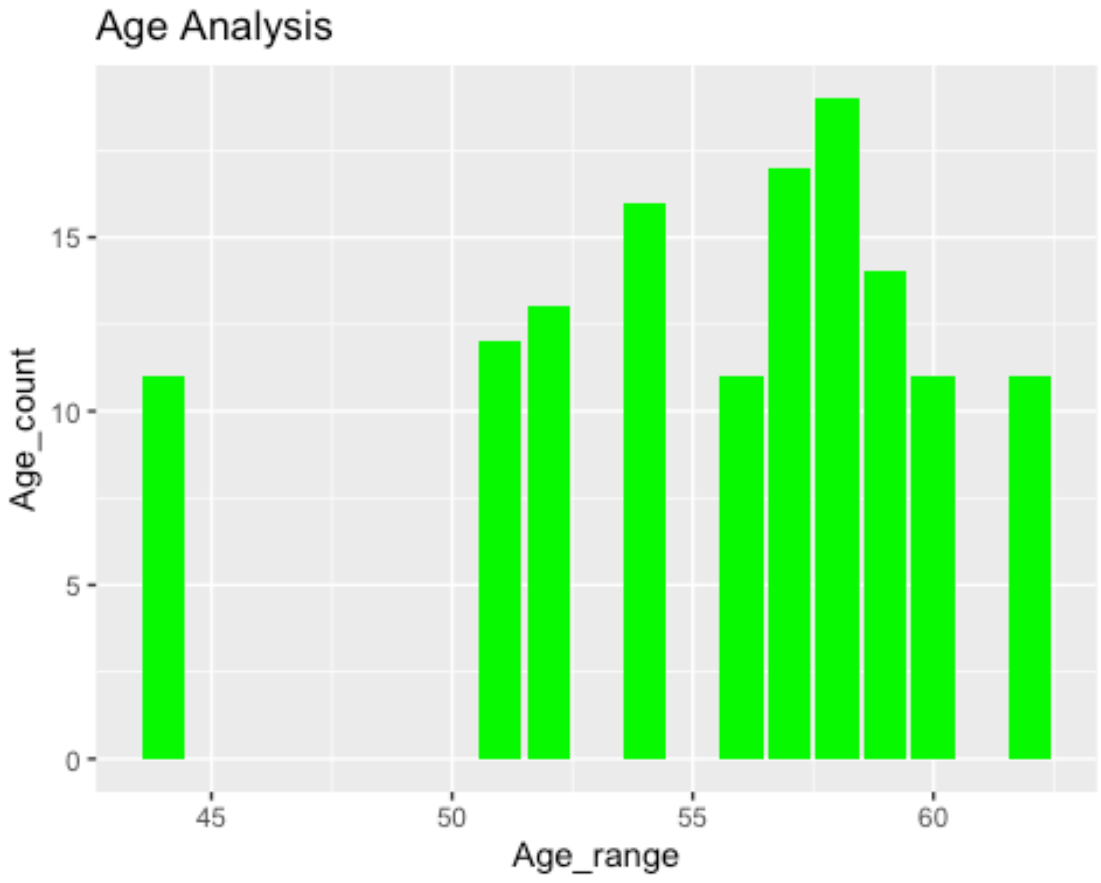


```
#Finding the proportions of presence and ansence of heart disease from Bar graph
prop.table(table(heart_disease2$target))
```

```
##
##        No       Yes
## 0.4554455 0.5445545
```

As we can see that proportion of presence of a heart disease is more than absence, we can conclude that the data is not that unbalanced , it is more bent towards presence of heart disease in the people

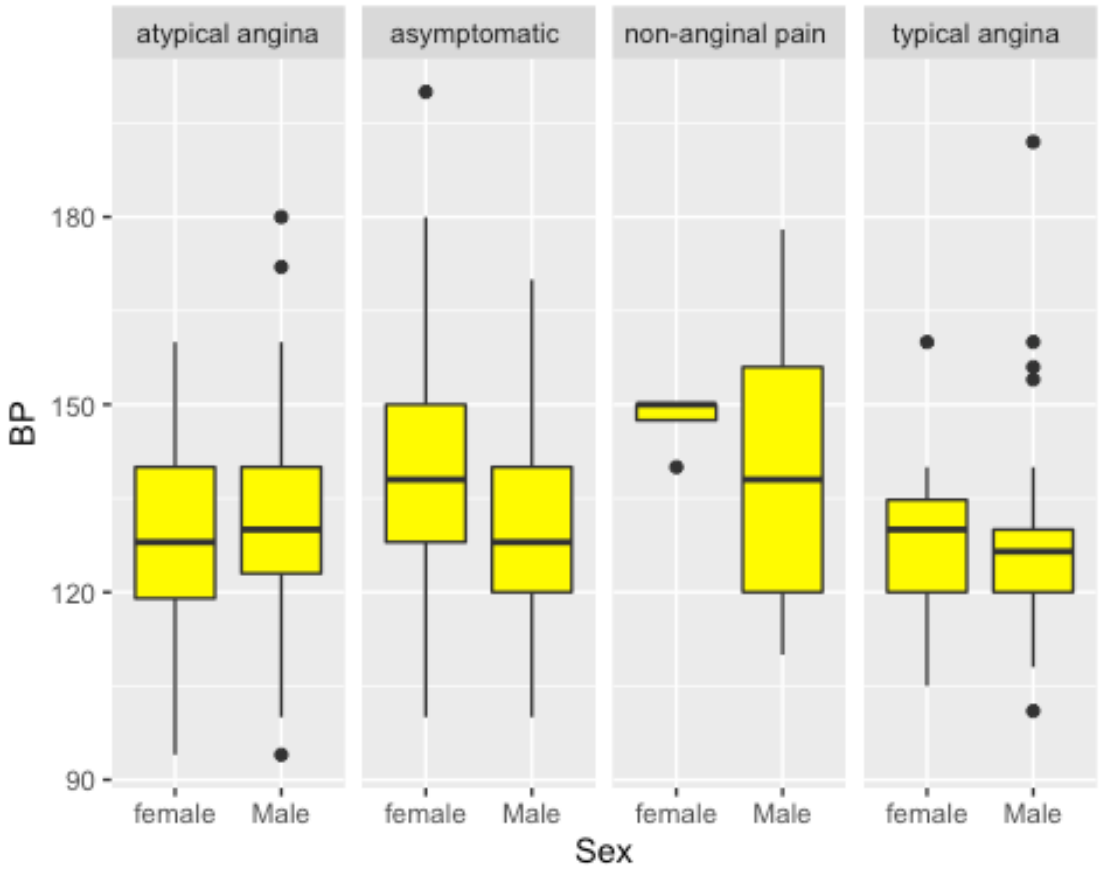
```
#Count the frequency of value of age
heart_disease2 %>%
  group_by(age) %>%
  count() %>%
  filter(n>10)%>%
  ggplot()+
  geom_col(aes(age,n), fill='green')+
  ggtitle("Age Analysis")+
  xlab("Age_range")+
  ylab("Age_count")
```



From the analysis we can see a major gap between the age 45 and 50 , this is a type of countplot(histgram plot for categorical variables) We can see how many people are falling in

the age_range(in x axis) and the count will be in y axis if we check the age_range between 55 and 60 we can see that a lot of patients have the heart disease, which might be because of their late stage in life (age)

```
#Compare blood pressure(trestbps) across cheast pain (cp)
heart_disease2 %>%
  ggplot(aes(x=sex,y=trestbps))+
  geom_boxplot(fill="yellow")+
  xlab("Sex")+
  ylab("BP")+
  facet_grid(~cp)
```



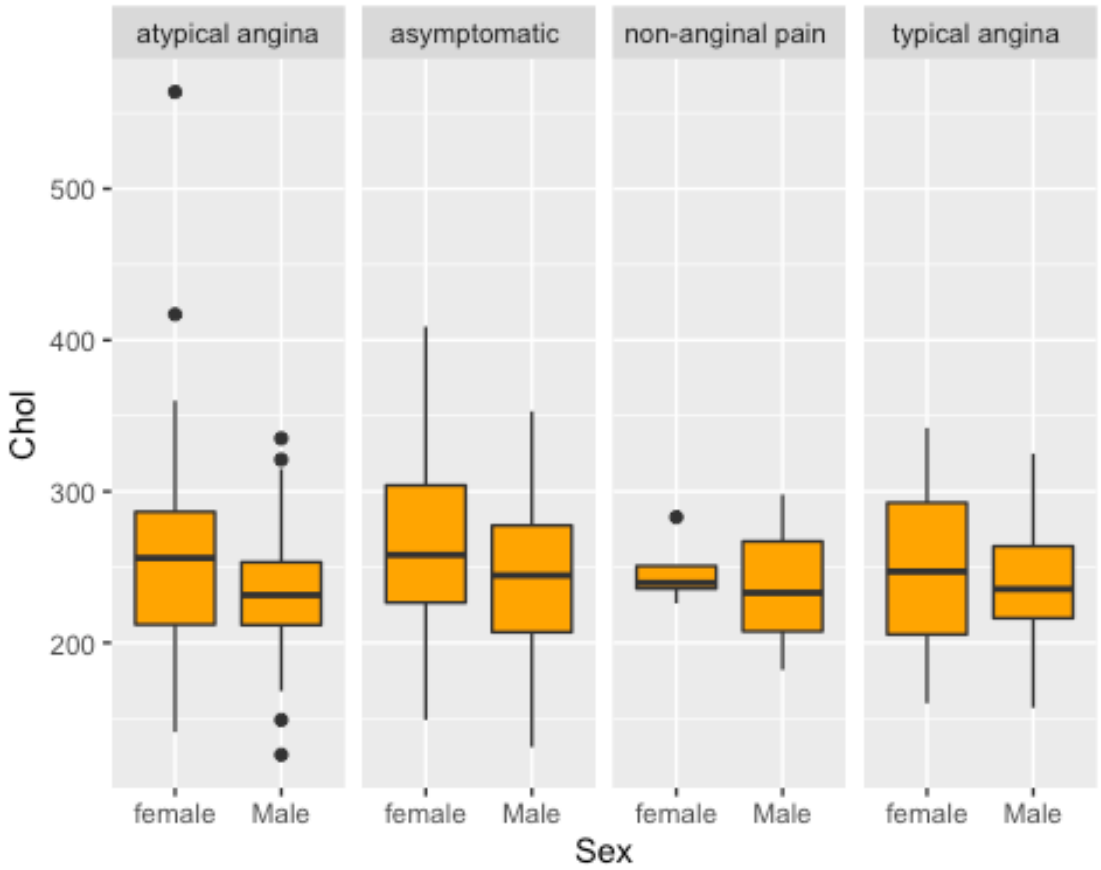
Now we have transformed the dataset and given various different names for the cp attribute(atypical angina, asymptomatic, non anignal pain, typical anigna) and we have mapped the number of males and females having those cp symptoms We notice the outliers here (the dots in the graph) atypical angina - female = 0 outlier ; male= 3 outliers asymptomatic - female = 1 outlier ; male= 0 outlier non anignal pain- female = 1 outlier ; male= 0 outlier typical anigna - female = 1 outlier ; male= 5 outliers

SO , what we notice from these outiers ? atypical angina - female = 0 outlier ; male= 3 outliers - males have more/less than normal BP asymptomatic - female = 1 outlier ; male= 0 outlier - in this females have more than normal BP non anignal pain- female = 1 outlier ;

male= 0 outlier - in this females have less than normal BP typical anigna - female = 1 outlier ; male= 5 outliers - female/male have more than BP value some males have less BP value

Now this was the visualization of the one of the feature/attribute with respect to the chest pain (cp), we can perform this with other attributes too

```
#Compare cholestrol(chol) across cheast pain (cp)
heart_disease2 %>%
  ggplot(aes(x=sex,y=chol))+
  geom_boxplot(fill="orange")+
  xlab("Sex")+
  ylab("Chol")+
  facet_grid(~cp)
```

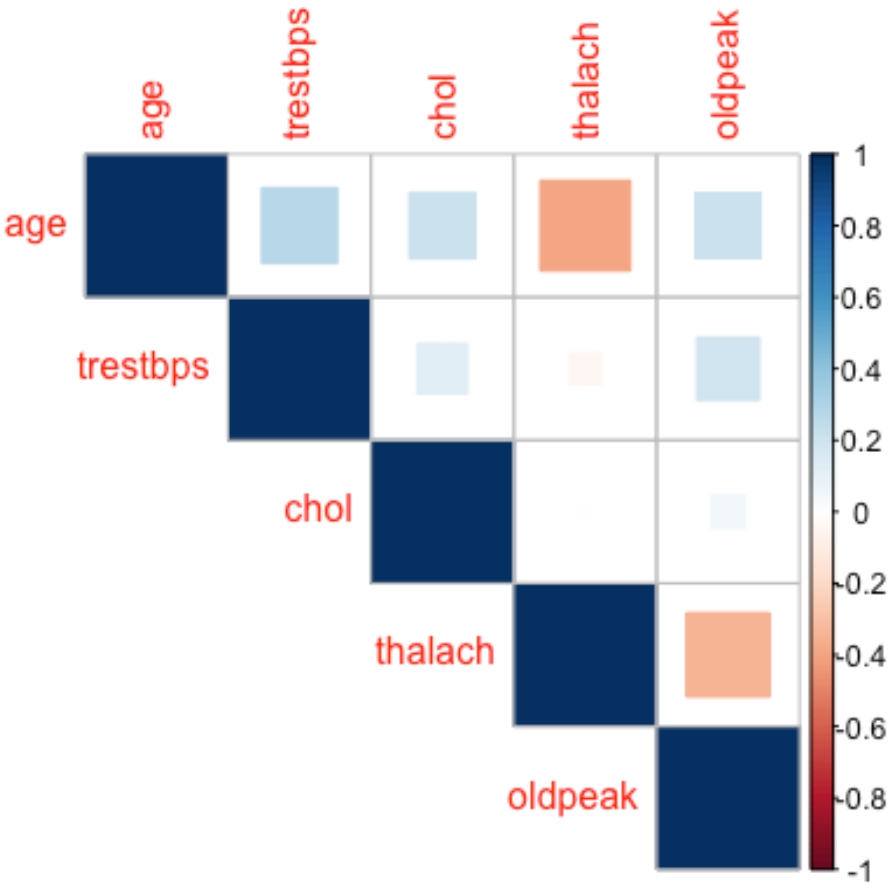


6.Correlation

```
cor_heart<-cor(heart_disease2[,10:14])
cor_heart
```

```
##                 age    trestbps         chol      thalach     oldpeak
## age       1.0000000  0.27935091  0.213677957 -0.398521938  0.21001257
## trestbps  0.2793509  1.00000000  0.123174207 -0.046697728  0.19321647
## chol      0.2136780  0.12317421  1.000000000 -0.009939839  0.05395192
## thalach  -0.3985219 -0.04669773 -0.009939839  1.000000000 -0.34418695
## oldpeak   0.2100126  0.19321647  0.053951920 -0.344186948  1.00000000
```

```
corrplot(cor_heart, method="square", type='upper')
```



We can notice the correlation between the different variables , only containing numerical variables The dark blue color shows high correlation between the variables The light blue squares shows that its a fair correlation between the variables The light red shows the negative correlation

If we check the correlation table we can see how some variables like age and age having correlation value 1 are showing highest value and some show negative values (thalach and age)

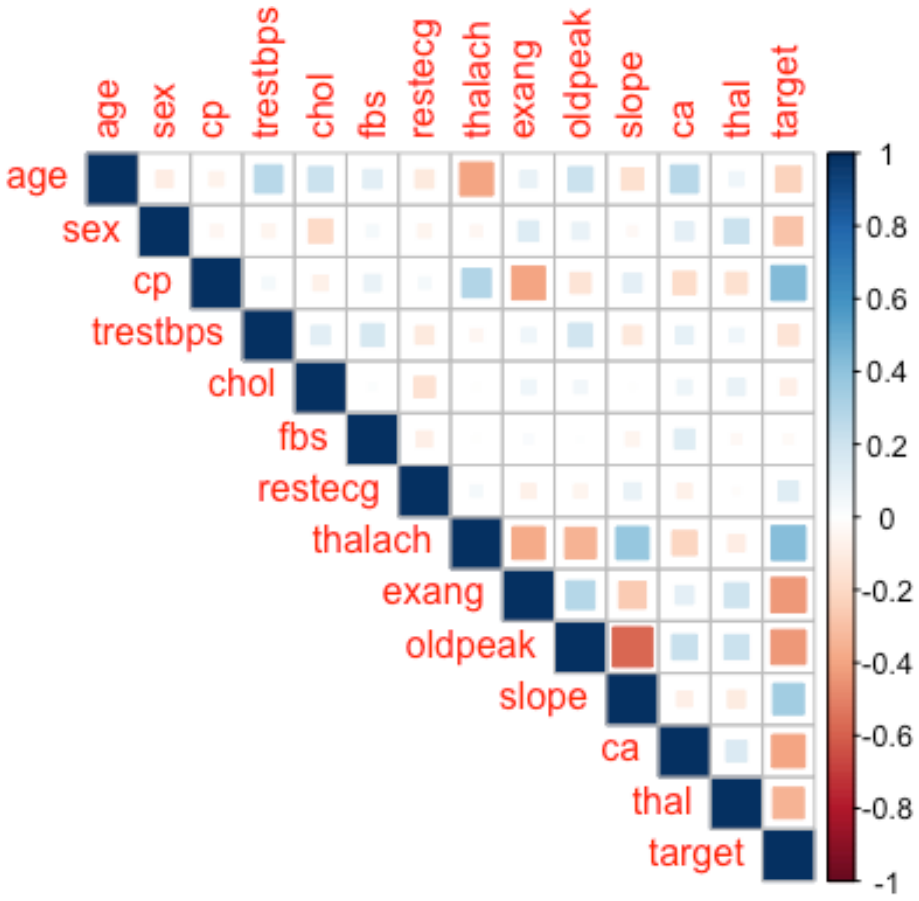
If we want to see the correlation between all the 14 variables we can use our scaled dataset with all the numerical values

```
# Using all the variables in dataset scaled and checking the correlation b
etween all of them
cor_heart<-cor(scaled[,1:14])
cor_heart
```

```
##                  age         sex          cp    trestbps         chol
## age       1.00000000 -0.09844660 -0.06865302  0.27935091  0.213677957
## sex      -0.09844660  1.00000000 -0.04935288 -0.05676882 -0.197912174
## cp       -0.06865302 -0.04935288  1.00000000  0.04760776 -0.076904391
## trestbps  0.27935091 -0.05676882  0.04760776  1.00000000  0.123174207
```

```
## chol      0.21367796 -0.19791217 -0.07690439  0.12317421  1.000000000
## fbs       0.12130765  0.04503179  0.09444403  0.17753054  0.013293602
## restecg  -0.11621090 -0.05819627  0.04442059 -0.11410279 -0.151040078
## thalach  -0.39852194 -0.04401991  0.29576212 -0.04669773 -0.009939839
## exang     0.09680083  0.14166381 -0.39428027  0.06761612  0.067022783
## oldpeak   0.21001257  0.09609288 -0.14923016  0.19321647  0.053951920
## slope    -0.16881424 -0.03071057  0.11971659 -0.12147458 -0.004037770
## ca        0.27632624  0.11826141 -0.18105303  0.10138899  0.070510925
## thal      0.06800138  0.21004110 -0.16173557  0.06220989  0.098802993
## target   -0.22543872 -0.28093658  0.43379826 -0.14493113 -0.085239105
##                  fbs     restecg      thalach       exang      oldpeak
## age       0.121307648 -0.11621090 -0.398521938  0.09680083  0.210012567
## sex       0.045031789 -0.05819627 -0.044019908  0.14166381  0.096092877
## cp        0.094444035  0.04442059  0.295762125 -0.39428027 -0.149230158
## trestbps  0.177530542 -0.11410279 -0.046697728  0.06761612  0.193216472
## chol      0.013293602 -0.15104008 -0.009939839  0.06702278  0.053951920
## fbs       1.000000000 -0.08418905 -0.008567107  0.02566515  0.005747223
## restecg  -0.084189054  1.00000000  0.044123444 -0.07073286 -0.058770226
## thalach  -0.008567107  0.04412344  1.000000000 -0.37881209 -0.344186948
## exang     0.025665147 -0.07073286 -0.378812094  1.00000000  0.288222808
## oldpeak   0.005747223 -0.05877023 -0.344186948  0.28822281  1.000000000
## slope    -0.059894178  0.09304482  0.386784410 -0.25774837 -0.577536817
## ca        0.137979327 -0.07204243 -0.213176928  0.11573938  0.222682322
## thal     -0.032019339 -0.01198140 -0.096439132  0.20675379  0.210244126
## target   -0.028045760  0.13722950  0.421740934 -0.43675708 -0.430696002
##                slope          ca        thal      target
## age      -0.16881424  0.27632624  0.06800138 -0.22543872
## sex      -0.03071057  0.11826141  0.21004110 -0.28093658
## cp        0.11971659 -0.18105303 -0.16173557  0.43379826
## trestbps -0.12147458  0.10138899  0.06220989 -0.14493113
## chol     -0.00403777  0.07051093  0.09880299 -0.08523911
## fbs      -0.05989418  0.13797933 -0.03201934 -0.02804576
## restecg   0.09304482 -0.07204243 -0.01198140  0.13722950
## thalach   0.38678441 -0.21317693 -0.09643913  0.42174093
## exang    -0.25774837  0.11573938  0.20675379 -0.43675708
## oldpeak  -0.57753682  0.22268232  0.21024413 -0.43069600
## slope     1.00000000 -0.08015521 -0.10476379  0.34587708
## ca       -0.08015521  1.00000000  0.15183213 -0.39172399
## thal     -0.10476379  0.15183213  1.00000000 -0.34402927
## target    0.34587708 -0.39172399 -0.34402927  1.00000000
```

```
corrplot(cor_heart, method="square", type='upper')
```



7.Let's start grouping patients

Now that we have cleaned, transformed, scaled and visualized the data, we can start the clustering process. For the k-means algorithm, it is necessary to select the number of clusters in advance. It is also important to make sure that our results are reproducible when conducting a statistical analysis. This means that when someone runs our code on the same data, they will get the same results. Because we are doing an analysis that has a random aspect, it is necessary to set a seed to ensure reproducibility. Reproducibility is especially important because doctors will potentially use our results to treat patients. It is vital that other analysts see where the groups come from and can verify the results.

```
# Set the seed so that results are reproducible
seed_val <- 10
set.seed(seed_val)

# Select a number of clusters
k <- 5

# Run the k-means algorithm
#first_clust <- kmeans(heart_disease2, centers = k, nstart = 1)
#Find the centres of the clusters
#first_clust$centers
```

```
 #How many patients are in each cluster?
#first_clust$size
```

Since we used our heart_disease2 dataset with non numerical values so that's why its showing this error K- means cannot handle non-numerical value So for this part we will use the numerical dataset which we scaled before named scaled

```
# Run the k-means algorithm
first_clust <- kmeans(scaled, centers = k, nstart = 1)
# Find the centres of the clusters
first_clust$centers
```

```
##          age        sex         cp   trestbps        chol        fbs
restecg
## 1  0.8074853 -0.4359734  0.9234058  0.5323390  0.48843540  0.7621751 -0
.3195773
## 2 -0.2605495  0.6798805  0.1679703 -0.3696749 -0.33816188 -0.1214260  0
.1968703
## 3  0.7537666 -0.1004369 -0.3790884  1.1704633  0.30067977  0.2636439 -0
.1397851
## 4  0.2101965  0.3688845 -0.8948367 -0.1512559  0.06592868 -0.0507585 -0
.1498070
## 5 -0.6904341 -0.4158844  0.1969048 -0.3588735 -0.20870270 -0.3870785  0
.2096473
##      thalach       exang    oldpeak      slope         ca       thal
target
## 1 -0.0282410 -0.39747738 -0.2166991  0.1633492 -0.2829759 -0.4467448  0
.5922713
## 2  0.4673188 -0.39673051 -0.3710630  0.3768730  0.2646444  0.7200254 -0
.2475806
## 3 -0.7492642 -0.05045264  1.3360938 -0.9430923  1.0351059  0.1808187 -0
.9701577
## 4 -0.8786284  1.30971384  0.5550494 -0.5304484  0.2646444  0.4584065 -0
.9754401
## 5  0.6396378 -0.49167883 -0.5362123  0.4050400 -0.5676053 -0.5989497  0
.8916925
```

```
# Show the clusters
first_clust$cluster
```

```
##   [1] 1 5 5 5 5 5 1 2 1 5 5 5 5 1 1 5 5 1 5 1 2 5 5 1 2 1 1 5 1 1 5 2 5
1 1 5 1
##  [38] 2 1 1 1 5 4 5 5 5 5 5 5 5 5 1 2 5 1 5 5 5 5 5 1 2 5 5 1 5 5 5 5 5
2 2 5 5
##  [75] 5 5 1 5 5 2 5 5 5 1 5 1 2 2 5 5 2 2 2 1 5 4 1 2 5 1 5 3 5 2 5 1 1
5 5 5 1
## [112] 2 1 2 5 5 5 2 5 5 3 5 5 5 5 5 5 1 5 1 5 5 5 5 5 5 1 1 4 4 5 5 5 5
1 1 5 1
## [149] 5 5 1 5 1 1 5 5 5 5 2 2 5 5 5 2 2 3 4 3 4 4 1 2 2 2 4 4 4 1 4 4 4
3 1 2 3
## [186] 2 4 4 2 2 4 4 4 4 3 3 3 2 4 2 2 4 4 3 3 2 4 3 2 2 2 4 2 4 4 4 3 4
3 4 3 4
```

```
## [223] 1 3 4 4 4 4 1 4 2 3 4 4 3 4 2 3 4 2 3 4 3 4 4 2 4 3 2 3 3 4 3 4 1
4 4 4 4
## [260] 2 3 2 4 4 4 4 3 2 4 4 2 3 4 2 4 2 3 2 1 4 4 4 1 2 4 4 2 2 4 4 2 3
3 2 4 4
## [297] 4 3 4 2 3 4 5
```

```
# How many patients are in each cluster?
first_clust$size
```

```
## [1] 50 57 33 69 94
```

first_clust$centers Thi shows the centre points of the clusters whch are formed

first_clust$cluster Here if we see the observation of assigned cluster we see clusters have been assigned from 1 to 5 and we can note what observation has been classified to which cluster

8.Plotting the clusters

```
#plotting thr groups
#plot(first_clust[first_clust$cluster==1],
#col="red",
#xlim = c(min(first_clust[,1]),max(first_clust[,1])),
#ylim = c(min(first_clust[,5]),max(first_clust[,5])))
#points(first_clust[first_clust$cluster==2],
#col="blue")
#points(first_clust[first_clust$cluster==3],
#col="green")
#points(first_clust[first_clust$cluster==4],
#col="yellow")
#points(first_clust[first_clust$cluster==5],
#col="pink")

#plotting the centre of the plot
#points(first_clust$centers,pch=2,col="green")
```

9.Another round of k-means

Because the k-means algorithm initially selects the cluster centers by randomly selecting points, different iterations of the algorithm can result in different clusters. If the algorithm is genuinely grouping similar observations (as opposed to clustering noise), then cluster assignments will be somewhat robust between various iterations of the algorithm. With regards to the heart disease data, this would mean that the same patients would be grouped even when the algorithm is initialized at different random points. If patients are not in similar clusters with various algorithm runs, then the clustering method is not picking up on meaningful relationships between patients. We're going to explore how the patients are grouped with another iteration of the k-means algorithm. We will then be able to compare the resulting groups of patients.

```
# Set the seed so that results are reproducible
seed_val <- 38
```

```r
set.seed(seed_val)

# Select a number of clusters
k <- 5
# Run the k-means algorithm
second_clust <- kmeans(scaled, centers = k, nstart = 1)
# Find the centres of the clusters
second_clust$centers
```

```
##          age        sex         cp     trestbps       chol         fbs
## 1  0.4332974 -1.3297465  0.2319189 -0.001173886  0.4962325 -0.14957067
## 2  0.1798773  0.6185699  0.0596624  0.032856251 -0.2096058  0.38517758
## 3  0.3688103  0.1677972 -0.8488789  0.225553134  0.1367575 -0.06601626
## 4 -0.9297693  0.3175903  0.1074804 -0.450987128 -0.3538816 -0.34402458
## 5  0.1385872  0.5725869  1.2431570  0.345032977 -0.2178056  0.70602653
##       restecg     thalach      exang    oldpeak      slope         ca
## 1 -0.06843902  0.07709327 -0.4927571 -0.4264664  0.3822959 -0.3717624
## 2  0.08248545  0.38214686 -0.3305788 -0.4229120  0.5110880  1.4101767
## 3 -0.11817549 -0.95994844  0.9735284  0.8467354 -0.7033633  0.4535556
## 4  0.32945276  0.76497897 -0.4466838 -0.6425920  0.6375384 -0.5989497
## 5 -0.33859381  0.18350167 -0.2165472  0.4159043 -0.7291802 -0.4932230
##          thal     target
## 1 -0.53799936  0.8175583
## 2  0.56120315 -0.5188896
## 3  0.39734797 -1.0005315
## 4 -0.23633143  0.6526718
## 5 -0.06293129  0.1111501
```

```r
# Show the clusters
second_clust$cluster
```

```
##   [1] 5 5 4 4 1 4 1 4 5 4 4 1 4 5 1 1 1 5 4 1 4 4 4 5 4 1 5 4 1 5 4 4 4
5 5 5 1
##  [38] 5 1 1 1 4 3 1 4 4 4 4 1 1 1 1 2 4 1 4 4 4 4 1 1 4 4 4 5 4 4 1 4 1
5 4 4 4
##  [75] 4 1 5 4 4 5 4 4 1 5 1 1 2 4 1 1 2 4 2 1 4 3 1 2 4 2 4 5 1 5 4 1 5
1 1 1 1
## [112] 2 1 4 4 4 5 5 4 1 1 4 4 1 4 4 4 1 1 1 1 1 4 4 1 1 1 5 3 3 1 4 4 1
1 1 1 1
## [149] 4 4 1 1 5 1 4 1 4 4 2 4 4 1 4 2 2 3 3 3 3 3 5 5 5 2 3 3 2 1 3 3 3
3 1 5 3
## [186] 4 3 3 2 4 3 3 3 3 5 3 5 2 3 2 4 3 3 5 3 2 3 3 2 2 2 3 4 3 3 3 3 3
3 2 3 3
## [223] 5 3 3 3 3 3 5 3 4 2 3 3 3 3 2 2 2 4 3 3 3 3 3 4 3 3 2 2 3 2 3 3 5
3 3 3 3
## [260] 5 3 4 3 3 3 3 3 2 3 3 4 5 3 2 3 2 3 2 2 3 3 5 5 4 3 3 2 2 3 3 2 3
3 5 3 3
## [297] 3 3 3 5 3 3 1
```

```r
# How many patients are in each cluster?
second_clust$size
```

```
## [1] 63 35 88 77 40
```

10.Plotting the clusters for second iteration

```
#plotting thr groups
#plot(first_clust[first_clust$cluster==1],
#col="red",
#xlim = c(min(first_clust[,1]),max(first_clust[,1])),
#ylim = c(min(first_clust[,5]),max(first_clust[,5])))
#points(first_clust[first_clust$cluster==2],
#col="blue")
#points(first_clust[first_clust$cluster==3],
#col="green")
#points(first_clust[first_clust$cluster==4],
#col="yellow")
#points(first_clust[first_clust$cluster==5],
#col="pink")

#plotting the centre of the plot
#points(first_clust$centers,pch=2,col="green")
```

11.Comparing patient clusters

It is important that the clusters are stable. Even though the algorithm begins by randomly initializing the cluster centers, if the k-means algorithm is the right choice for the data, then different initialization of the algorithm will result in similar clusters. The clusters from different iterations may not be the same, but the clusters should be roughly the same size and have similar distributions of variables. If there is a lot of change in clusters between different iterations of the algorithm, then k-means clustering is not the right choice for the data. It is not possible to validate that the clusters obtained from the algorithm are accurate because there is no patient labeling. Thus, it is necessary to examine how the clusters change between different iterations of the algorithm. We're going to use some visualizations to get an idea of the cluster stabilities. That way we can see how certain patient characteristics may have been used to group patients together.

```
# Add cluster assignments to the data
heart_disease2["first_clust"] <- first_clust$cluster
heart_disease2["second_clust"] <- second_clust$cluster
heart_disease2
```

```
##     target    sex   fbs exang                cp              restecg sl
ope ca
## 1      Yes   Male  >120    No non-anginal pain                Normal
0  0
## 2      Yes   Male <=120    No  atypical angina              Abnormal
0  0
## 3      Yes female <=120    No   typical angina                Normal
2  0
## 4      Yes   Male <=120    No   typical angina              Abnormal
2  0
## 5      Yes female <=120   Yes     asymptomatic              Abnormal
```

```
2  0
## 6      Yes   Male <=120    No     asymptomatic              Abnormal
1  0
## 7      Yes female <=120    No   typical angina                Normal
1  0
## 8      Yes   Male <=120    No   typical angina              Abnormal
2  0
## 9      Yes   Male  >120    No  atypical angina              Abnormal
2  0
## 10     Yes   Male <=120    No  atypical angina              Abnormal
2  0
## 11     Yes   Male <=120    No     asymptomatic              Abnormal
2  0
## 12     Yes female <=120    No  atypical angina              Abnormal
2  0
## 13     Yes   Male <=120    No   typical angina              Abnormal
2  0
## 14     Yes   Male <=120   Yes non-anginal pain                Normal
1  0
## 15     Yes female  >120    No non-anginal pain                Normal
2  0
## 16     Yes female <=120    No  atypical angina              Abnormal
1  0
## 17     Yes female <=120    No  atypical angina              Abnormal
2  0
## 18     Yes female <=120    No non-anginal pain              Abnormal
0  0
## 19     Yes   Male <=120    No     asymptomatic              Abnormal
2  0
## 20     Yes female <=120    No non-anginal pain              Abnormal
2  2
## 21     Yes   Male <=120    No     asymptomatic              Abnormal
1  0
## 22     Yes   Male <=120   Yes  atypical angina              Abnormal
2  0
## 23     Yes   Male <=120    No     asymptomatic              Abnormal
2  0
## 24     Yes   Male  >120   Yes  atypical angina              Abnormal
1  0
## 25     Yes   Male <=120   Yes non-anginal pain              Abnormal
2  0
## 26     Yes female <=120    No   typical angina              Abnormal
2  2
## 27     Yes   Male  >120    No  atypical angina              Abnormal
2  0
## 28     Yes   Male <=120    No  atypical angina              Abnormal
2  0
## 29     Yes female  >120    No  atypical angina                Normal
2  1
## 30     Yes   Male  >120    No  atypical angina                Normal
0  0
```

```
## 31     Yes female <=120    No   typical angina              Abnormal
2  1
## 32     Yes   Male <=120    No     asymptomatic              Abnormal
2  0
## 33     Yes   Male <=120    No   typical angina                Normal
2  0
## 34     Yes   Male <=120    No  atypical angina                Normal
0  1
## 35     Yes   Male <=120   Yes non-anginal pain                Normal
2  1
## 36     Yes female <=120   Yes  atypical angina                Normal
0  0
## 37     Yes female  >120    No  atypical angina              Abnormal
2  0
## 38     Yes   Male <=120    No  atypical angina                Normal
2  0
## 39     Yes female <=120    No  atypical angina              Abnormal
2  0
## 40     Yes female <=120    No  atypical angina                Normal
2  0
## 41     Yes female <=120    No  atypical angina                Normal
2  1
## 42     Yes   Male <=120    No   typical angina                Normal
1  0
## 43     Yes   Male <=120   Yes     asymptomatic                Normal
1  0
## 44     Yes female <=120    No     asymptomatic                Normal
1  0
## 45     Yes   Male <=120    No  atypical angina                Normal
2  0
## 46     Yes   Male <=120    No   typical angina              Abnormal
2  0
## 47     Yes   Male <=120    No  atypical angina                Normal
2  0
## 48     Yes   Male <=120    No  atypical angina                Normal
2  0
## 49     Yes female <=120    No  atypical angina                Normal
2  0
## 50     Yes female <=120    No     asymptomatic                Normal
2  0
## 51     Yes female <=120    No  atypical angina                Normal
2  0
## 52     Yes   Male <=120    No     asymptomatic                Normal
1  0
## 53     Yes   Male <=120    No  atypical angina              Abnormal
1  3
## 54     Yes female <=120    No  atypical angina              Abnormal
1  0
## 55     Yes female <=120    No  atypical angina                Normal
2  0
## 56     Yes   Male <=120    No   typical angina              Abnormal
```

```
2  1
## 57     Yes   Male <=120    No     asymptomatic               Normal
2  0
## 58     Yes   Male <=120    No     asymptomatic               Normal
2  0
## 59     Yes   Male <=120    No non-anginal pain               Normal
2  0
## 60     Yes female <=120    No     asymptomatic               Normal
2  1
## 61     Yes female  >120    No  atypical angina               Normal
2  1
## 62     Yes   Male <=120    No   typical angina             Abnormal
2  0
## 63     Yes   Male <=120    No non-anginal pain               Normal
1  0
## 64     Yes   Male <=120    No   typical angina             Abnormal
1  0
## 65     Yes   Male  >120    No  atypical angina               Normal
2  0
## 66     Yes female <=120    No     asymptomatic             Abnormal
2  0
## 67     Yes   Male <=120   Yes  atypical angina             Abnormal
1  0
## 68     Yes female <=120    No   typical angina               Normal
1  0
## 69     Yes   Male <=120    No   typical angina             Abnormal
2  0
## 70     Yes female <=120    No     asymptomatic             Abnormal
2  0
## 71     Yes   Male <=120    No  atypical angina               Normal
1  0
## 72     Yes   Male <=120   Yes  atypical angina             Abnormal
2  1
## 73     Yes   Male <=120    No   typical angina               Normal
2  0
## 74     Yes   Male <=120   Yes     asymptomatic               Normal
2  0
## 75     Yes female <=120    No  atypical angina             Abnormal
1  0
## 76     Yes female <=120    No   typical angina               Normal
1  0
## 77     Yes   Male  >120    No  atypical angina               Normal
1  0
## 78     Yes   Male <=120   Yes   typical angina             Abnormal
2  0
## 79     Yes   Male  >120    No   typical angina             Abnormal
2  0
## 80     Yes   Male <=120   Yes  atypical angina               Normal
1  0
## 81     Yes   Male <=120    No  atypical angina             Abnormal
2  0
```

```
## 82     Yes   Male <=120    No   typical angina               Normal
2  0
## 83     Yes female <=120    No  atypical angina             Abnormal
2  1
## 84     Yes   Male  >120    No non-anginal pain             Abnormal
1  0
## 85     Yes female <=120    No     asymptomatic               Normal
1  0
## 86     Yes female <=120    No  atypical angina               Normal
1  0
## 87     Yes   Male <=120    No  atypical angina             Abnormal
2  1
## 88     Yes   Male  >120    No   typical angina             Abnormal
2  0
## 89     Yes female <=120    No  atypical angina             Abnormal
1  0
## 90     Yes female <=120    No     asymptomatic               Normal
1  0
## 91     Yes   Male  >120    No  atypical angina             Abnormal
2  2
## 92     Yes   Male <=120   Yes     asymptomatic             Abnormal
2  0
## 93     Yes   Male <=120    No  atypical angina             Abnormal
2  4
## 94     Yes female  >120   Yes   typical angina               Normal
2  1
## 95     Yes female <=120    No   typical angina             Abnormal
1  0
## 96     Yes   Male <=120   Yes     asymptomatic               Normal
2  0
## 97     Yes female <=120    No     asymptomatic               Normal
1  0
## 98     Yes   Male  >120    No     asymptomatic             Abnormal
2  3
## 99     Yes   Male <=120    No  atypical angina             Abnormal
2  1
## 100    Yes   Male  >120    No  atypical angina               Normal
2  3
## 101    Yes   Male <=120    No non-anginal pain               Normal
2  2
## 102    Yes   Male <=120    No non-anginal pain               Normal
0  0
## 103    Yes female <=120    No   typical angina             Abnormal
2  2
## 104    Yes   Male  >120    No  atypical angina             Abnormal
0  0
## 105    Yes   Male <=120    No  atypical angina             Abnormal
2  0
## 106    Yes female <=120    No  atypical angina               Normal
1  0
## 107    Yes   Male  >120    No non-anginal pain               Normal
```

```
1  1
## 108    Yes female <=120   Yes     asymptomatic                Normal
1  0
## 109    Yes female <=120    No   typical angina              Abnormal
2  0
## 110    Yes female <=120    No     asymptomatic                Normal
2  0
## 111    Yes female <=120   Yes     asymptomatic              Abnormal
2  0
## 112    Yes   Male  >120    No  atypical angina              Abnormal
2  1
## 113    Yes female <=120    No  atypical angina              Abnormal
2  0
## 114    Yes   Male <=120    No     asymptomatic              Abnormal
2  0
## 115    Yes   Male <=120    No   typical angina              Abnormal
2  0
## 116    Yes female <=120    No  atypical angina              Abnormal
2  0
## 117    Yes   Male <=120    No  atypical angina                Normal
1  0
## 118    Yes   Male <=120    No non-anginal pain                Normal
1  0
## 119    Yes female <=120    No   typical angina              Abnormal
2  0
## 120    Yes female <=120   Yes     asymptomatic                Normal
1  0
## 121    Yes female <=120    No     asymptomatic              Abnormal
1  2
## 122    Yes   Male <=120    No     asymptomatic                Normal
2  0
## 123    Yes female <=120   Yes  atypical angina                Normal
2  0
## 124    Yes female <=120    No  atypical angina                Normal
2  0
## 125    Yes female <=120    No  atypical angina              Abnormal
2  0
## 126    Yes female <=120    No   typical angina              Abnormal
2  0
## 127    Yes   Male <=120    No     asymptomatic              Abnormal
2  0
## 128    Yes female <=120    No  atypical angina              Abnormal
2  1
## 129    Yes female <=120    No  atypical angina                Normal
1  0
## 130    Yes female <=120   Yes   typical angina                Normal
2  1
## 131    Yes female <=120    No  atypical angina              Abnormal
2  1
## 132    Yes female <=120    No   typical angina              Abnormal
1  0
```

```
## 133    Yes   Male <=120    No   typical angina              Abnormal
2  0
## 134    Yes   Male <=120    No   typical angina              Abnormal
2  0
## 135    Yes female <=120    No   typical angina              Abnormal
2  0
## 136    Yes female <=120    No     asymptomatic              Abnormal
2  0
## 137    Yes female  >120    No  atypical angina              Abnormal
2  0
## 138    Yes   Male  >120    No   typical angina                Normal
2  0
## 139    Yes   Male <=120   Yes     asymptomatic              Abnormal
1  0
## 140    Yes   Male <=120   Yes     asymptomatic              Abnormal
1  1
## 141    Yes female <=120    No  atypical angina                Normal
2  0
## 142    Yes   Male <=120    No     asymptomatic              Abnormal
1  0
## 143    Yes female <=120    No  atypical angina              Abnormal
1  0
## 144    Yes female <=120    No     asymptomatic              Abnormal
2  2
## 145    Yes female <=120    No  atypical angina  Probable or Definite
1  0
## 146    Yes   Male <=120    No   typical angina                Normal
2  0
## 147    Yes female <=120    No  atypical angina              Abnormal
1  1
## 148    Yes female <=120    No non-anginal pain              Abnormal
2  0
## 149    Yes   Male <=120    No  atypical angina              Abnormal
2  0
## 150    Yes   Male <=120    No  atypical angina              Abnormal
2  0
## 151    Yes   Male <=120    No     asymptomatic                Normal
2  0
## 152    Yes female <=120    No     asymptomatic              Abnormal
1  0
## 153    Yes   Male <=120    No non-anginal pain                Normal
1  0
## 154    Yes female <=120    No  atypical angina                Normal
1  1
## 155    Yes female <=120    No  atypical angina              Abnormal
1  0
## 156    Yes female <=120    No     asymptomatic              Abnormal
1  0
## 157    Yes   Male <=120    No  atypical angina              Abnormal
2  0
## 158    Yes   Male <=120    No   typical angina              Abnormal
```

```
2  0
## 159    Yes   Male <=120    No   typical angina              Abnormal
1  4
## 160    Yes   Male <=120    No   typical angina                Normal
2  0
## 161    Yes   Male <=120    No   typical angina              Abnormal
0  0
## 162    Yes female <=120    No   typical angina              Abnormal
2  0
## 163    Yes   Male <=120    No   typical angina              Abnormal
2  0
## 164    Yes   Male <=120    No  atypical angina              Abnormal
2  4
## 165    Yes   Male <=120    No  atypical angina              Abnormal
2  4
## 166     No   Male <=120   Yes     asymptomatic                Normal
1  3
## 167     No   Male <=120   Yes     asymptomatic                Normal
1  2
## 168     No female <=120    No     asymptomatic                Normal
0  2
## 169     No   Male <=120    No     asymptomatic                Normal
1  1
## 170     No   Male  >120   Yes     asymptomatic                Normal
0  0
## 171     No   Male  >120   Yes  atypical angina                Normal
1  1
## 172     No   Male <=120    No   typical angina              Abnormal
0  0
## 173     No   Male <=120    No   typical angina                Normal
1  0
## 174     No   Male <=120    No  atypical angina                Normal
2  2
## 175     No   Male <=120   Yes     asymptomatic                Normal
1  2
## 176     No   Male <=120   Yes     asymptomatic                Normal
1  0
## 177     No   Male  >120   Yes     asymptomatic              Abnormal
2  2
## 178     No   Male <=120    No  atypical angina              Abnormal
2  0
## 179     No   Male <=120   Yes     asymptomatic                Normal
1  0
## 180     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 181     No   Male <=120   Yes     asymptomatic              Abnormal
1  1
## 182     No female <=120    No     asymptomatic                Normal
1  3
## 183     No female <=120    No     asymptomatic                Normal
2  0
```

```
## 184     No   Male <=120    No  atypical angina                Normal
1  1
## 185     No   Male <=120    No     asymptomatic                Normal
1  0
## 186     No   Male <=120    No     asymptomatic                Normal
2  1
## 187     No   Male <=120   Yes     asymptomatic              Abnormal
2  1
## 188     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 189     No   Male <=120    No  atypical angina              Abnormal
1  1
## 190     No   Male <=120    No     asymptomatic                Normal
2  0
## 191     No female <=120   Yes     asymptomatic              Abnormal
1  0
## 192     No   Male <=120   Yes     asymptomatic                Normal
1  3
## 193     No   Male <=120    No     asymptomatic              Abnormal
1  1
## 194     No   Male <=120   Yes     asymptomatic                Normal
1  2
## 195     No   Male <=120    No  atypical angina                Normal
1  0
## 196     No   Male <=120   Yes     asymptomatic                Normal
0  0
## 197     No   Male <=120    No  atypical angina              Abnormal
1  0
## 198     No   Male  >120    No     asymptomatic              Abnormal
1  2
## 199     No   Male <=120   Yes     asymptomatic              Abnormal
1  2
## 200     No   Male <=120    No     asymptomatic                Normal
2  2
## 201     No   Male <=120    No     asymptomatic                Normal
2  1
## 202     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 203     No   Male <=120   Yes     asymptomatic                Normal
2  0
## 204     No   Male  >120   Yes  atypical angina                Normal
1  0
## 205     No female <=120    No     asymptomatic                Normal
0  3
## 206     No   Male <=120   Yes     asymptomatic              Abnormal
2  1
## 207     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 208     No female <=120    No     asymptomatic                Normal
1  2
## 209     No   Male <=120    No  atypical angina              Abnormal
```

```
1  3
## 210     No   Male <=120   Yes     asymptomatic              Abnormal
2  1
## 211     No   Male <=120    No  atypical angina                Normal
1  1
## 212     No   Male <=120   Yes     asymptomatic              Abnormal
1  1
## 213     No   Male <=120    No     asymptomatic              Abnormal
1  0
## 214     No female <=120   Yes     asymptomatic                Normal
1  0
## 215     No   Male  >120   Yes     asymptomatic                Normal
1  1
## 216     No female  >120   Yes     asymptomatic                Normal
1  0
## 217     No female <=120    No  atypical angina              Abnormal
1  1
## 218     No   Male  >120   Yes     asymptomatic                Normal
2  3
## 219     No   Male <=120    No     asymptomatic                Normal
1  1
## 220     No   Male  >120   Yes     asymptomatic                Normal
2  2
## 221     No female <=120    No     asymptomatic                Normal
1  3
## 222     No   Male <=120   Yes     asymptomatic              Abnormal
0  0
## 223     No   Male  >120    No non-anginal pain                Normal
1  1
## 224     No female  >120   Yes     asymptomatic                Normal
0  2
## 225     No   Male <=120   Yes     asymptomatic              Abnormal
1  1
## 226     No   Male <=120   Yes     asymptomatic              Abnormal
0  0
## 227     No   Male <=120    No   typical angina                Normal
1  1
## 228     No   Male <=120   Yes     asymptomatic              Abnormal
1  0
## 229     No   Male <=120    No non-anginal pain                Normal
1  0
## 230     No   Male <=120   Yes  atypical angina              Abnormal
1  0
## 231     No   Male <=120    No  atypical angina              Abnormal
2  0
## 232     No   Male  >120    No     asymptomatic                Normal
1  3
## 233     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 234     No   Male <=120   Yes     asymptomatic                Normal
0  1
```

```
## 235     No   Male <=120    No     asymptomatic                Normal
1  3
## 236     No   Male <=120   Yes     asymptomatic              Abnormal
2  0
## 237     No   Male <=120    No     asymptomatic                Normal
2  2
## 238     No   Male <=120    No     asymptomatic                Normal
1  2
## 239     No   Male <=120   Yes     asymptomatic                Normal
2  3
## 240     No   Male <=120   Yes     asymptomatic                Normal
2  0
## 241     No   Male <=120   Yes  atypical angina              Abnormal
1  1
## 242     No female <=120   Yes     asymptomatic              Abnormal
1  0
## 243     No   Male <=120    No     asymptomatic                Normal
1  2
## 244     No   Male <=120   Yes     asymptomatic              Abnormal
1  1
## 245     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 246     No   Male <=120    No     asymptomatic                Normal
1  0
## 247     No female <=120   Yes     asymptomatic                Normal
1  2
## 248     No   Male <=120   Yes   typical angina              Abnormal
1  3
## 249     No   Male <=120    No   typical angina                Normal
2  1
## 250     No   Male <=120    No  atypical angina                Normal
1  3
## 251     No   Male <=120   Yes     asymptomatic              Abnormal
1  3
## 252     No   Male  >120   Yes     asymptomatic                Normal
1  4
## 253     No female  >120    No     asymptomatic              Abnormal
1  3
## 254     No   Male <=120   Yes     asymptomatic                Normal
1  2
## 255     No   Male <=120    No non-anginal pain                Normal
2  0
## 256     No   Male <=120   Yes     asymptomatic                Normal
1  3
## 257     No   Male <=120   Yes     asymptomatic                Normal
1  2
## 258     No   Male <=120   Yes     asymptomatic                Normal
1  0
## 259     No female <=120   Yes     asymptomatic              Abnormal
1  0
## 260     No   Male <=120   Yes non-anginal pain              Abnormal
```

```
1  0
## 261     No female  >120   Yes     asymptomatic              Abnormal
1  2
## 262     No   Male <=120    No     asymptomatic              Abnormal
2  1
## 263     No   Male <=120   Yes     asymptomatic              Abnormal
1  2
## 264     No female <=120   Yes     asymptomatic              Abnormal
1  2
## 265     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 266     No   Male <=120   Yes     asymptomatic                Normal
2  1
## 267     No female <=120   Yes     asymptomatic  Probable or Definite
1  0
## 268     No   Male <=120    No  atypical angina                Normal
2  3
## 269     No   Male <=120   Yes     asymptomatic                Normal
1  2
## 270     No   Male  >120   Yes     asymptomatic                Normal
0  0
## 271     No   Male <=120    No     asymptomatic                Normal
2  0
## 272     No   Male <=120    No non-anginal pain              Abnormal
1  2
## 273     No   Male <=120    No     asymptomatic              Abnormal
1  0
## 274     No   Male <=120    No     asymptomatic              Abnormal
2  1
## 275     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 276     No   Male <=120    No     asymptomatic              Abnormal
2  2
## 277     No   Male <=120    No     asymptomatic              Abnormal
1  1
## 278     No   Male <=120    No   typical angina              Abnormal
2  0
## 279     No female  >120    No   typical angina                Normal
2  2
## 280     No   Male <=120   Yes     asymptomatic                Normal
1  1
## 281     No   Male <=120   Yes     asymptomatic              Abnormal
1  0
## 282     No   Male  >120   Yes     asymptomatic              Abnormal
1  0
## 283     No   Male  >120    No  atypical angina              Abnormal
1  1
## 284     No   Male <=120    No     asymptomatic              Abnormal
2  0
## 285     No   Male <=120   Yes     asymptomatic                Normal
2  1
```

```
## 286     No   Male <=120   Yes     asymptomatic              Abnormal
1  2
## 287     No   Male <=120    No non-anginal pain              Abnormal
2  2
## 288     No   Male <=120    No   typical angina                Normal
2  1
## 289     No   Male <=120   Yes     asymptomatic              Abnormal
1  1
## 290     No female <=120   Yes     asymptomatic  Probable or Definite
1  1
## 291     No   Male <=120    No     asymptomatic              Abnormal
2  1
## 292     No   Male <=120    No     asymptomatic  Probable or Definite
0  3
## 293     No female  >120   Yes     asymptomatic                Normal
1  2
## 294     No   Male <=120    No  atypical angina                Normal
1  0
## 295     No   Male <=120   Yes     asymptomatic              Abnormal
0  0
## 296     No   Male <=120   Yes     asymptomatic                Normal
2  2
## 297     No female <=120   Yes     asymptomatic              Abnormal
1  0
## 298     No   Male  >120    No     asymptomatic                Normal
1  2
## 299     No female <=120   Yes     asymptomatic              Abnormal
1  0
## 300     No   Male <=120    No non-anginal pain              Abnormal
1  0
## 301     No   Male  >120    No     asymptomatic              Abnormal
1  2
## 302     No   Male <=120   Yes     asymptomatic              Abnormal
1  1
## 303     No female <=120    No   typical angina                Normal
1  1
##     thal age trestbps chol thalach oldpeak first_clust second_clust
## 1      1  63      145  233     150     2.3           1            5
## 2      2  37      130  250     187     3.5           5            5
## 3      2  41      130  204     172     1.4           5            4
## 4      2  56      120  236     178     0.8           5            4
## 5      2  57      120  354     163     0.6           5            1
## 6      1  57      140  192     148     0.4           5            4
## 7      2  56      140  294     153     1.3           1            1
## 8      3  44      120  263     173     0.0           2            4
## 9      3  52      172  199     162     0.5           1            5
## 10     2  57      150  168     174     1.6           5            4
## 11     2  54      140  239     160     1.2           5            4
## 12     2  48      130  275     139     0.2           5            1
## 13     2  49      130  266     171     0.6           5            4
## 14     2  64      110  211     144     1.8           1            5
```

```
## 15     2  58      150  283     162     1.0           1             1
## 16     2  50      120  219     158     1.6           5             1
## 17     2  58      120  340     172     0.0           5             1
## 18     2  66      150  226     114     2.6           1             5
## 19     2  43      150  247     171     1.5           5             4
## 20     2  69      140  239     151     1.8           1             1
## 21     3  59      135  234     161     0.5           2             4
## 22     2  44      130  233     179     0.4           5             4
## 23     2  42      140  226     178     0.0           5             4
## 24     2  61      150  243     137     1.0           1             5
## 25     3  40      140  199     178     1.4           2             4
## 26     2  71      160  302     162     0.4           1             1
## 27     2  59      150  212     157     1.6           1             5
## 28     2  51      110  175     123     0.6           5             4
## 29     2  65      140  417     157     0.8           1             1
## 30     2  53      130  197     152     1.2           1             5
## 31     2  41      105  198     168     0.0           5             4
## 32     3  65      120  177     140     0.4           2             4
## 33     2  44      130  219     188     0.0           5             4
## 34     2  54      125  273     152     0.5           1             5
## 35     2  51      125  213     125     1.4           1             5
## 36     2  46      142  177     160     1.4           5             5
## 37     2  54      135  304     170     0.0           1             1
## 38     3  54      150  232     165     1.6           2             5
## 39     2  65      155  269     148     0.8           1             1
## 40     2  65      160  360     151     0.8           1             1
## 41     2  51      140  308     142     1.5           1             1
## 42     2  48      130  245     180     0.2           5             4
## 43     2  45      104  208     148     3.0           4             3
## 44     2  53      130  264     143     0.4           5             1
## 45     2  39      140  321     182     0.0           5             4
## 46     2  52      120  325     172     0.2           5             4
## 47     2  44      140  235     180     0.0           5             4
## 48     2  47      138  257     156     0.0           5             4
## 49     0  53      128  216     115     0.0           5             1
## 50     2  53      138  234     160     0.0           5             1
## 51     2  51      130  256     149     0.5           5             1
## 52     2  66      120  302     151     0.4           1             1
## 53     3  62      130  231     146     1.8           2             2
## 54     2  44      108  141     175     0.6           5             4
## 55     2  63      135  252     172     0.0           1             1
## 56     2  52      134  201     158     0.8           5             4
## 57     2  48      122  222     186     0.0           5             4
## 58     2  45      115  260     185     0.0           5             4
## 59     2  34      118  182     174     0.0           5             4
## 60     2  57      128  303     159     0.0           5             1
## 61     2  71      110  265     130     0.0           1             1
## 62     3  54      108  309     156     0.0           2             4
## 63     1  52      118  186     190     0.0           5             4
## 64     1  41      135  203     132     0.0           5             4
## 65     2  58      140  211     165     0.0           1             5
```

```
## 66     2  35      138  183      182     1.4            5              4
## 67     2  51      100  222      143     1.2            5              4
## 68     2  45      130  234      175     0.6            5              1
## 69     2  44      120  220      170     0.0            5              4
## 70     2  62      124  209      163     0.0            5              1
## 71     3  54      120  258      147     0.4            2              5
## 72     3  51       94  227      154     0.0            2              4
## 73     2  29      130  204      202     0.0            5              4
## 74     2  51      140  261      186     0.0            5              4
## 75     2  43      122  213      165     0.2            5              4
## 76     2  55      135  250      161     1.4            5              1
## 77     2  51      125  245      166     2.4            1              5
## 78     2  59      140  221      164     0.0            5              4
## 79     2  52      128  205      184     0.0            5              4
## 80     3  58      105  240      154     0.6            2              5
## 81     2  41      112  250      179     0.0            5              4
## 82     2  45      128  308      170     0.0            5              4
## 83     2  60      102  318      160     0.0            5              1
## 84     3  52      152  298      178     1.2            1              5
## 85     2  42      102  265      122     0.6            5              1
## 86     3  67      115  564      160     1.6            1              1
## 87     3  68      118  277      151     1.0            2              2
## 88     3  46      101  197      156     0.0            2              4
## 89     2  54      110  214      158     1.6            5              1
## 90     2  58      100  248      122     1.0            5              1
## 91     2  48      124  255      175     0.0            2              2
## 92     3  57      132  207      168     0.0            2              4
## 93     2  52      138  223      169     0.0            2              2
## 94     2  54      132  288      159     0.0            1              1
## 95     2  45      112  160      138     0.0            5              4
## 96     3  53      142  226      111     0.0            4              3
## 97     2  62      140  394      157     1.2            1              1
## 98     3  52      108  233      147     0.1            2              2
## 99     2  43      130  315      162     1.9            5              4
## 100    2  53      130  246      173     0.0            1              2
## 101    2  42      148  244      178     0.8            5              4
## 102    3  59      178  270      145     4.2            3              5
## 103    2  63      140  195      179     0.0            5              1
## 104    3  42      120  240      194     0.8            2              5
## 105    2  50      129  196      163     0.0            5              4
## 106    2  68      120  211      115     1.5            1              1
## 107    2  69      160  234      131     0.1            1              5
## 108    2  45      138  236      152     0.2            5              1
## 109    2  50      120  244      162     1.1            5              1
## 110    2  50      110  254      159     0.0            5              1
## 111    2  64      180  325      154     0.0            1              1
## 112    3  57      150  126      173     0.2            2              2
## 113    3  64      140  313      133     0.2            1              1
## 114    3  43      110  211      161     0.0            2              4
## 115    2  55      130  262      155     0.0            5              4
## 116    2  37      120  215      170     0.0            5              4
```

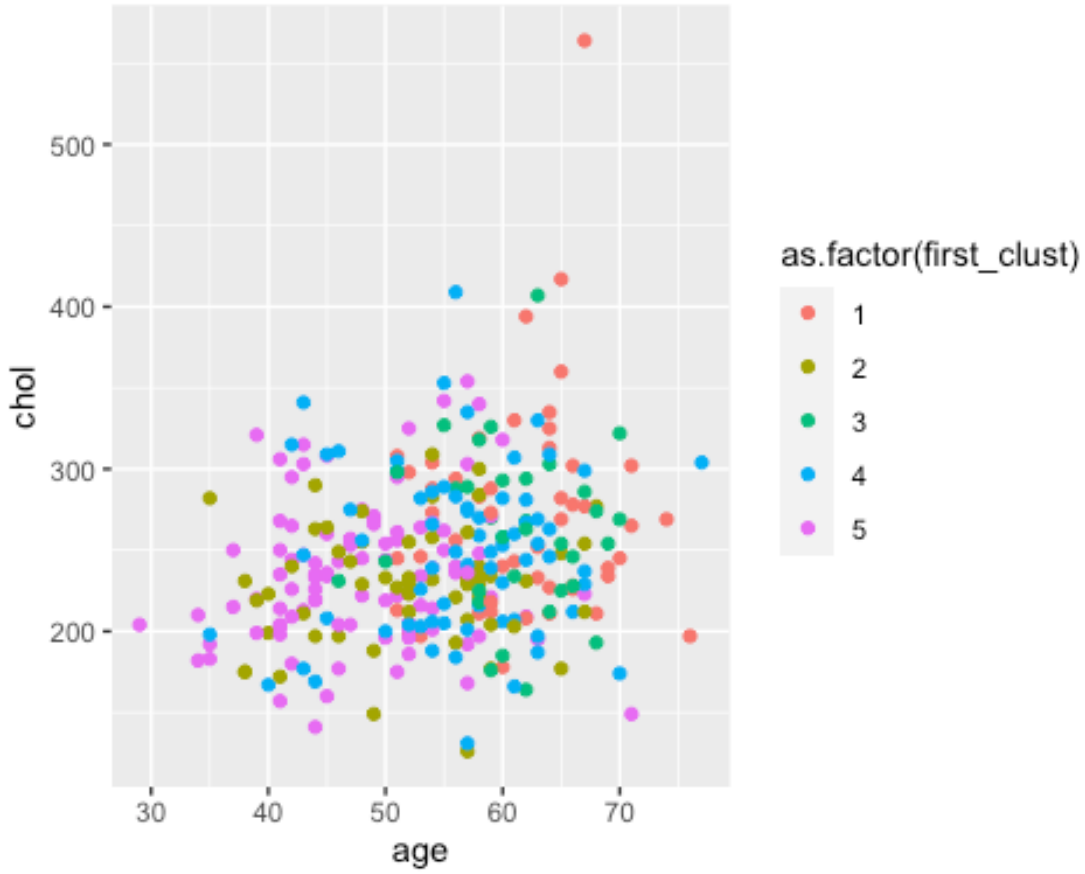
```
## 117    2  41      130  214     168     2.0           5             5
## 118    3  56      120  193     162     1.9           2             5
## 119    2  46      105  204     172     0.0           5             4
## 120    2  46      138  243     152     0.0           5             1
## 121    2  64      130  303     122     2.0           3             1
## 122    2  59      138  271     182     0.0           5             4
## 123    2  41      112  268     172     0.0           5             4
## 124    2  54      108  267     167     0.0           5             1
## 125    2  39       94  199     179     0.0           5             4
## 126    2  34      118  210     192     0.7           5             4
## 127    2  47      112  204     143     0.1           5             4
## 128    2  67      152  277     172     0.0           1             1
## 129    2  52      136  196     169     0.1           5             1
## 130    2  74      120  269     121     0.2           1             1
## 131    2  54      160  201     163     0.0           5             1
## 132    2  49      134  271     162     0.0           5             1
## 133    2  42      120  295     162     0.0           5             4
## 134    2  41      110  235     153     0.0           5             4
## 135    2  41      126  306     163     0.0           5             1
## 136    2  49      130  269     163     0.0           5             1
## 137    2  60      120  178      96     0.0           1             1
## 138    2  62      128  208     140     0.0           1             5
## 139    1  57      110  201     126     1.5           4             3
## 140    3  64      128  263     105     0.2           4             3
## 141    2  51      120  295     157     0.6           5             1
## 142    2  43      115  303     181     1.2           5             4
## 143    2  42      120  209     173     0.0           5             4
## 144    2  67      106  223     142     0.3           5             1
## 145    2  76      140  197     116     1.1           1             1
## 146    2  70      156  245     143     0.0           1             1
## 147    2  44      118  242     149     0.3           5             1
## 148    2  60      150  240     171     0.9           1             1
## 149    2  44      120  226     169     0.0           5             4
## 150    2  42      130  180     150     0.0           5             4
## 151    1  66      160  228     138     2.3           1             1
## 152    2  71      112  149     125     1.6           5             1
## 153    3  64      170  227     155     0.6           1             5
## 154    2  66      146  278     152     0.0           1             1
## 155    2  39      138  220     152     0.0           5             4
## 156    2  58      130  197     131     0.6           5             1
## 157    2  47      130  253     179     0.0           5             4
## 158    2  35      122  192     174     0.0           5             4
## 159    3  58      125  220     144     0.4           2             2
## 160    3  56      130  221     163     0.0           2             4
## 161    2  56      120  240     169     0.0           5             4
## 162    2  55      132  342     166     1.2           5             1
## 163    2  41      120  157     182     0.0           5             4
## 164    2  38      138  175     173     0.0           2             2
## 165    2  38      138  175     173     0.0           2             2
## 166    2  67      160  286     108     1.5           3             3
## 167    3  67      120  229     129     2.6           4             3
```

```
## 168    2  62      140  268     160     3.6           3             3
## 169    3  63      130  254     147     1.4           4             3
## 170    3  53      140  203     155     3.1           4             3
## 171    1  56      130  256     142     0.6           1             5
## 172    3  48      110  229     168     1.0           2             5
## 173    2  58      120  284     160     1.8           2             5
## 174    3  58      132  224     173     3.2           2             2
## 175    3  60      130  206     132     2.4           4             3
## 176    3  40      110  167     114     2.0           4             3
## 177    3  60      117  230     160     1.4           4             2
## 178    2  64      140  335     158     0.0           1             1
## 179    3  43      120  177     120     2.5           4             3
## 180    1  57      150  276     112     0.6           4             3
## 181    3  55      132  353     132     1.2           4             3
## 182    3  65      150  225     114     1.0           3             3
## 183    2  61      130  330     169     0.0           1             1
## 184    3  58      112  230     165     2.5           2             5
## 185    3  50      150  243     128     2.6           3             3
## 186    2  44      112  290     153     0.0           2             4
## 187    3  60      130  253     144     1.4           4             3
## 188    3  54      124  266     109     2.2           4             3
## 189    3  50      140  233     163     0.6           2             2
## 190    3  41      110  172     158     0.0           2             4
## 191    3  51      130  305     142     1.2           4             3
## 192    3  58      128  216     131     2.2           4             3
## 193    3  54      120  188     113     1.4           4             3
## 194    3  60      145  282     142     2.8           4             3
## 195    2  60      140  185     155     3.0           3             5
## 196    3  59      170  326     140     3.4           3             3
## 197    2  46      150  231     147     3.6           3             5
## 198    3  67      125  254     163     0.2           2             2
## 199    3  62      120  267      99     1.8           4             3
## 200    1  65      110  248     158     0.6           2             2
## 201    2  44      110  197     177     0.0           2             4
## 202    3  60      125  258     141     2.8           4             3
## 203    3  58      150  270     111     0.8           4             3
## 204    3  68      180  274     150     1.6           3             5
## 205    3  62      160  164     145     6.2           3             3
## 206    3  52      128  255     161     0.0           2             2
## 207    3  59      110  239     142     1.2           4             3
## 208    3  60      150  258     157     2.6           3             3
## 209    3  49      120  188     139     2.0           2             2
## 210    3  59      140  177     162     0.0           2             2
## 211    3  57      128  229     150     0.4           2             2
## 212    3  61      120  260     140     3.6           4             3
## 213    3  39      118  219     140     1.2           2             4
## 214    3  61      145  307     146     1.0           4             3
## 215    2  56      125  249     144     1.2           4             3
## 216    3  43      132  341     136     3.0           4             3
## 217    3  62      130  263      97     1.2           3             3
## 218    3  63      130  330     132     1.8           4             3
```

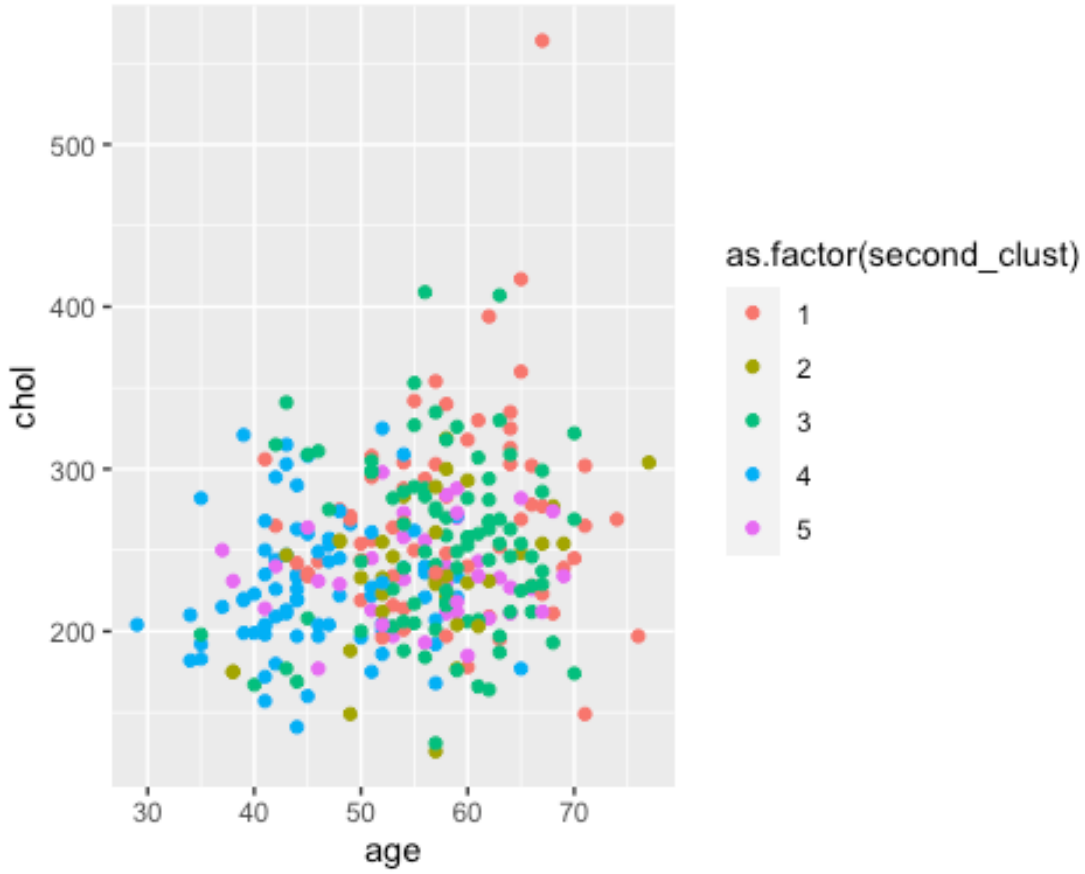
```
## 219    3  65      135  254     127     2.8           3             3
## 220    3  48      130  256     150     0.0           4             2
## 221    3  63      150  407     154     4.0           3             3
## 222    3  55      140  217     111     5.6           4             3
## 223    2  65      138  282     174     1.4           1             5
## 224    3  56      200  288     133     4.0           3             3
## 225    3  54      110  239     126     2.8           4             3
## 226    3  70      145  174     125     2.6           4             3
## 227    3  62      120  281     103     1.4           4             3
## 228    3  35      120  198     130     1.6           4             3
## 229    3  59      170  288     159     0.2           1             5
## 230    3  64      125  309     131     1.8           4             3
## 231    2  47      108  243     152     0.0           2             4
## 232    3  57      165  289     124     1.0           3             2
## 233    3  55      160  289     145     0.8           4             3
## 234    2  64      120  246      96     2.2           4             3
## 235    2  70      130  322     109     2.4           3             3
## 236    3  51      140  299     173     1.6           4             3
## 237    3  58      125  300     171     0.0           2             2
## 238    3  60      140  293     170     1.2           3             2
## 239    2  77      125  304     162     0.0           4             2
## 240    3  35      126  282     156     0.0           2             4
## 241    3  70      160  269     112     2.9           3             3
## 242    2  59      174  249     143     0.0           4             3
## 243    1  64      145  212     132     2.0           3             3
## 244    3  57      152  274      88     1.2           4             3
## 245    1  56      132  184     105     2.1           4             3
## 246    3  48      124  274     166     0.5           2             4
## 247    3  56      134  409     150     1.9           4             3
## 248    1  66      160  246     120     0.0           3             3
## 249    3  54      192  283     195     0.0           2             2
## 250    3  69      140  254     146     2.0           3             2
## 251    3  51      140  298     122     4.2           3             3
## 252    3  43      132  247     143     0.1           4             2
## 253    2  62      138  294     106     1.9           3             3
## 254    2  67      100  299     125     0.9           4             3
## 255    2  59      160  273     125     0.0           1             5
## 256    3  45      142  309     147     0.0           4             3
## 257    3  58      128  259     130     3.0           4             3
## 258    3  50      144  200     126     0.9           4             3
## 259    2  62      150  244     154     1.4           4             3
## 260    3  38      120  231     182     3.8           2             5
## 261    3  66      178  228     165     1.0           3             3
## 262    2  52      112  230     160     0.0           2             4
## 263    3  53      123  282      95     2.0           4             3
## 264    2  63      108  269     169     1.8           4             3
## 265    2  54      110  206     108     0.0           4             3
## 266    2  66      112  212     132     0.1           4             3
## 267    2  55      180  327     117     3.4           3             3
## 268    2  49      118  149     126     0.8           2             2
## 269    2  54      122  286     116     3.2           4             3
```

```
## 270    3  56      130  283     103     1.6           4             3
## 271    3  46      120  249     144     0.8           2             4
## 272    2  61      134  234     145     2.6           3             5
## 273    2  67      120  237      71     1.0           4             3
## 274    3  58      100  234     156     0.1           2             2
## 275    2  47      110  275     118     1.0           4             3
## 276    3  52      125  212     168     1.0           2             2
## 277    3  58      146  218     105     2.0           3             3
## 278    3  57      124  261     141     0.3           2             2
## 279    2  58      136  319     152     0.0           1             2
## 280    2  61      138  166     125     3.6           4             3
## 281    1  42      136  315     125     1.8           4             3
## 282    0  52      128  204     156     1.0           4             5
## 283    1  59      126  218     134     2.2           1             5
## 284    3  40      152  223     181     0.0           2             4
## 285    3  61      140  207     138     1.9           4             3
## 286    3  46      140  311     120     1.8           4             3
## 287    2  59      134  204     162     0.8           2             2
## 288    2  57      154  232     164     0.0           2             2
## 289    3  57      110  335     143     3.0           4             3
## 290    3  55      128  205     130     2.0           4             3
## 291    3  61      148  203     161     0.0           2             2
## 292    1  58      114  318     140     4.4           3             3
## 293    1  58      170  225     146     2.8           3             3
## 294    3  67      152  212     150     0.8           2             5
## 295    1  44      120  169     144     2.8           4             3
## 296    3  63      140  187     144     4.0           4             3
## 297    2  63      124  197     136     0.0           4             3
## 298    1  59      164  176      90     1.0           3             3
## 299    3  57      140  241     123     0.2           4             3
## 300    3  45      110  264     132     1.2           2             5
## 301    3  68      144  193     141     3.4           3             3
## 302    3  57      130  131     115     1.2           4             3
## 303    2  57      130  236     174     0.0           5             1
```

```
# Create and print the plot of age and chol for the first clustering algor
ithm
plot_one <- ggplot(heart_disease2, aes(x=age, y=chol, color=as.factor(firs
t_clust))) +
  geom_point()
plot_one
```



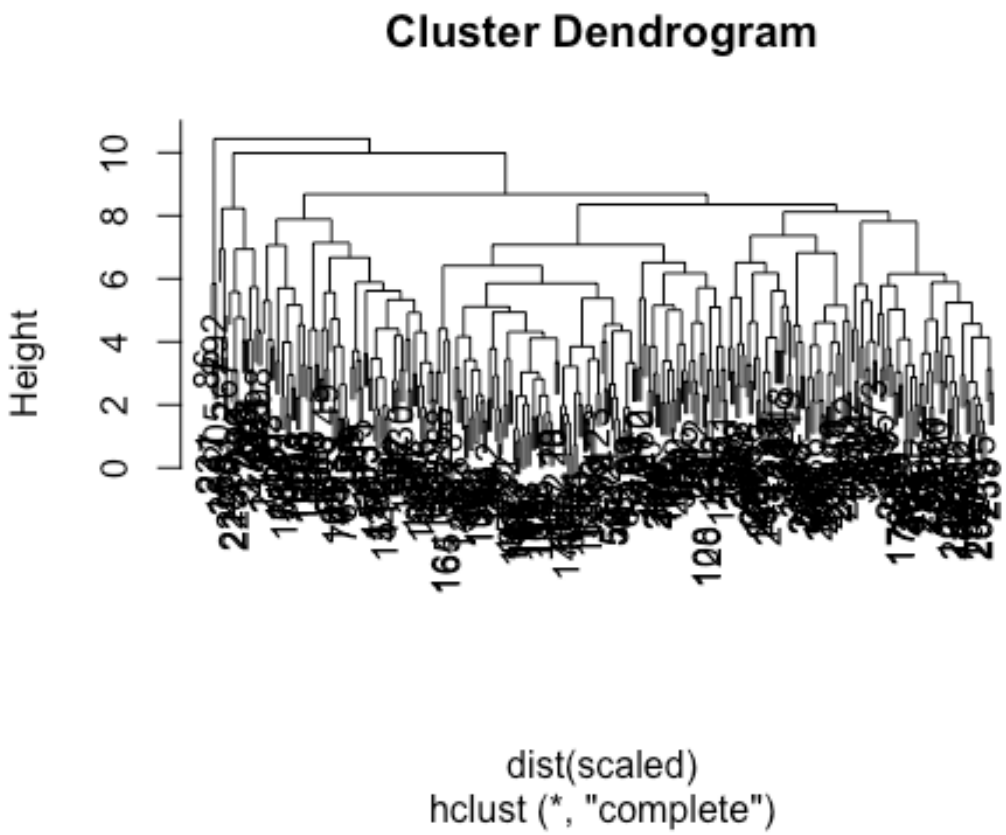

```
# Create and print the plot of age and chol for the second clustering algo
rithm
plot_two <- ggplot(heart_disease2, aes(x=age, y=chol, color=as.factor(seco
nd_clust))) +
  geom_point()
plot_two
```



##

12. Hierarchical clustering: another clustering approach An alternative to k-means clustering is hierarchical clustering. This method works well when data have a nested structure. Heart disease patient data might follow this type of structure. For example, if men are more likely to exhibit specific characteristics, those characteristics might be nested inside the gender variable. Hierarchical clustering also does not require the number of clusters to be selected before running the algorithm. Clusters can be selected by using the dendrogram. The dendrogram allows us to see how similar observations are to one another, and they are useful in helping us choose the number of clusters to group the data. It is now time for us to see how hierarchical clustering groups the data.

```
# Execute hierarchical clustering with complete linkage
hier_clust_1 <- hclust(dist(scaled), method = "complete")

# Print the dendrogram
plot(hier_clust_1)
```

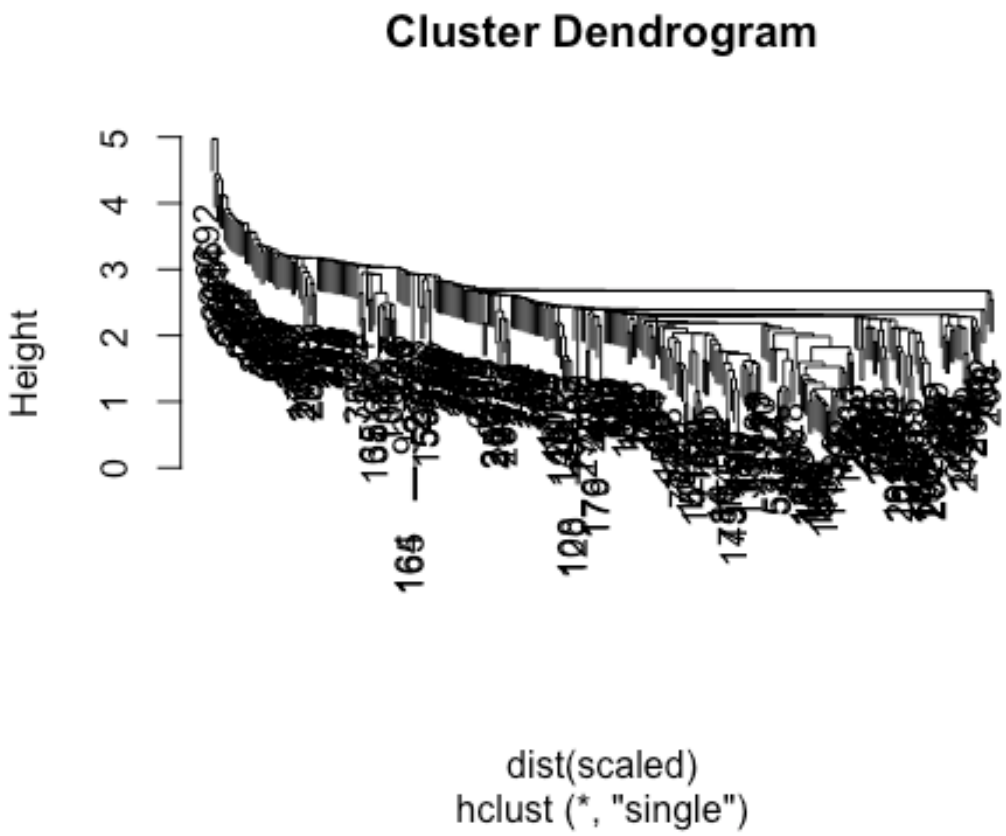



```
# Get cluster assignments based on number of selected clusters
hc_1_assign <- cutree(hier_clust_1, 5)
```

13. Hierarchical clustering round two

In hierarchical clustering, there are multiple ways to measure the dissimilarity between clusters of observations. Complete linkage records the largest dissimilarity between any two points in the two clusters being compared. On the other hand, single linkage is the smallest dissimilarity between any two points in the clusters. Different linkages will result in different clusters being formed. We want to explore different algorithms to group our heart disease patients. The best way to measure dissimilarity between patients could be to look at the smallest difference between patients and minimize that difference when grouping together clusters. It is always a good idea to explore different dissimilarity measures. Let's implement hierarchical clustering using a new linkage function.

```
# Execute hierarchical clustering with single linkage
hier_clust_2 <- hclust(dist(scaled), method = "single")

# Print the dendrogram
plot(hier_clust_2)
```




```
# Get cluster assignments based on number of selected clusters
hc_2_assign <- cutree(hier_clust_2, 5)
```

14. Comparing clustering results

The doctors are interested in grouping similar patients together to determine appropriate treatments. Therefore, they want clusters with more than a few patients to see different treatment options. While a patient can be in a cluster by themselves, this means that the treatment they received might not be recommended for someone else in the group. Like the k-means algorithm, the way to evaluate hierarchical clusters is to investigate which patients are grouped together. Are there patterns evident in the cluster assignments, or do they seem to be groups of noise? We're going to examine the clusters resulting from the two hierarchical algorithms.

```
heart_disease["hc_clust"] <- hc_1_assign
heart_disease
```

```
##     age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca
thal
## 1    63   1  3      145  233   1       0     150     0     2.3     0  0
1
## 2    37   1  2      130  250   0       1     187     0     3.5     0  0
2
## 3    41   0  1      130  204   0       0     172     0     1.4     2  0
```

```
2
## 4    56   1  1      120  236   0       1     178     0     0.8     2  0
2
## 5    57   0  0      120  354   0       1     163     1     0.6     2  0
2
## 6    57   1  0      140  192   0       1     148     0     0.4     1  0
1
## 7    56   0  1      140  294   0       0     153     0     1.3     1  0
2
## 8    44   1  1      120  263   0       1     173     0     0.0     2  0
3
## 9    52   1  2      172  199   1       1     162     0     0.5     2  0
3
## 10   57   1  2      150  168   0       1     174     0     1.6     2  0
2
## 11   54   1  0      140  239   0       1     160     0     1.2     2  0
2
## 12   48   0  2      130  275   0       1     139     0     0.2     2  0
2
## 13   49   1  1      130  266   0       1     171     0     0.6     2  0
2
## 14   64   1  3      110  211   0       0     144     1     1.8     1  0
2
## 15   58   0  3      150  283   1       0     162     0     1.0     2  0
2
## 16   50   0  2      120  219   0       1     158     0     1.6     1  0
2
## 17   58   0  2      120  340   0       1     172     0     0.0     2  0
2
## 18   66   0  3      150  226   0       1     114     0     2.6     0  0
2
## 19   43   1  0      150  247   0       1     171     0     1.5     2  0
2
## 20   69   0  3      140  239   0       1     151     0     1.8     2  2
2
## 21   59   1  0      135  234   0       1     161     0     0.5     1  0
3
## 22   44   1  2      130  233   0       1     179     1     0.4     2  0
2
## 23   42   1  0      140  226   0       1     178     0     0.0     2  0
2
## 24   61   1  2      150  243   1       1     137     1     1.0     1  0
2
## 25   40   1  3      140  199   0       1     178     1     1.4     2  0
3
## 26   71   0  1      160  302   0       1     162     0     0.4     2  2
2
## 27   59   1  2      150  212   1       1     157     0     1.6     2  0
2
## 28   51   1  2      110  175   0       1     123     0     0.6     2  0
2
```

```
## 29   65   0  2      140  417   1       0     157     0     0.8     2  1
2
## 30   53   1  2      130  197   1       0     152     0     1.2     0  0
2
## 31   41   0  1      105  198   0       1     168     0     0.0     2  1
2
## 32   65   1  0      120  177   0       1     140     0     0.4     2  0
3
## 33   44   1  1      130  219   0       0     188     0     0.0     2  0
2
## 34   54   1  2      125  273   0       0     152     0     0.5     0  1
2
## 35   51   1  3      125  213   0       0     125     1     1.4     2  1
2
## 36   46   0  2      142  177   0       0     160     1     1.4     0  0
2
## 37   54   0  2      135  304   1       1     170     0     0.0     2  0
2
## 38   54   1  2      150  232   0       0     165     0     1.6     2  0
3
## 39   65   0  2      155  269   0       1     148     0     0.8     2  0
2
## 40   65   0  2      160  360   0       0     151     0     0.8     2  0
2
## 41   51   0  2      140  308   0       0     142     0     1.5     2  1
2
## 42   48   1  1      130  245   0       0     180     0     0.2     1  0
2
## 43   45   1  0      104  208   0       0     148     1     3.0     1  0
2
## 44   53   0  0      130  264   0       0     143     0     0.4     1  0
2
## 45   39   1  2      140  321   0       0     182     0     0.0     2  0
2
## 46   52   1  1      120  325   0       1     172     0     0.2     2  0
2
## 47   44   1  2      140  235   0       0     180     0     0.0     2  0
2
## 48   47   1  2      138  257   0       0     156     0     0.0     2  0
2
## 49   53   0  2      128  216   0       0     115     0     0.0     2  0
0
## 50   53   0  0      138  234   0       0     160     0     0.0     2  0
2
## 51   51   0  2      130  256   0       0     149     0     0.5     2  0
2
## 52   66   1  0      120  302   0       0     151     0     0.4     1  0
2
## 53   62   1  2      130  231   0       1     146     0     1.8     1  3
3
## 54   44   0  2      108  141   0       1     175     0     0.6     1  0
```

```
2
## 55   63   0  2      135  252   0       0     172     0     0.0     2  0
2
## 56   52   1  1      134  201   0       1     158     0     0.8     2  1
2
## 57   48   1  0      122  222   0       0     186     0     0.0     2  0
2
## 58   45   1  0      115  260   0       0     185     0     0.0     2  0
2
## 59   34   1  3      118  182   0       0     174     0     0.0     2  0
2
## 60   57   0  0      128  303   0       0     159     0     0.0     2  1
2
## 61   71   0  2      110  265   1       0     130     0     0.0     2  1
2
## 62   54   1  1      108  309   0       1     156     0     0.0     2  0
3
## 63   52   1  3      118  186   0       0     190     0     0.0     1  0
1
## 64   41   1  1      135  203   0       1     132     0     0.0     1  0
1
## 65   58   1  2      140  211   1       0     165     0     0.0     2  0
2
## 66   35   0  0      138  183   0       1     182     0     1.4     2  0
2
## 67   51   1  2      100  222   0       1     143     1     1.2     1  0
2
## 68   45   0  1      130  234   0       0     175     0     0.6     1  0
2
## 69   44   1  1      120  220   0       1     170     0     0.0     2  0
2
## 70   62   0  0      124  209   0       1     163     0     0.0     2  0
2
## 71   54   1  2      120  258   0       0     147     0     0.4     1  0
3
## 72   51   1  2       94  227   0       1     154     1     0.0     2  1
3
## 73   29   1  1      130  204   0       0     202     0     0.0     2  0
2
## 74   51   1  0      140  261   0       0     186     1     0.0     2  0
2
## 75   43   0  2      122  213   0       1     165     0     0.2     1  0
2
## 76   55   0  1      135  250   0       0     161     0     1.4     1  0
2
## 77   51   1  2      125  245   1       0     166     0     2.4     1  0
2
## 78   59   1  1      140  221   0       1     164     1     0.0     2  0
2
## 79   52   1  1      128  205   1       1     184     0     0.0     2  0
2
```

```
## 80   58   1  2      105  240   0       0     154     1     0.6     1  0
3
## 81   41   1  2      112  250   0       1     179     0     0.0     2  0
2
## 82   45   1  1      128  308   0       0     170     0     0.0     2  0
2
## 83   60   0  2      102  318   0       1     160     0     0.0     2  1
2
## 84   52   1  3      152  298   1       1     178     0     1.2     1  0
3
## 85   42   0  0      102  265   0       0     122     0     0.6     1  0
2
## 86   67   0  2      115  564   0       0     160     0     1.6     1  0
3
## 87   68   1  2      118  277   0       1     151     0     1.0     2  1
3
## 88   46   1  1      101  197   1       1     156     0     0.0     2  0
3
## 89   54   0  2      110  214   0       1     158     0     1.6     1  0
2
## 90   58   0  0      100  248   0       0     122     0     1.0     1  0
2
## 91   48   1  2      124  255   1       1     175     0     0.0     2  2
2
## 92   57   1  0      132  207   0       1     168     1     0.0     2  0
3
## 93   52   1  2      138  223   0       1     169     0     0.0     2  4
2
## 94   54   0  1      132  288   1       0     159     1     0.0     2  1
2
## 95   45   0  1      112  160   0       1     138     0     0.0     1  0
2
## 96   53   1  0      142  226   0       0     111     1     0.0     2  0
3
## 97   62   0  0      140  394   0       0     157     0     1.2     1  0
2
## 98   52   1  0      108  233   1       1     147     0     0.1     2  3
3
## 99   43   1  2      130  315   0       1     162     0     1.9     2  1
2
## 100  53   1  2      130  246   1       0     173     0     0.0     2  3
2
## 101  42   1  3      148  244   0       0     178     0     0.8     2  2
2
## 102  59   1  3      178  270   0       0     145     0     4.2     0  0
3
## 103  63   0  1      140  195   0       1     179     0     0.0     2  2
2
## 104  42   1  2      120  240   1       1     194     0     0.8     0  0
3
## 105  50   1  2      129  196   0       1     163     0     0.0     2  0
```

```
2
## 106  68   0  2      120  211   0       0     115     0     1.5     1  0
2
## 107  69   1  3      160  234   1       0     131     0     0.1     1  1
2
## 108  45   0  0      138  236   0       0     152     1     0.2     1  0
2
## 109  50   0  1      120  244   0       1     162     0     1.1     2  0
2
## 110  50   0  0      110  254   0       0     159     0     0.0     2  0
2
## 111  64   0  0      180  325   0       1     154     1     0.0     2  0
2
## 112  57   1  2      150  126   1       1     173     0     0.2     2  1
3
## 113  64   0  2      140  313   0       1     133     0     0.2     2  0
3
## 114  43   1  0      110  211   0       1     161     0     0.0     2  0
3
## 115  55   1  1      130  262   0       1     155     0     0.0     2  0
2
## 116  37   0  2      120  215   0       1     170     0     0.0     2  0
2
## 117  41   1  2      130  214   0       0     168     0     2.0     1  0
2
## 118  56   1  3      120  193   0       0     162     0     1.9     1  0
3
## 119  46   0  1      105  204   0       1     172     0     0.0     2  0
2
## 120  46   0  0      138  243   0       0     152     1     0.0     1  0
2
## 121  64   0  0      130  303   0       1     122     0     2.0     1  2
2
## 122  59   1  0      138  271   0       0     182     0     0.0     2  0
2
## 123  41   0  2      112  268   0       0     172     1     0.0     2  0
2
## 124  54   0  2      108  267   0       0     167     0     0.0     2  0
2
## 125  39   0  2       94  199   0       1     179     0     0.0     2  0
2
## 126  34   0  1      118  210   0       1     192     0     0.7     2  0
2
## 127  47   1  0      112  204   0       1     143     0     0.1     2  0
2
## 128  67   0  2      152  277   0       1     172     0     0.0     2  1
2
## 129  52   0  2      136  196   0       0     169     0     0.1     1  0
2
## 130  74   0  1      120  269   0       0     121     1     0.2     2  1
2
```

```
## 131  54   0  2      160  201   0       1     163     0     0.0     2  1
2
## 132  49   0  1      134  271   0       1     162     0     0.0     1  0
2
## 133  42   1  1      120  295   0       1     162     0     0.0     2  0
2
## 134  41   1  1      110  235   0       1     153     0     0.0     2  0
2
## 135  41   0  1      126  306   0       1     163     0     0.0     2  0
2
## 136  49   0  0      130  269   0       1     163     0     0.0     2  0
2
## 137  60   0  2      120  178   1       1      96     0     0.0     2  0
2
## 138  62   1  1      128  208   1       0     140     0     0.0     2  0
2
## 139  57   1  0      110  201   0       1     126     1     1.5     1  0
1
## 140  64   1  0      128  263   0       1     105     1     0.2     1  1
3
## 141  51   0  2      120  295   0       0     157     0     0.6     2  0
2
## 142  43   1  0      115  303   0       1     181     0     1.2     1  0
2
## 143  42   0  2      120  209   0       1     173     0     0.0     1  0
2
## 144  67   0  0      106  223   0       1     142     0     0.3     2  2
2
## 145  76   0  2      140  197   0       2     116     0     1.1     1  0
2
## 146  70   1  1      156  245   0       0     143     0     0.0     2  0
2
## 147  44   0  2      118  242   0       1     149     0     0.3     1  1
2
## 148  60   0  3      150  240   0       1     171     0     0.9     2  0
2
## 149  44   1  2      120  226   0       1     169     0     0.0     2  0
2
## 150  42   1  2      130  180   0       1     150     0     0.0     2  0
2
## 151  66   1  0      160  228   0       0     138     0     2.3     2  0
1
## 152  71   0  0      112  149   0       1     125     0     1.6     1  0
2
## 153  64   1  3      170  227   0       0     155     0     0.6     1  0
3
## 154  66   0  2      146  278   0       0     152     0     0.0     1  1
2
## 155  39   0  2      138  220   0       1     152     0     0.0     1  0
2
## 156  58   0  0      130  197   0       1     131     0     0.6     1  0
```

```
2
## 157  47   1  2      130  253   0       1     179     0     0.0     2  0
2
## 158  35   1  1      122  192   0       1     174     0     0.0     2  0
2
## 159  58   1  1      125  220   0       1     144     0     0.4     1  4
3
## 160  56   1  1      130  221   0       0     163     0     0.0     2  0
3
## 161  56   1  1      120  240   0       1     169     0     0.0     0  0
2
## 162  55   0  1      132  342   0       1     166     0     1.2     2  0
2
## 163  41   1  1      120  157   0       1     182     0     0.0     2  0
2
## 164  38   1  2      138  175   0       1     173     0     0.0     2  4
2
## 165  38   1  2      138  175   0       1     173     0     0.0     2  4
2
## 166  67   1  0      160  286   0       0     108     1     1.5     1  3
2
## 167  67   1  0      120  229   0       0     129     1     2.6     1  2
3
## 168  62   0  0      140  268   0       0     160     0     3.6     0  2
2
## 169  63   1  0      130  254   0       0     147     0     1.4     1  1
3
## 170  53   1  0      140  203   1       0     155     1     3.1     0  0
3
## 171  56   1  2      130  256   1       0     142     1     0.6     1  1
1
## 172  48   1  1      110  229   0       1     168     0     1.0     0  0
3
## 173  58   1  1      120  284   0       0     160     0     1.8     1  0
2
## 174  58   1  2      132  224   0       0     173     0     3.2     2  2
3
## 175  60   1  0      130  206   0       0     132     1     2.4     1  2
3
## 176  40   1  0      110  167   0       0     114     1     2.0     1  0
3
## 177  60   1  0      117  230   1       1     160     1     1.4     2  2
3
## 178  64   1  2      140  335   0       1     158     0     0.0     2  0
2
## 179  43   1  0      120  177   0       0     120     1     2.5     1  0
3
## 180  57   1  0      150  276   0       0     112     1     0.6     1  1
1
## 181  55   1  0      132  353   0       1     132     1     1.2     1  1
3
```

```
## 182  65   0  0      150  225   0       0     114     0     1.0     1  3
3
## 183  61   0  0      130  330   0       0     169     0     0.0     2  0
2
## 184  58   1  2      112  230   0       0     165     0     2.5     1  1
3
## 185  50   1  0      150  243   0       0     128     0     2.6     1  0
3
## 186  44   1  0      112  290   0       0     153     0     0.0     2  1
2
## 187  60   1  0      130  253   0       1     144     1     1.4     2  1
3
## 188  54   1  0      124  266   0       0     109     1     2.2     1  1
3
## 189  50   1  2      140  233   0       1     163     0     0.6     1  1
3
## 190  41   1  0      110  172   0       0     158     0     0.0     2  0
3
## 191  51   0  0      130  305   0       1     142     1     1.2     1  0
3
## 192  58   1  0      128  216   0       0     131     1     2.2     1  3
3
## 193  54   1  0      120  188   0       1     113     0     1.4     1  1
3
## 194  60   1  0      145  282   0       0     142     1     2.8     1  2
3
## 195  60   1  2      140  185   0       0     155     0     3.0     1  0
2
## 196  59   1  0      170  326   0       0     140     1     3.4     0  0
3
## 197  46   1  2      150  231   0       1     147     0     3.6     1  0
2
## 198  67   1  0      125  254   1       1     163     0     0.2     1  2
3
## 199  62   1  0      120  267   0       1      99     1     1.8     1  2
3
## 200  65   1  0      110  248   0       0     158     0     0.6     2  2
1
## 201  44   1  0      110  197   0       0     177     0     0.0     2  1
2
## 202  60   1  0      125  258   0       0     141     1     2.8     1  1
3
## 203  58   1  0      150  270   0       0     111     1     0.8     2  0
3
## 204  68   1  2      180  274   1       0     150     1     1.6     1  0
3
## 205  62   0  0      160  164   0       0     145     0     6.2     0  3
3
## 206  52   1  0      128  255   0       1     161     1     0.0     2  1
3
## 207  59   1  0      110  239   0       0     142     1     1.2     1  1
```

```
3
## 208  60   0  0      150  258   0       0     157     0     2.6     1  2
3
## 209  49   1  2      120  188   0       1     139     0     2.0     1  3
3
## 210  59   1  0      140  177   0       1     162     1     0.0     2  1
3
## 211  57   1  2      128  229   0       0     150     0     0.4     1  1
3
## 212  61   1  0      120  260   0       1     140     1     3.6     1  1
3
## 213  39   1  0      118  219   0       1     140     0     1.2     1  0
3
## 214  61   0  0      145  307   0       0     146     1     1.0     1  0
3
## 215  56   1  0      125  249   1       0     144     1     1.2     1  1
2
## 216  43   0  0      132  341   1       0     136     1     3.0     1  0
3
## 217  62   0  2      130  263   0       1      97     0     1.2     1  1
3
## 218  63   1  0      130  330   1       0     132     1     1.8     2  3
3
## 219  65   1  0      135  254   0       0     127     0     2.8     1  1
3
## 220  48   1  0      130  256   1       0     150     1     0.0     2  2
3
## 221  63   0  0      150  407   0       0     154     0     4.0     1  3
3
## 222  55   1  0      140  217   0       1     111     1     5.6     0  0
3
## 223  65   1  3      138  282   1       0     174     0     1.4     1  1
2
## 224  56   0  0      200  288   1       0     133     1     4.0     0  2
3
## 225  54   1  0      110  239   0       1     126     1     2.8     1  1
3
## 226  70   1  0      145  174   0       1     125     1     2.6     0  0
3
## 227  62   1  1      120  281   0       0     103     0     1.4     1  1
3
## 228  35   1  0      120  198   0       1     130     1     1.6     1  0
3
## 229  59   1  3      170  288   0       0     159     0     0.2     1  0
3
## 230  64   1  2      125  309   0       1     131     1     1.8     1  0
3
## 231  47   1  2      108  243   0       1     152     0     0.0     2  0
2
## 232  57   1  0      165  289   1       0     124     0     1.0     1  3
3
```

```
## 233  55   1  0      160  289   0       0     145     1     0.8     1  1
3
## 234  64   1  0      120  246   0       0      96     1     2.2     0  1
2
## 235  70   1  0      130  322   0       0     109     0     2.4     1  3
2
## 236  51   1  0      140  299   0       1     173     1     1.6     2  0
3
## 237  58   1  0      125  300   0       0     171     0     0.0     2  2
3
## 238  60   1  0      140  293   0       0     170     0     1.2     1  2
3
## 239  77   1  0      125  304   0       0     162     1     0.0     2  3
2
## 240  35   1  0      126  282   0       0     156     1     0.0     2  0
3
## 241  70   1  2      160  269   0       1     112     1     2.9     1  1
3
## 242  59   0  0      174  249   0       1     143     1     0.0     1  0
2
## 243  64   1  0      145  212   0       0     132     0     2.0     1  2
1
## 244  57   1  0      152  274   0       1      88     1     1.2     1  1
3
## 245  56   1  0      132  184   0       0     105     1     2.1     1  1
1
## 246  48   1  0      124  274   0       0     166     0     0.5     1  0
3
## 247  56   0  0      134  409   0       0     150     1     1.9     1  2
3
## 248  66   1  1      160  246   0       1     120     1     0.0     1  3
1
## 249  54   1  1      192  283   0       0     195     0     0.0     2  1
3
## 250  69   1  2      140  254   0       0     146     0     2.0     1  3
3
## 251  51   1  0      140  298   0       1     122     1     4.2     1  3
3
## 252  43   1  0      132  247   1       0     143     1     0.1     1  4
3
## 253  62   0  0      138  294   1       1     106     0     1.9     1  3
2
## 254  67   1  0      100  299   0       0     125     1     0.9     1  2
2
## 255  59   1  3      160  273   0       0     125     0     0.0     2  0
2
## 256  45   1  0      142  309   0       0     147     1     0.0     1  3
3
## 257  58   1  0      128  259   0       0     130     1     3.0     1  2
3
## 258  50   1  0      144  200   0       0     126     1     0.9     1  0
```

```
3
## 259  62   0  0      150  244   0       1     154     1     1.4     1  0
2
## 260  38   1  3      120  231   0       1     182     1     3.8     1  0
3
## 261  66   0  0      178  228   1       1     165     1     1.0     1  2
3
## 262  52   1  0      112  230   0       1     160     0     0.0     2  1
2
## 263  53   1  0      123  282   0       1      95     1     2.0     1  2
3
## 264  63   0  0      108  269   0       1     169     1     1.8     1  2
2
## 265  54   1  0      110  206   0       0     108     1     0.0     1  1
2
## 266  66   1  0      112  212   0       0     132     1     0.1     2  1
2
## 267  55   0  0      180  327   0       2     117     1     3.4     1  0
2
## 268  49   1  2      118  149   0       0     126     0     0.8     2  3
2
## 269  54   1  0      122  286   0       0     116     1     3.2     1  2
2
## 270  56   1  0      130  283   1       0     103     1     1.6     0  0
3
## 271  46   1  0      120  249   0       0     144     0     0.8     2  0
3
## 272  61   1  3      134  234   0       1     145     0     2.6     1  2
2
## 273  67   1  0      120  237   0       1      71     0     1.0     1  0
2
## 274  58   1  0      100  234   0       1     156     0     0.1     2  1
3
## 275  47   1  0      110  275   0       0     118     1     1.0     1  1
2
## 276  52   1  0      125  212   0       1     168     0     1.0     2  2
3
## 277  58   1  0      146  218   0       1     105     0     2.0     1  1
3
## 278  57   1  1      124  261   0       1     141     0     0.3     2  0
3
## 279  58   0  1      136  319   1       0     152     0     0.0     2  2
2
## 280  61   1  0      138  166   0       0     125     1     3.6     1  1
2
## 281  42   1  0      136  315   0       1     125     1     1.8     1  0
1
## 282  52   1  0      128  204   1       1     156     1     1.0     1  0
0
## 283  59   1  2      126  218   1       1     134     0     2.2     1  1
1
```

```
## 284  40   1  0      152  223   0       1     181     0     0.0     2  0
3
## 285  61   1  0      140  207   0       0     138     1     1.9     2  1
3
## 286  46   1  0      140  311   0       1     120     1     1.8     1  2
3
## 287  59   1  3      134  204   0       1     162     0     0.8     2  2
2
## 288  57   1  1      154  232   0       0     164     0     0.0     2  1
2
## 289  57   1  0      110  335   0       1     143     1     3.0     1  1
3
## 290  55   0  0      128  205   0       2     130     1     2.0     1  1
3
## 291  61   1  0      148  203   0       1     161     0     0.0     2  1
3
## 292  58   1  0      114  318   0       2     140     0     4.4     0  3
1
## 293  58   0  0      170  225   1       0     146     1     2.8     1  2
1
## 294  67   1  2      152  212   0       0     150     0     0.8     1  0
3
## 295  44   1  0      120  169   0       1     144     1     2.8     0  0
1
## 296  63   1  0      140  187   0       0     144     1     4.0     2  2
3
## 297  63   0  0      124  197   0       1     136     1     0.0     1  0
2
## 298  59   1  0      164  176   1       0      90     0     1.0     1  2
1
## 299  57   0  0      140  241   0       1     123     1     0.2     1  0
3
## 300  45   1  3      110  264   0       1     132     0     1.2     1  0
3
## 301  68   1  0      144  193   1       1     141     0     3.4     1  2
3
## 302  57   1  0      130  131   0       1     115     1     1.2     1  1
3
## 303  57   0  1      130  236   0       0     174     0     0.0     1  1
2
##     target hc_clust
## 1        1        1
## 2        1        2
## 3        1        2
## 4        1        2
## 5        1        3
## 6        1        2
## 7        1        3
## 8        1        2
## 9        1        3
## 10       1        2
```

```
## 11       1        2
## 12       1        2
## 13       1        2
## 14       1        2
## 15       1        3
## 16       1        2
## 17       1        2
## 18       1        3
## 19       1        2
## 20       1        3
## 21       1        2
## 22       1        2
## 23       1        2
## 24       1        1
## 25       1        2
## 26       1        3
## 27       1        3
## 28       1        2
## 29       1        3
## 30       1        1
## 31       1        2
## 32       1        2
## 33       1        2
## 34       1        2
## 35       1        2
## 36       1        2
## 37       1        3
## 38       1        2
## 39       1        3
## 40       1        3
## 41       1        3
## 42       1        2
## 43       1        1
## 44       1        3
## 45       1        2
## 46       1        2
## 47       1        2
## 48       1        2
## 49       1        3
## 50       1        3
## 51       1        3
## 52       1        3
## 53       1        2
## 54       1        2
## 55       1        3
## 56       1        2
## 57       1        2
## 58       1        2
## 59       1        2
## 60       1        3
## 61       1        3
```

```
## 62       1        2
## 63       1        2
## 64       1        2
## 65       1        3
## 66       1        2
## 67       1        2
## 68       1        3
## 69       1        2
## 70       1        3
## 71       1        2
## 72       1        2
## 73       1        2
## 74       1        2
## 75       1        2
## 76       1        3
## 77       1        1
## 78       1        2
## 79       1        3
## 80       1        2
## 81       1        2
## 82       1        2
## 83       1        2
## 84       1        2
## 85       1        3
## 86       1        4
## 87       1        3
## 88       1        3
## 89       1        2
## 90       1        3
## 91       1        3
## 92       1        2
## 93       1        2
## 94       1        3
## 95       1        2
## 96       1        1
## 97       1        3
## 98       1        3
## 99       1        2
## 100      1        3
## 101      1        2
## 102      1        1
## 103      1        3
## 104      1        2
## 105      1        2
## 106      1        3
## 107      1        1
## 108      1        2
## 109      1        2
## 110      1        3
## 111      1        3
## 112      1        3
```

```
## 113      1        3
## 114      1        2
## 115      1        2
## 116      1        2
## 117      1        2
## 118      1        2
## 119      1        2
## 120      1        2
## 121      1        1
## 122      1        2
## 123      1        2
## 124      1        3
## 125      1        2
## 126      1        2
## 127      1        2
## 128      1        3
## 129      1        3
## 130      1        3
## 131      1        3
## 132      1        2
## 133      1        2
## 134      1        2
## 135      1        2
## 136      1        2
## 137      1        3
## 138      1        3
## 139      1        1
## 140      1        1
## 141      1        3
## 142      1        2
## 143      1        2
## 144      1        3
## 145      1        3
## 146      1        3
## 147      1        2
## 148      1        3
## 149      1        2
## 150      1        2
## 151      1        3
## 152      1        3
## 153      1        1
## 154      1        3
## 155      1        2
## 156      1        3
## 157      1        2
## 158      1        2
## 159      1        2
## 160      1        2
## 161      1        2
## 162      1        2
## 163      1        2
```

```
## 164      1        2
## 165      1        2
## 166      0        1
## 167      0        1
## 168      0        5
## 169      0        1
## 170      0        5
## 171      0        3
## 172      0        2
## 173      0        2
## 174      0        2
## 175      0        1
## 176      0        1
## 177      0        1
## 178      0        2
## 179      0        1
## 180      0        1
## 181      0        1
## 182      0        1
## 183      0        3
## 184      0        2
## 185      0        1
## 186      0        1
## 187      0        1
## 188      0        1
## 189      0        2
## 190      0        1
## 191      0        1
## 192      0        1
## 193      0        1
## 194      0        1
## 195      0        1
## 196      0        5
## 197      0        1
## 198      0        1
## 199      0        1
## 200      0        3
## 201      0        1
## 202      0        1
## 203      0        1
## 204      0        1
## 205      0        5
## 206      0        1
## 207      0        1
## 208      0        5
## 209      0        2
## 210      0        1
## 211      0        2
## 212      0        5
## 213      0        1
## 214      0        1
```

```
## 215      0        3
## 216      0        5
## 217      0        1
## 218      0        1
## 219      0        1
## 220      0        1
## 221      0        4
## 222      0        5
## 223      0        1
## 224      0        5
## 225      0        5
## 226      0        1
## 227      0        1
## 228      0        1
## 229      0        1
## 230      0        1
## 231      0        2
## 232      0        1
## 233      0        1
## 234      0        1
## 235      0        1
## 236      0        1
## 237      0        1
## 238      0        1
## 239      0        1
## 240      0        1
## 241      0        1
## 242      0        1
## 243      0        3
## 244      0        1
## 245      0        1
## 246      0        2
## 247      0        4
## 248      0        1
## 249      0        1
## 250      0        2
## 251      0        5
## 252      0        1
## 253      0        1
## 254      0        1
## 255      0        1
## 256      0        1
## 257      0        1
## 258      0        1
## 259      0        1
## 260      0        2
## 261      0        5
## 262      0        1
## 263      0        1
## 264      0        1
## 265      0        1
```

```
## 266      0        1
## 267      0        5
## 268      0        2
## 269      0        1
## 270      0        5
## 271      0        1
## 272      0        2
## 273      0        1
## 274      0        1
## 275      0        1
## 276      0        1
## 277      0        1
## 278      0        2
## 279      0        3
## 280      0        1
## 281      0        1
## 282      0        3
## 283      0        3
## 284      0        1
## 285      0        1
## 286      0        1
## 287      0        2
## 288      0        1
## 289      0        5
## 290      0        1
## 291      0        1
## 292      0        5
## 293      0        5
## 294      0        2
## 295      0        1
## 296      0        1
## 297      0        1
## 298      0        3
## 299      0        1
## 300      0        2
## 301      0        1
## 302      0        1
## 303      0        3
```

```
# Remove the sex, first_clust, and second_clust variables
hd_simple <- heart_disease[, !(names(heart_disease) %in% c("sex", "first_c
lust", "second_clust"))]

# Get the mean and standard deviation summary statistics
clust_summary <- do.call(data.frame, aggregate(. ~hc_clust, data = hd_simp
le, function(x) c(avg = mean(x), sd = sd(x))))
clust_summary
```

```
##   hc_clust  age.avg   age.sd    cp.avg     cp.sd trestbps.avg trestbps.
sd
## 1        1 56.89320 8.502494 0.4174757 0.9131316     134.5728    18.328
49
```

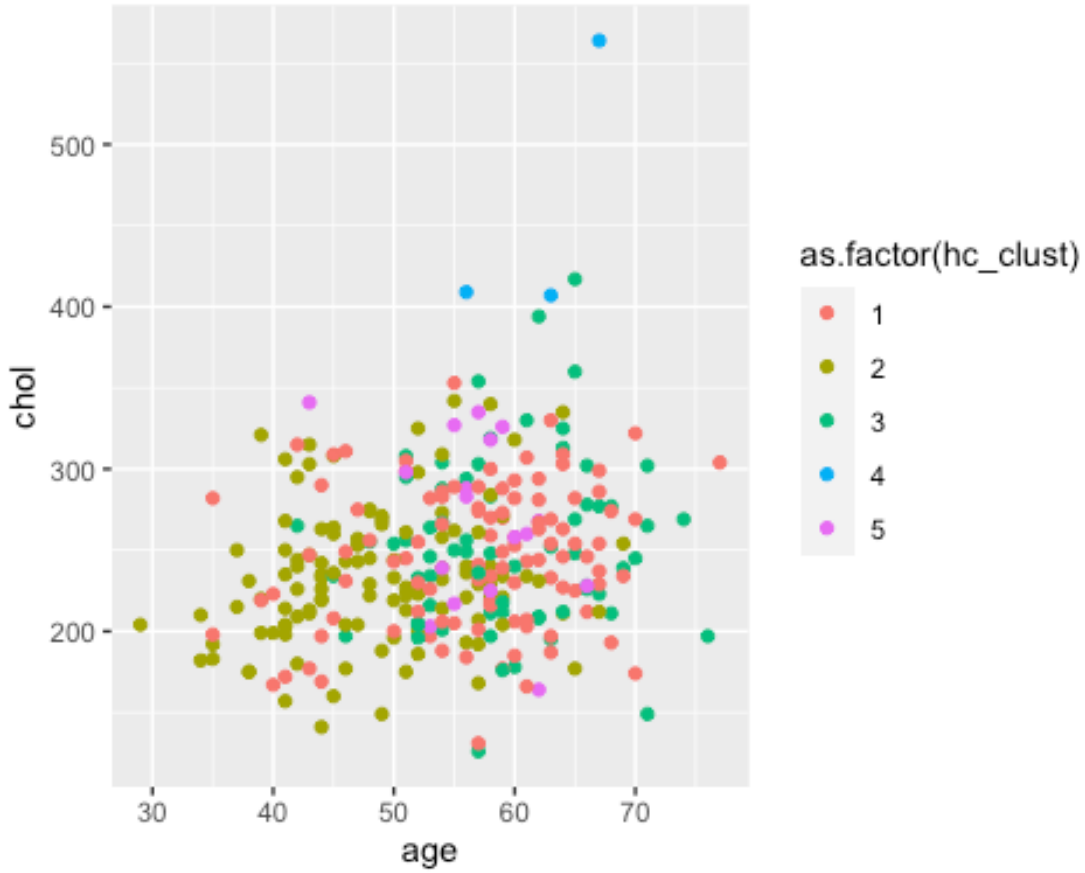
```
## 2        2 48.58407 8.011962 1.4778761 0.8973905     125.5487    12.591
09
## 3        3 59.26866 7.316611 1.2089552 0.9775869     133.5970    17.409
56
## 4        4 62.00000 5.567764 0.6666667 1.1547005     133.0000    17.521
42
## 5        5 56.82353 5.174883 0.0000000 0.0000000     146.1176    26.631
47
##   chol.avg  chol.sd    fbs.avg    fbs.sd restecg.avg restecg.sd thalach
.avg
## 1 246.4078 43.80201 0.14563107 0.3544608   0.4271845  0.5164346    135.
6505
## 2 233.5310 42.10745 0.01769912 0.1324428   0.6814159  0.4680027    163.
8938
## 3 252.1045 53.84690 0.32835821 0.4731602   0.4328358  0.5286885    149.
9701
## 4 460.0000 90.07219 0.00000000 0.0000000   0.0000000  0.0000000    154.
6667
## 5 269.2941 51.69353 0.35294118 0.4925922   0.5882353  0.7122871    137.
5882
##   thalach.sd exang.avg  exang.sd oldpeak.avg oldpeak.sd slope.avg  slop
e.sd
## 1  23.036079 0.6116505 0.4897580   1.3611650  1.0950184 1.2038835 0.530
6161
## 2  15.640699 0.1327434 0.3408085   0.6185841  0.8326537 1.5663717 0.595
7604
## 3  19.986336 0.1044776 0.3081877   0.5835821  0.6756808 1.6567164 0.509
0764
## 4   5.033223 0.3333333 0.5773503   2.5000000  1.3076697 1.0000000 0.000
0000
## 5  17.381955 0.7647059 0.4372373   3.4294118  1.2628166 0.5294118 0.514
4958
##      ca.avg     ca.sd thal.avg   thal.sd target.avg target.sd
## 1 1.0485437 1.0134260 2.572816 0.6199611  0.1165049 0.3223982
## 2 0.4159292 0.9794830 2.230088 0.4819760  0.8407080 0.3675783
## 3 0.5820896 0.8375515 1.940299 0.5471856  0.8507463 0.3590278
## 4 1.6666667 1.5275252 3.000000 0.0000000  0.3333333 0.5773503
## 5 1.2941176 1.1599949 2.647059 0.7018882  0.0000000 0.0000000
```

15. Visualizing the cluster contents

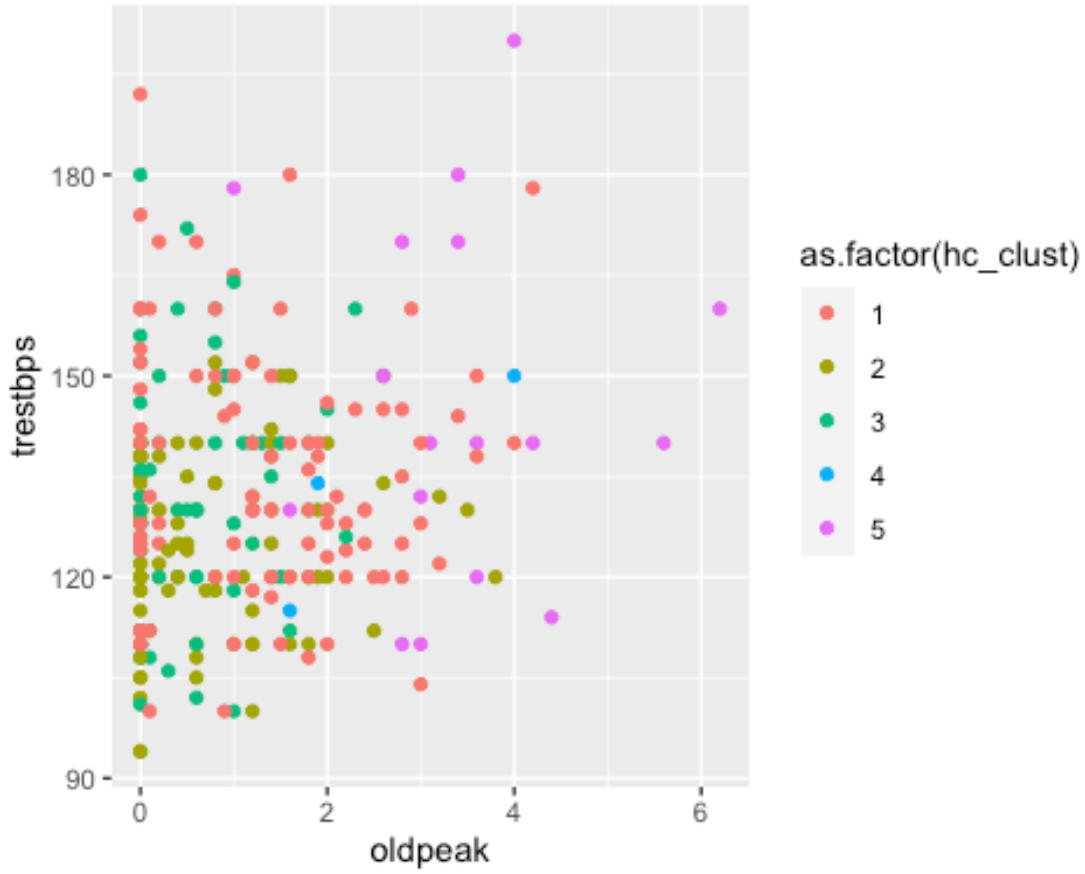
In addition to looking at the distributions of variables in each of the hierarchical clustering runs, we will make visualizations to evaluate the algorithms. Even though the data has more than two dimensions, we can get an idea of how the data clusters by looking at a scatter plot of two variables. We want to look for patterns that appear in the data and see what patients get clustered together.

```
# Plot age and chol
plot_one <- ggplot(heart_disease, aes(x = age, y = chol,
                                      color = as.factor(hc_clust))) +
```

```
  geom_point()
plot_one
```




```
# Plot oldpeak and trestbps
plot_two <- ggplot(heart_disease, aes(x = oldpeak, y = trestbps,
                                      color = as.factor(hc_clust))) +
  geom_point()
plot_two
```



##

16. Conclusion Now that we've tried multiple clustering algorithms, it is necessary to determine if we think any of them will work for clustering our patients. For the k-means algorithm, similar clusters must be produced for each iteration of the algorithm to make sure that the algorithm clusters the signal, not the noise. For the sake of the doctors, we also want to have multiple patients in each group so they can compare treatments. We only did some preliminary work to explore the performance of the algorithms, and it is necessary to explore further before making a recommendation. Remember that it is important the k-mean algorithm seems stable when running multiple iterations. This means that we would see similar groups of patients showing up in the plots from the different iterations of the algorithm. For the hierarchical clustering, we need a method that puts a balanced number of patients in each group.

```
#Add TRUE if the algorithm shows promise, add FALSE if it does not
explore_kmeans <- FALSE
explore_hierarch_complete <- TRUE
explore_hierarch_single <- FALSE
```

EXPLANATION OF CODE STEP BY STEP

In this section the code is explained line by line and why that method was applied and its description in detail.

6.1 IMPORTING BASIC LIBRARIES IN R

In this section some of the important libraries of R like tidyverse, dplyr, ggplot, stats, corrplot and others, are imported so as to use their functionalities later in code.

- **tidyverse** : It is one of the most important library , it's a collection of most important libraries , packages under its umbrella help in performing and interacting with the data .
- **dplyr**: Most useful package for data manipulation , one of the most amazing functions of this package is that we can use %>% pipe function to combine different functions , from filtering to grouping the data , in this project we will use the function mutate() , very interesting function . Major functions like select(), filter(), group_by(), summarise()

- **ggplot2** : One of the most amazing packages for visualization . We have done a lot of plotting using this function in our project. Be it scatter plot, boxplot, Histogram
- **corrplot** : used in graphical display of correlation matrix , how the attributes are correlated with each other

we can import our libraries by using function **library()**

```
# Loading the important libraries
library(tidyverse)
library(dplyr)
library(ggplot2)
library(stats)
library(ggfortify)
library(corrplot)
```

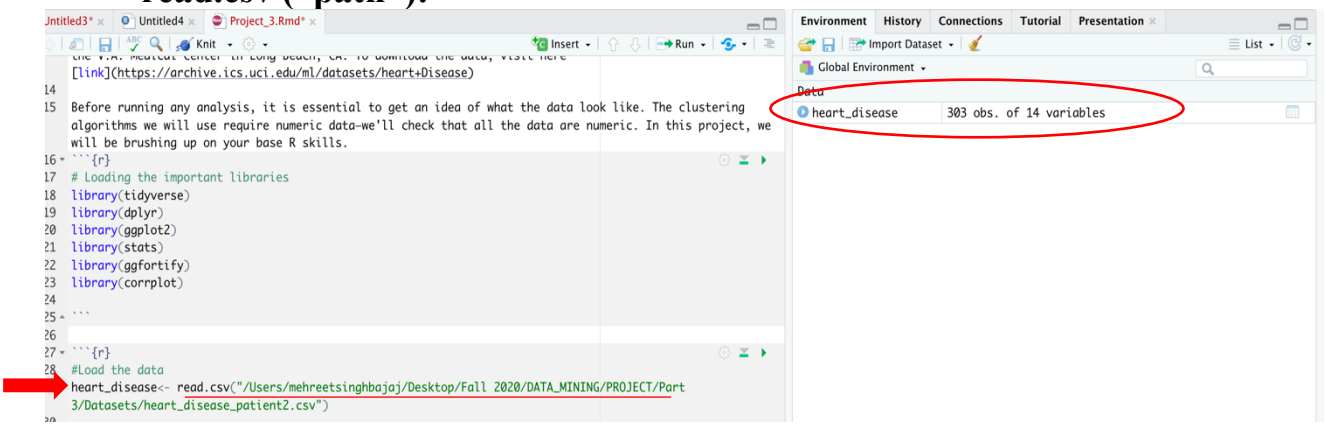
After this we will observe that the libraries are loaded up and we can check it in the packages terminal in R-Studio on right side there will be a ✓ tick mark Infront of the packages

☐	bayestestR	Understand and Describe Bayesian Models and Posterior Distributions	0.7.2
☐	broom	Convert Statistical Analysis Objects into Tidy Tibbles	0.5.6
☐	broom.mixed	Tidying Methods for Mixed Models	0.2.6
☐	broomExtra	Enhancements for 'broom' and 'easystats' Package Families	4.0.3
☐	caret	Classification and Regression Training	6.0-86
☐	coda	Output Analysis and Diagnostics for MCMC	0.19-3
☐	commonmark	High Performance CommonMark and Github Markdown Rendering in R	1.7
☑	corrplot	Visualization of a Correlation Matrix	0.84
☐	crosstalk	Inter-Widget Interactivity for HTML Widgets	1.1.0.1
☐	cubelyr	A Data Cube 'dplyr' Backend	1.0.0
☐	data.table	Extension of `data.frame`	1.13.0
☑	dplyr	A Grammar of Data Manipulation	1.0.0
☐	dslabs	Data Science Labs	0.7.3
☐	fastmap	Fast Implementation of a Key-Value Store	1.0.1

6.2LOADING UP THE DATA AND PLAYING WITH IT

The read.csv function with argument as path is created for loading the desired data csv files in the code.

- **read.csv ("path"):**



After running this line of code, we will observe that our dataset has been inserted in the environment. On top right panel we can see that it shows us the number of observations and variables.

Now, we can check our loaded dataset sample to check if it is right, suppose first 10 observations, to do so use :

- **head(dataset_name, n=no of observation rows we want to display)**

```
head(heart_disease,n=10)
```

	age <int>	sex <int>	cp <int>	trestbps <int>	chol <int>	fbs <int>	restecg <int>	thalach <int>	exang <int>
1	63	1	3	145	233	1	0	150	0
2	37	1	2	130	250	0	1	187	0
3	41	0	1	130	204	0	0	172	0
4	56	1	1	120	236	0	1	178	0
5	57	0	0	120	354	0	1	163	1
6	57	1	0	140	192	0	1	148	0
7	56	0	1	140	294	0	0	153	0
8	44	1	1	120	263	0	1	173	0
9	52	1	2	172	199	1	1	162	0
10	57	1	2	150	168	0	1	174	0

1–10 of 10 rows | 1–10 of 14 columns

chol <int>	fbs <int>	restecg <int>	thalach <int>	exang <int>	oldpeak <dbl>	slope <int>	ca <int>	thal <int>	target <int>
233	1	0	150	0	2.3	0	0	1	1
250	0	1	187	0	3.5	0	0	2	1
204	0	0	172	0	1.4	2	0	2	1
236	0	1	178	0	0.8	2	0	2	1
354	0	1	163	1	0.6	2	0	2	1
192	0	1	148	0	0.4	1	0	1	1
294	0	0	153	0	1.3	1	0	2	1
263	0	1	173	0	0.0	2	0	3	1
199	1	1	162	0	0.5	2	0	3	1
168	0	1	174	0	1.6	2	0	2	1

1–10 of 10 rows | 6–15 of 14 columns

We can see that using this we can see our csv file loaded up in an object/dataset_variable heart_disease and we can play with it in whatever way we want now.
Here we observe all our 14 variables/features and the first 10 observations/rows for it .

Another important function we can bring into play here is

- **glimpse(dataset_variable)**

```
#getting a glimpse of the data
glimpse(heart_disease)
```

```
Rows: 303
Columns: 14
$ age      <int> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 50, 58, 66, 43, 69, 5…
$ sex      <int> 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, …
$ cp       <int> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3, 0, 3, 0, 2, 0, 2, 3, 1, 2, …
$ trestbps <int> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 130, 110, 150, 120, 1…
$ chol     <int> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 266, 211, 283, 219, 3…
$ fbs      <int> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, …
$ restecg  <int> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ thalach  <int> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171, 144, 162, 158, 1…
$ exang    <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, …
$ oldpeak  <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2, 0.2, 0.6, 1.8, 1.0, 1.6, 0…
$ slope    <int> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1, 2, 2, 1, 2, 2, 2, …
$ ca       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, …
$ thal     <int> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, …
$ target   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
```

This is a very useful and handy function as it shows the properties of our variables as we can see here, they are int /numerical. So, our data is all numerical at present stage .

Now to check number of rows and column we can also use the functions

- **ncol(dataset_name)**
- **nrow(dataset_name)**

```
# To particularly get the no. of rows and columns
ncol(heart_disease)
nrow(heart_disease)
```

```
[1] 14
[1] 303
```

It shows we have 14 columns/variables and 303 rows/observations

Now if we want to check what attributes/features/variables are we using we can simply do it using function

colnames(dataset_name)

```
# To check the col/attribute names
colnames(heart_disease)
```

```
[1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"      "restecg"  "thalach"
[9] "exang"    "oldpeak"  "slope"    "ca"       "thal"     "target"
```

Here it shows us all the 14 variables

6.3 EXPLORATORY ANALYSIS:

Now, we have our heart_disease dataset all set and loaded we know our dataset and before applying any model to our data, we should do some exploratory data analysis (EDA)
This helps us to gather more knowledge about our variables and and then we can decide if we want to scale the data or not?
Exploratory data analysis helps us to understand the characteristics of the patients in the data. We need to get an idea of the value ranges of the variables and their distributions. This will also be helpful when we evaluate the clusters of patients from the algorithms. Are there more patients of one gender? What might an outlier look like?

```
# Remove id
heart_disease <- heart_disease[ , !(names(heart_disease) %in% c("id"))]
```

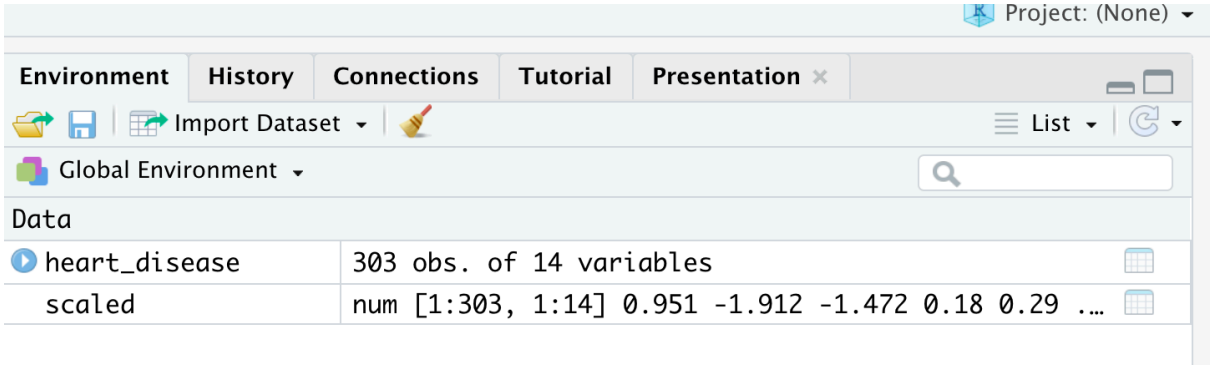
In our dataset before there was a 15th attribute called ID (a patient's ID), which was removed and data was scaled at this step

```
# Scaling data and saving as a data frame
scaled <- scale(heart_disease)

# What do the data look like now?
summary(scaled)
```

```
      age                sex               cp              trestbps            chol        
 Min.   :-2.79300   Min.   :-1.4660   Min.   :-0.93696   Min.   :-2.14525   Min.   :-2.3203  
 1st Qu.:-0.75603   1st Qu.:-1.4660   1st Qu.:-0.93696   1st Qu.:-0.66277   1st Qu.:-0.6804  
 Median : 0.06977   Median : 0.6799   Median : 0.03198   Median :-0.09259   Median :-0.1209  
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000  
 3rd Qu.: 0.73041   3rd Qu.: 0.6799   3rd Qu.: 1.00092   3rd Qu.: 0.47760   3rd Qu.: 0.5448  
 Max.   : 2.49212   Max.   : 0.6799   Max.   : 1.96986   Max.   : 3.89872   Max.   : 6.1303  
      fbs               restecg            thalach             exang             oldpeak       
 Min.   :-0.4169   Min.   :-1.0042   Min.   :-3.4336   Min.   :-0.6955   Min.   :-0.8954  
 1st Qu.:-0.4169   1st Qu.:-1.0042   1st Qu.:-0.7049   1st Qu.:-0.6955   1st Qu.:-0.8954  
 Median :-0.4169   Median : 0.8975   Median : 0.1464   Median :-0.6955   Median :-0.2064  
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.:-0.4169   3rd Qu.: 0.8975   3rd Qu.: 0.7139   3rd Qu.: 1.4331   3rd Qu.: 0.4827  
 Max.   : 2.3905   Max.   : 2.7991   Max.   : 2.2856   Max.   : 1.4331   Max.   : 4.4445  
     slope               ca                thal              target       
 Min.   :-2.2708   Min.   :-0.7132   Min.   :-3.7786   Min.   :-1.092  
 1st Qu.:-0.6480   1st Qu.:-0.7132   1st Qu.:-0.5121   1st Qu.:-1.092  
 Median :-0.6480   Median :-0.7132   Median :-0.5121   Median : 0.913  
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000  
 3rd Qu.: 0.9747   3rd Qu.: 0.2646   3rd Qu.: 1.1212   3rd Qu.: 0.913  
 Max.   : 0.9747   Max.   : 3.1983   Max.   : 1.1212   Max.   : 0.913
```

This is how scaling of the data is done, it is a very important step in our analysis

And to show the result of our scaled data we use **summary(new_dataset_name)** function and this summary function tells us the Min,1st, median, mean, 3rd and max values of the attributes for their observations .



We can see our scaled data set is loaded up in the environment

Till now, our data is numeric.

6.4 DATA TRANSFORMATION

Now moving to the next stage of our analysis, our data is still numeric and to observe the features of the attributes in a more clear and human readable form we change the values from numeric to categorical wherever possible.
We name our new categorical dataset as heart_disease2
To transfer our data , we use the **mutate()** function

```
heart_disease2<- heart_disease %>%
  mutate(sex=if_else(sex==1,"Male", "female"),
         fbs=if_else(fbs==1,">120", "<=120"),
         exang=if_else(exang==1,"Yes", "No"),
         cp=if_else(cp==1,"typical angina ",
                    if_else(cp==2," atypical angina ",
                    if_else(cp==3,"non-anginal pain ","asymptomatic "))),
         restecg=if_else(restecg==0,"Normal",
                         if_else(restecg==1,"Abnormal","Probable or Definite")),
          #Performiing coercion
         slope=as.factor(slope),
         ca=as.factor(ca),
         thal=as.factor(thal),
         target=if_else(target==1, "Yes", "No")
        ) %>%
  mutate_if(is.character,as.factor)%>%
  dplyr::select(target,sex,fbs,exang,cp,restecg,slope,ca,thal,everything())
```

Coercion means if we have some function with an argument of the wrong type, **R** will try to **coerce** values to a different type so that the function will work.
as.factor ()- This function converts a variable into a factor, but preserves variable and value label attributes.

check the clarity of data by observing the first 10 rows of it

```
#check the clarity of data by observing the first 10 rows of it
head(heart_disease2,n=10)
```

	target <fctr>	sex <fctr>	fbs <fctr>	exang <fctr>	cp <fctr>	restecg <fctr>	slope <fctr>	ca <fctr>	thal <fctr>
1	Yes	Male	>120	No	non-anginal pain	Normal	0	0	1
2	Yes	Male	<=120	No	atypical angina	Abnormal	0	0	2
3	Yes	female	<=120	No	typical angina	Normal	2	0	2
4	Yes	Male	<=120	No	typical angina	Abnormal	2	0	2
5	Yes	female	<=120	Yes	asymptomatic	Abnormal	2	0	2
6	Yes	Male	<=120	No	asymptomatic	Abnormal	1	0	1
7	Yes	female	<=120	No	typical angina	Normal	1	0	2
8	Yes	Male	<=120	No	typical angina	Abnormal	2	0	3
9	Yes	Male	>120	No	atypical angina	Abnormal	2	0	3
10	Yes	Male	<=120	No	atypical angina	Abnormal	2	0	2

1-10 of 10 rows | 1-10 of 14 columns

We can see that our data is in much clear and human readable form, our data is now transformed

6.5 REMOVING N/A VALUES

At this step it is urgent to remove all the N/A values because later during clustering process it can create issues

Is.na(dataset_name)

```
is.na(heart_disease2)
```

	target	sex	fbs	exang	cp	restecg	slope	ca	thal	age	trestbps	chol	thalach	oldpeak
[1,]	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
[3,]	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
[5,]	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
[7,]	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
[9,]	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
[11,]	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
[13,]	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
[15,]	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
[17,]	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
[19,]	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
[21,]	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
[23,]	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
[25,]	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
[27,]	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
[29,]	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

We can see that there is no TRUE value here

```
which(is.na(heart_disease2))
```

```
integer(0)
```

As it shows no N/A value no need to remove or replace

6.6DATA VISUALIZATION

In this section we will do some data visualization to get a broader idea of our data

So firstly, we visualized how many people have the heart disease

In how many people the disease is present

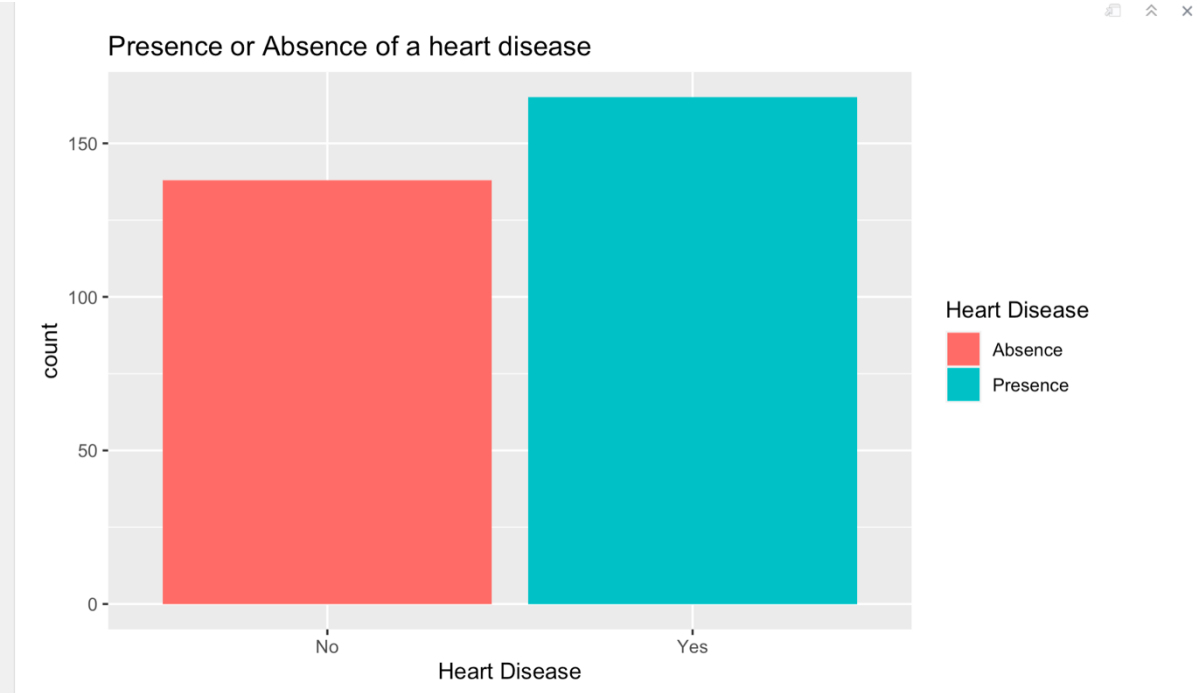
That mean we have to find how many values in our ‘target’ feature is Yes and how many are NO

Here we will use the package

ggplot()

```
#To visualize and find how many people are infected with the heart_disease, that is finding Target is yes
or no
ggplot(heart_disease2,aes(target,fill=target))+
  geom_bar() +
  xlab("Heart Disease")+
  ylab("count")+
  ggtitle("Presence or Absence of a heart disease")+
  scale_fill_discrete(name='Heart Disease', labels=c("Absence","Presence"))
```

ggplot(dataset_name, aes())
here aes means aesthetic mapping which is of the target variable here so we put targe in the aes field and fill means we will fill the target plotting with some color
+ here is used for concatenation
geom_bar- shows its a bar graph
xlab("x label goes here ")
ylab("y label goes here")
ggtitile("the title we want to keep ")
scale_filldiscrete – To make a separate identifying label on side with labels absence and presence showing different color



Here we can see that people with the presence of a heart disease are more than the people with absence we can conclude that the data is not that unbalanced , it is more bent towards presence of heart disease in the people

If we want to check this value vise
Use **prop.table( )method**

```
#Finding the proportions of presence and ansence of heart disease from Bar graph
prop.table(table(heart_disease2$target))
```

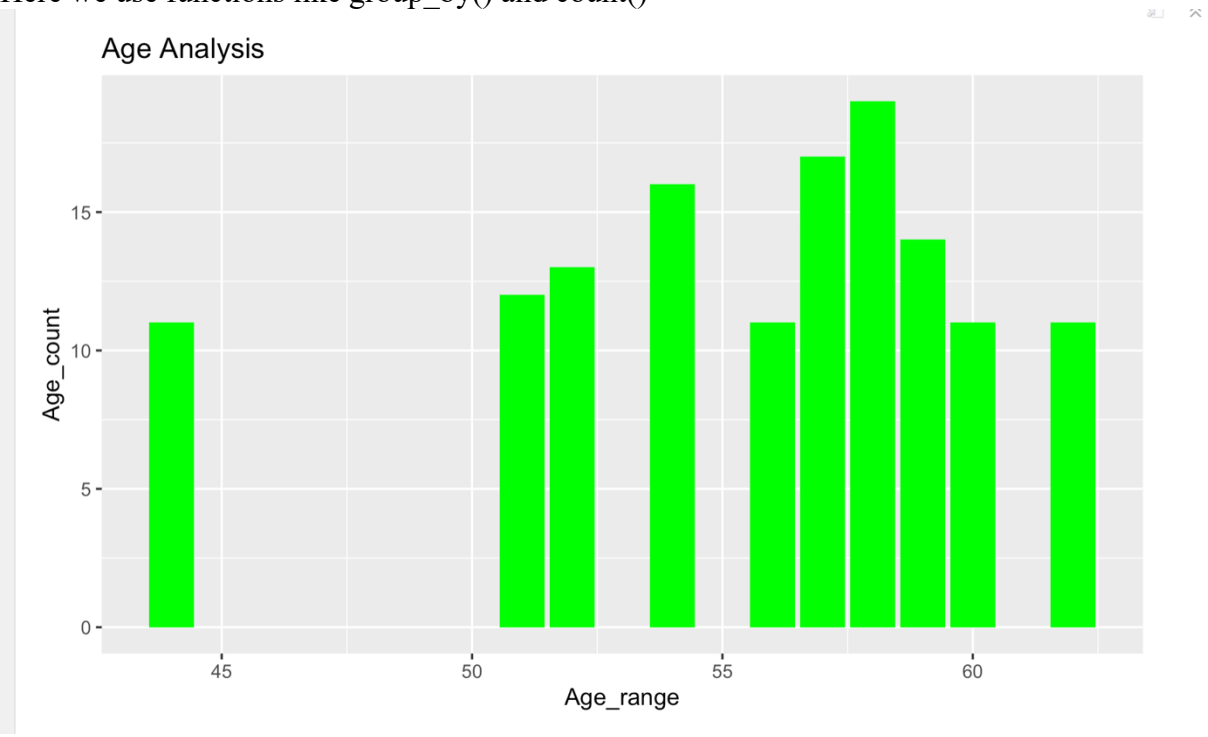
```
       No       Yes
0.4554455 0.5445545
```

We can clearly observe no of cases of presence =54.45%

Let's perform an analysis on age
Let's calculate frequency on value of age
Let's check which age group is more likely to have a heart disease

```
#Count the frequency of value of age
heart_disease2 %>%
  group_by(age) %>%
  count() %>%
  filter(n>10)%>%
  ggplot()+
  geom_col(aes(age,n), fill='green')+
  ggtitle("Age Analysis")+
  xlab("Age_range")+
  ylab("Age_count")
```

Here we use functions like group_by() and count()



From the analysis we can see a major gap between the age 45 and 50, this is a type of countplot(histgram plot for categorical variables)

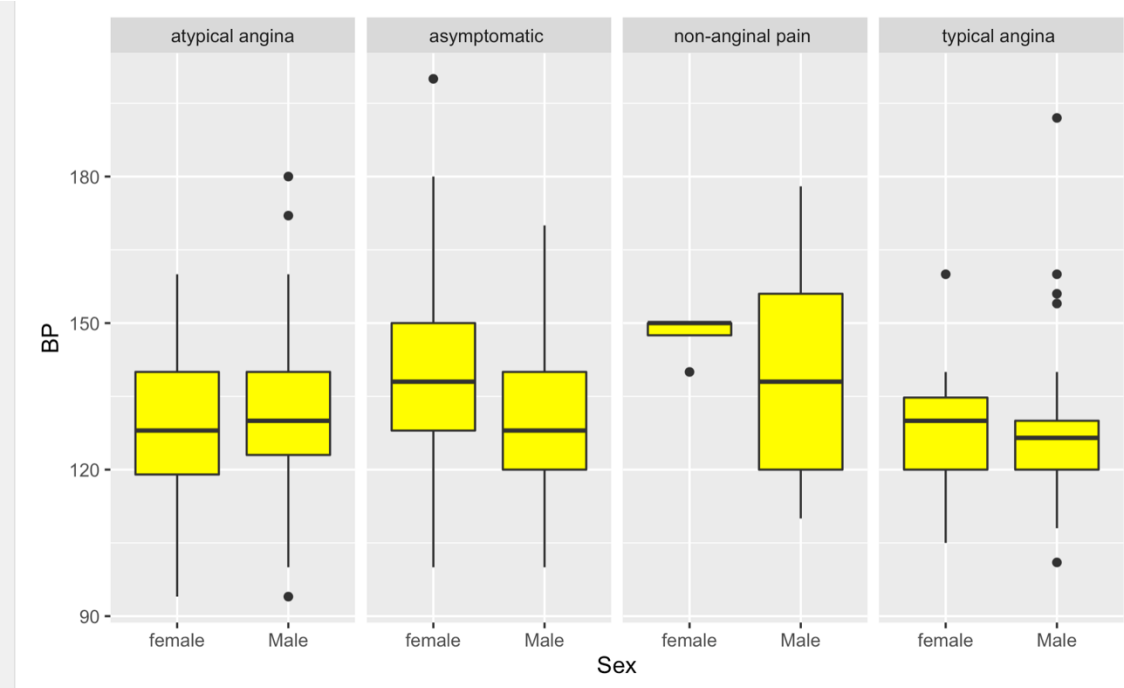
We can see how many people are falling in the age_range(in x axis) and the count will be in y axis

if we check the age_range between 55 and 60 we can see that a lot of patients have the heart disease, which might be because of their late stage in life (age)

#Compare blood pressure(trestbps) across cheast pain (cp)

```r
{r}
#Compare blood pressure(trestbps) across cheast pain (cp)
heart_disease2 %>%
  ggplot(aes(x=sex,y=trestbps))+
  geom_boxplot(fill="yellow")+
  xlab("Sex")+
  ylab("BP")+
  facet_grid(~cp)
```

Now we have transformed the dataset and given various different names for the cp attribute(atypical angina, asymptomatic, non anignal pain, typical anigna) and we have mapped the number of males and females having those cp symptoms



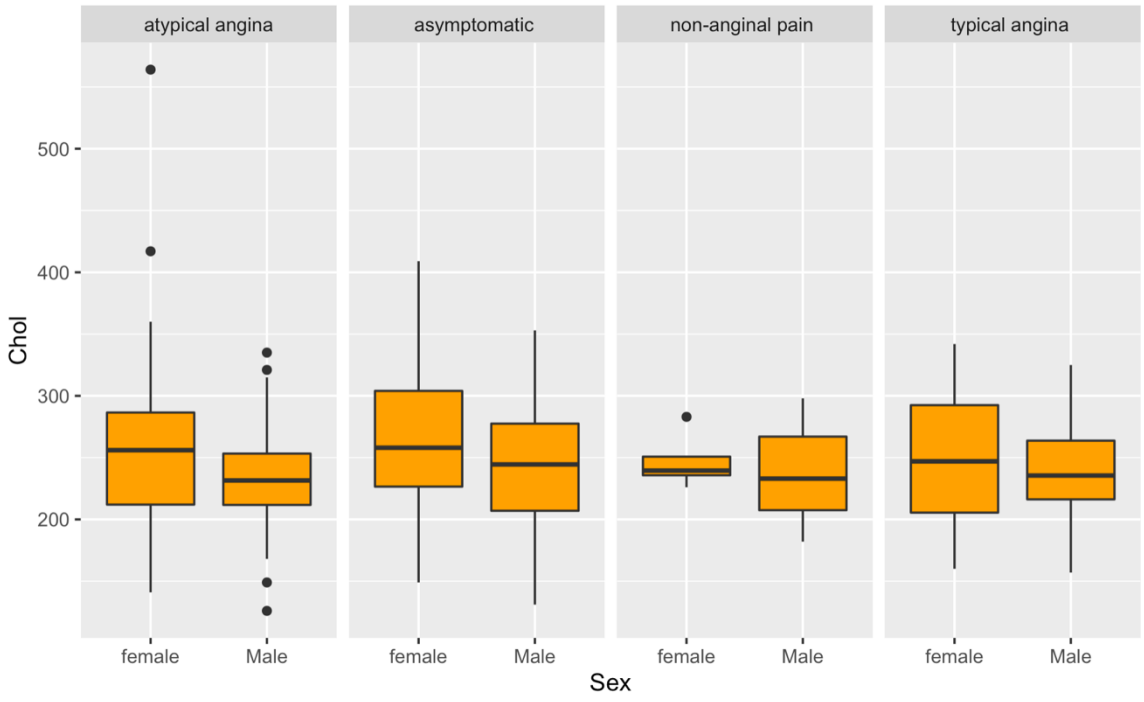
```
We notice the outliers here (the dots in the graph)
atypical angina - female = 0 outlier ; male= 3 outliers
asymptomatic    - female = 1 outlier ; male= 0 outlier
non anignal pain- female = 1 outlier ; male= 0 outlier
typical anigna  - female = 1 outlier ; male= 5 outliers

SO , what we notice from these outiers ?
atypical angina - female = 0 outlier ; male= 3 outliers - males have more/less than normal BP
asymptomatic    - female = 1 outlier ; male= 0 outlier - in this females have more than normal BP
non anignal pain- female = 1 outlier ; male= 0 outlier - in this females have less than normal BP
typical anigna  - female = 1 outlier ; male= 5 outliers - female/male have more than BP value some males
                                                          have less BP value

Now this was the visualization of the one of the feature/attribute with respect to the chest pain (cp), we
can perform this with other attributes too
```

In the same way we can check cholesterol levels across chest pain ~cp

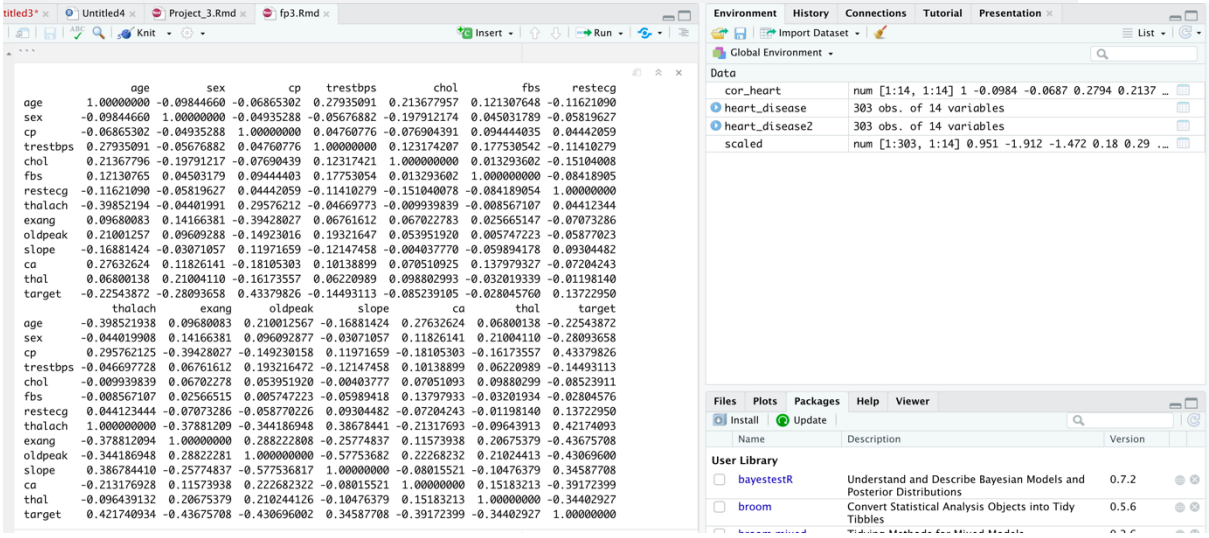
```
#Compare cholestrol(chol) across cheast pain (cp)
heart_disease2 %>%
  ggplot(aes(x=sex,y=chol))+
  geom_boxplot(fill="orange")+
  xlab("Sex")+
  ylab("Chol")+
  facet_grid(~cp)
```



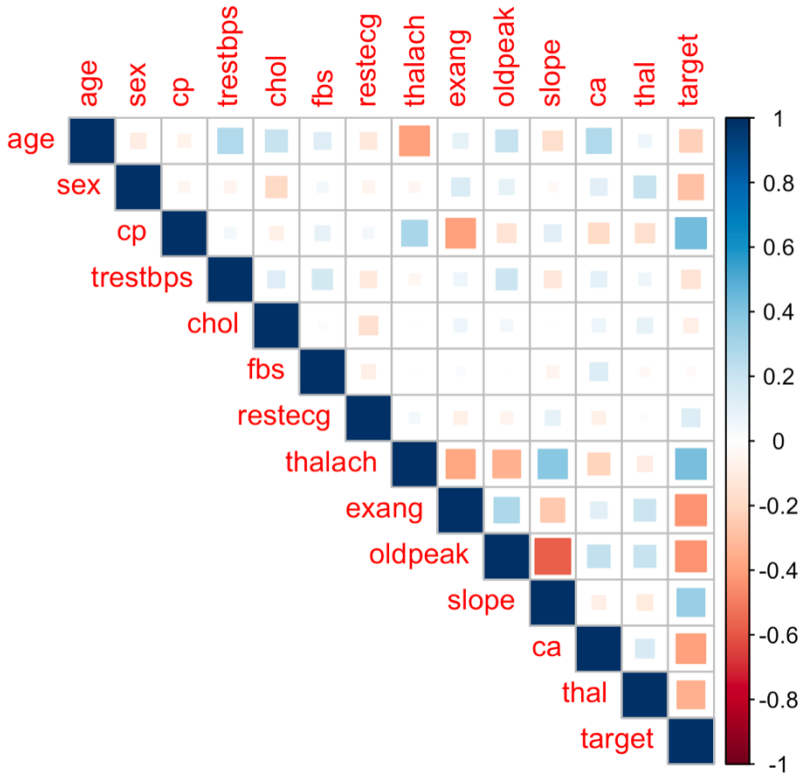
6.7 CORRELATION

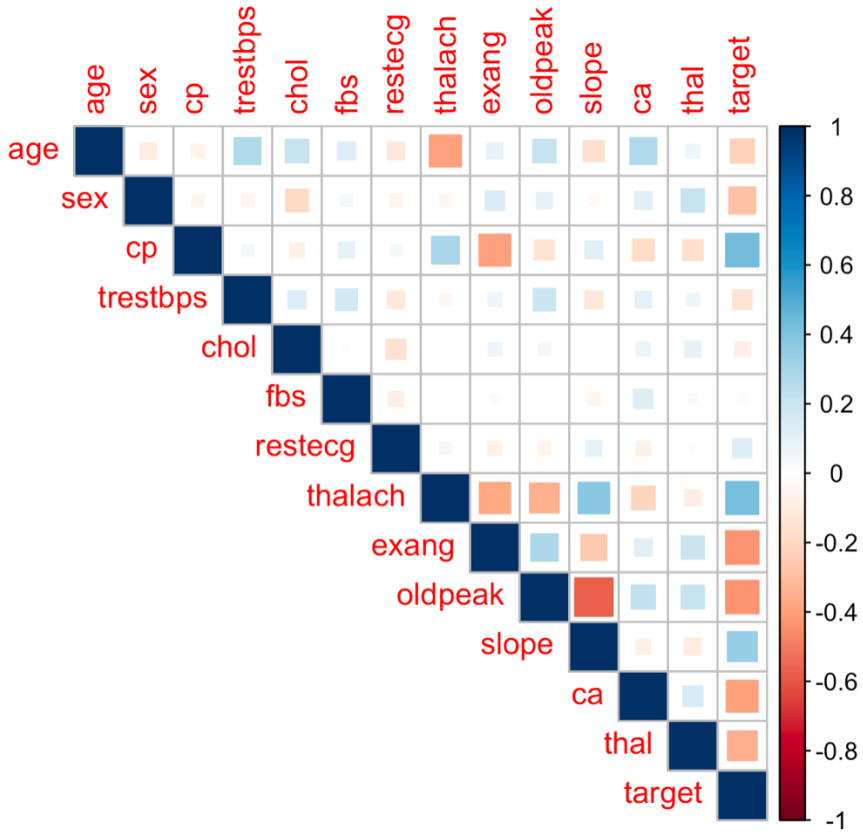
Now , let's find some correlation between the variables, between each of them so we come to know which factor is supporting which and what can be derived from it .
Correlation can be found out between the variables which
For this we use **cor(dataset_name[rowrange, colrange])**

```
## **6.Correlation**
```{r}
cor_heart<-cor(heart_disease2[,10:14])
cor_heart
```




```
              age         sex          cp    trestbps        chol         fbs      restecg
age     1.00000000 -0.09844660 -0.06865302  0.27935091  0.21367795 0.12130765 -0.11621090
sex    -0.09844660  1.00000000 -0.04935288 -0.05676882 -0.19791217 0.04503179 -0.05819627
cp     -0.06865302 -0.04935288  1.00000000  0.04760776 -0.07690439 0.09444403  0.04442059
trestbps 0.27935091 -0.05676882 0.04760776 1.00000000  0.12317421 0.17753054 -0.11410279
chol    0.21367796 -0.19791217 -0.07690439  0.12317421 1.00000000 0.01329360 -0.15104008
fbs     0.12130765  0.04503179  0.09444403  0.17753054 0.01329360 1.00000000 -0.08418905
restecg -0.11621090 -0.05819627 0.04442059 -0.11410279 -0.15104008 -0.08418905 1.00000000
thalach -0.39852194 -0.04401991 0.29576212 -0.04669773 -0.00993983 -0.00856710 0.04412344
exang   0.09680083  0.14166381 -0.39428027  0.06761612  0.06702278 0.02566515 -0.07073286
oldpeak 0.21001257  0.09609288 -0.14923016  0.19321647  0.05395192 0.00574722 -0.05877023
slope   -0.16881424 -0.03071057  0.11971659 -0.12147458 -0.00403777 -0.05989418 0.09304482
ca      0.27632624  0.11826141 -0.18105303  0.10138899  0.07051093 0.13797933 -0.07204243
thal    0.06800138  0.21004110 -0.16173557  0.06220989  0.09880299 -0.03201934 -0.01198140
target  -0.22543872 -0.28093658  0.43379826 -0.14493113 -0.08523911 -0.02804576 0.13722950
             thalach       exang     oldpeak       slope          ca        thal      target
age     -0.398521938  0.09680083  0.210012567 -0.16881424  0.27632624  0.06800138 -0.22543872
sex     -0.044019908  0.14166381  0.096092877 -0.03071057  0.11826141  0.21004110 -0.28093658
cp       0.295762125 -0.39428027 -0.149230158  0.11971659 -0.18105303 -0.16173557  0.43379826
trestbps -0.046697728  0.06761612  0.193216472 -0.12147458  0.10138899  0.06220989 -0.14493113
chol    -0.009939839  0.06702278  0.053951920 -0.00403777  0.07051093  0.09880299 -0.08523911
fbs     -0.008567107  0.02566515  0.005747223 -0.05989418  0.13797933 -0.03201934 -0.02804576
restecg  0.044123444 -0.07073286 -0.058770226  0.09304482 -0.07204243 -0.01198140  0.13722950
thalach  1.000000000 -0.37881209 -0.344186948  0.38678441 -0.21317693 -0.09643913  0.42174093
exang   -0.378812094  1.00000000  0.288222808 -0.25774837  0.11573938  0.20675379 -0.43675708
oldpeak -0.344186948  0.28822281  1.000000000 -0.57753682  0.22268232  0.21024413 -0.43069600
slope    0.386784410 -0.25774837 -0.577536817  1.00000000 -0.08015521 -0.10476379  0.34587708
ca      -0.213176928  0.11573938  0.222682322 -0.08015521  1.00000000  0.15183213 -0.39172399
thal    -0.096439132  0.20675379  0.210244126 -0.10476379  0.15183213  1.00000000 -0.34402927
target   0.421740934 -0.43675708 -0.430696002  0.34587708 -0.39172399 -0.34402927  1.00000000
```

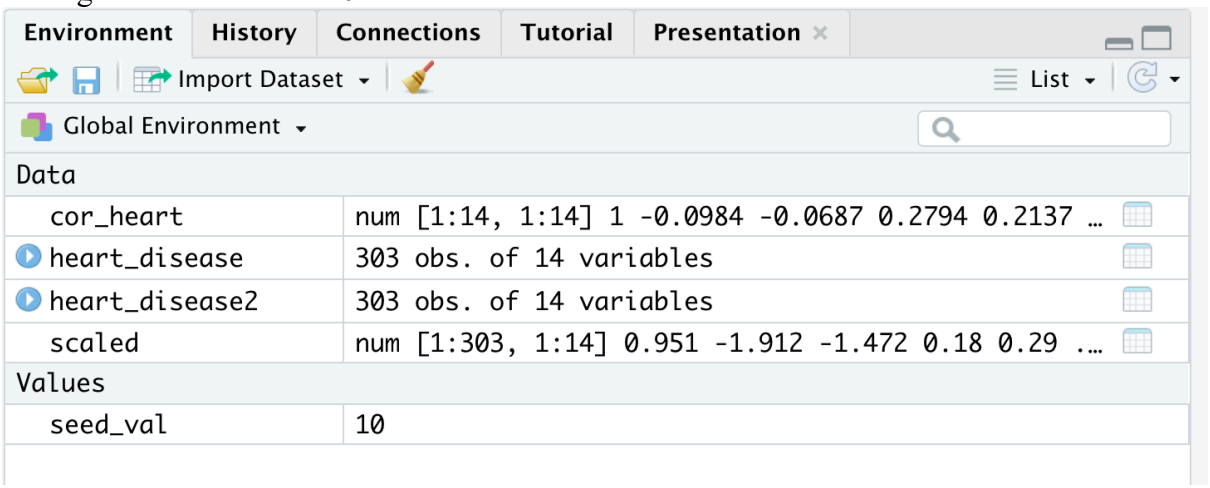
This will give us the correlation in values of variables with respect to each other.
If we want to see a plotted value for the same we will use
Corrplot(cor(datasetname[rowrange,colrange]), method="Type of plot we want ", type='what kind')
```

```
corrplot(cor_heart, method="square", type='upper')
```



We can notice the correlation between the different variables, only containing numerical variables.
The dark blue color shows high correlation between the variables
The light blue squares show that it's a fair correlation between the variables
The light red shows the negative correlation

If we check the correlation table, we can see how some variables like age and age having correlation value 1 are showing highest value and some show negative values (thalach and age)

If we want to see the correlation between all the 14 variables, we can use our scaled dataset with all the numerical values

```
```{r}
# Using all the variables in dataset scaled and checking the correlation between all of them
cor_heart<-cor(scaled[,1:14])
cor_heart
```
```

```
                age         sex          cp    trestbps         chol          fbs     restecg
age      1.00000000 -0.09844660 -0.06865302  0.27935091  0.213677957  0.121307648 -0.11621090
sex     -0.09844660  1.00000000 -0.04935288 -0.05676882 -0.197912174  0.045031789 -0.05819627
cp      -0.06865302 -0.04935288  1.00000000  0.04760776 -0.076904391  0.094444035  0.04442059
trestbps 0.27935091 -0.05676882  0.04760776  1.00000000  0.123174207  0.177530542 -0.11410279
chol     0.21367796 -0.19791217 -0.07690439  0.12317421  1.000000000  0.013293602 -0.15104008
fbs      0.12130765  0.04503179  0.09444403  0.17753054  0.013293602  1.000000000 -0.08418905
restecg -0.11621090 -0.05819627  0.04442059 -0.11410279 -0.151040078 -0.084189054  1.00000000
thalach -0.39852194 -0.04401991  0.29576212 -0.04669773 -0.009939839 -0.008567107  0.04412344
exang    0.09680083  0.14166381 -0.39428027  0.06761612  0.067022783  0.025665147 -0.07073286
oldpeak  0.21001257  0.09609288 -0.14923016  0.19321647  0.053951920  0.005747223 -0.05877023
slope   -0.16881424 -0.03071057  0.11971659 -0.12147458 -0.004037770 -0.059894178  0.09304482
ca       0.27632624  0.11826141 -0.18105303  0.10138899  0.070510925  0.137979327 -0.07204243
thal     0.06800138  0.21004110 -0.16173557  0.06220989  0.098802993 -0.032019339 -0.01198140
target  -0.22543872 -0.28093658  0.43379826 -0.14493113 -0.085239105 -0.028045760  0.13722950
             thalach       exang      oldpeak       slope          ca        thal      target
age     -0.398521938  0.09680083  0.210012567 -0.16881424  0.27632624  0.06800138 -0.22543872
sex     -0.044019908  0.14166381  0.096092877 -0.03071057  0.11826141  0.21004110 -0.28093658
cp       0.295762125 -0.39428027 -0.149230158  0.11971659 -0.18105303 -0.16173557  0.43379826
trestbps -0.046697728  0.06761612  0.193216472 -0.12147458  0.10138899  0.06220989 -0.14493113
chol     -0.009939839  0.06702278  0.053951920 -0.00403777  0.07051093  0.09880299 -0.08523911
fbs      -0.008567107  0.02566515  0.005747223 -0.05989418  0.13797933 -0.03201934 -0.02804576
restecg   0.044123444 -0.07073286 -0.058770226  0.09304482 -0.07204243 -0.01198140  0.13722950
thalach   1.000000000 -0.37881209 -0.344186948  0.38678441 -0.21317693 -0.09643913  0.42174093
exang    -0.378812094  1.00000000  0.288222808 -0.25774837  0.11573938  0.20675379 -0.43675708
oldpeak  -0.344186948  0.28822281  1.000000000 -0.57753682  0.22268232  0.21024413 -0.43069600
slope     0.386784410 -0.25774837 -0.577536817  1.00000000 -0.08015521 -0.10476379  0.34587708
ca       -0.213176928  0.11573938  0.222682322 -0.08015521  1.00000000  0.15183213 -0.39172399
thal     -0.096439132  0.20675379  0.210244126 -0.10476379  0.15183213  1.00000000 -0.34402927
target    0.421740934 -0.43675708 -0.430696002  0.34587708 -0.39172399 -0.34402927  1.00000000
```

```
corrplot(cor_heart, method="square", type='upper')
```
```



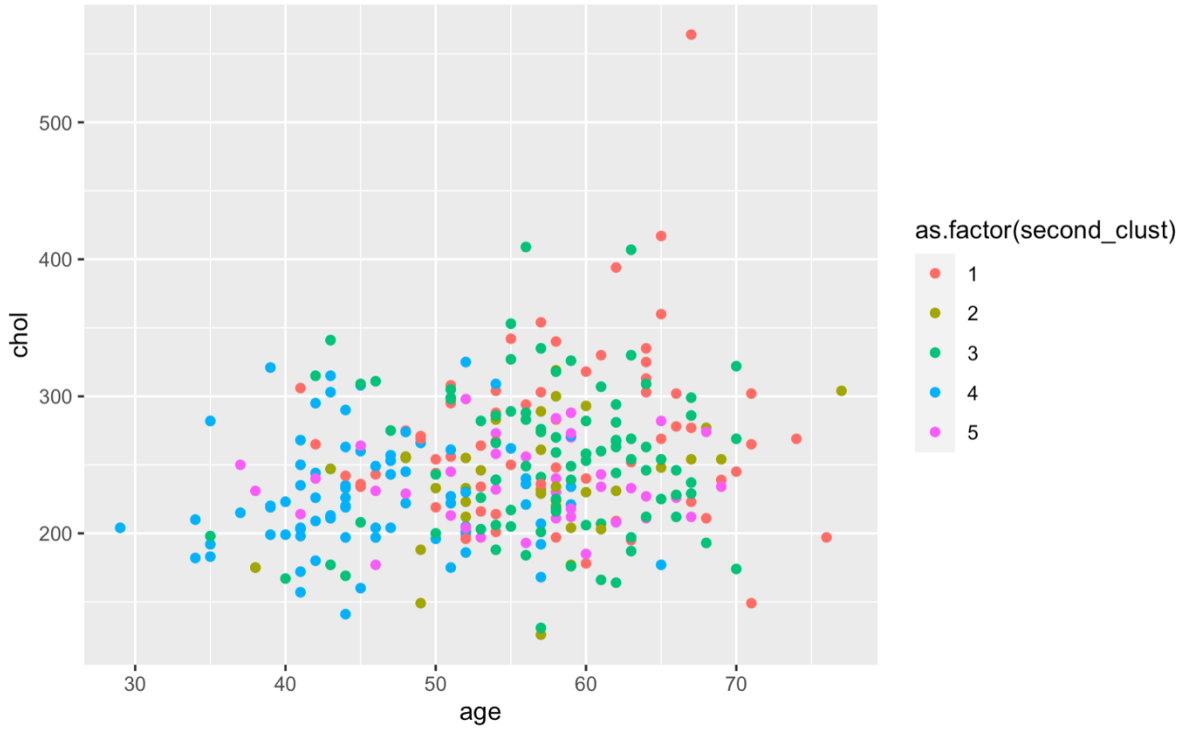
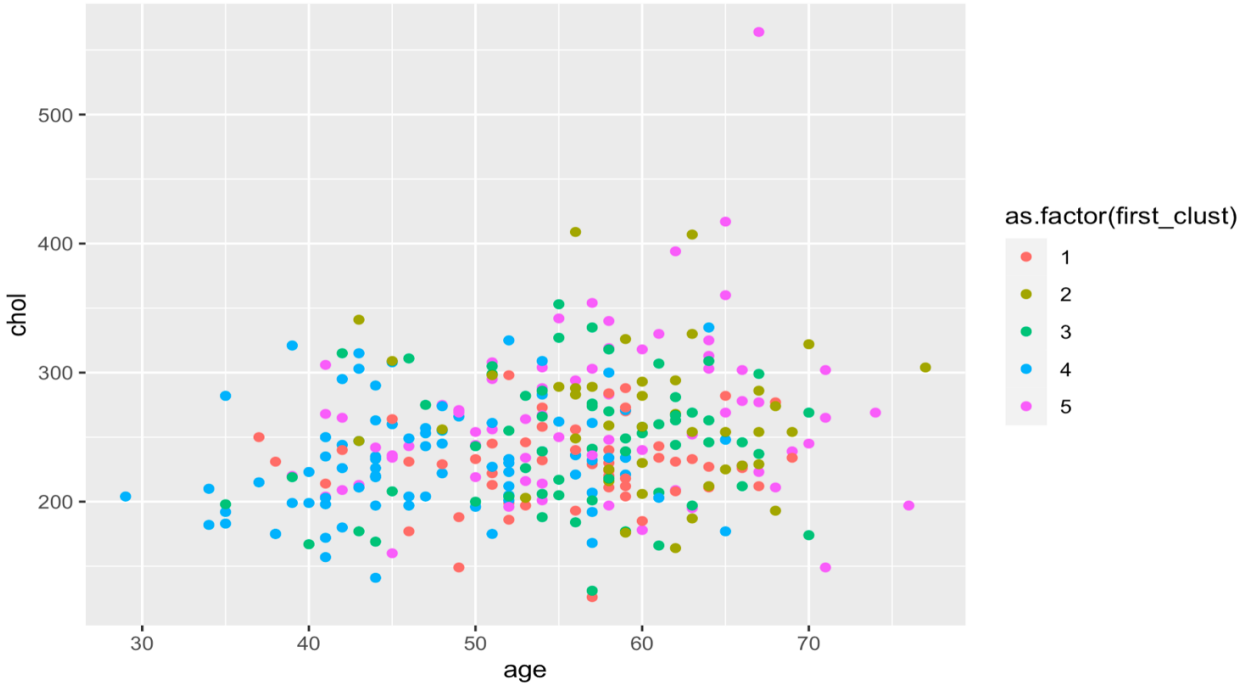
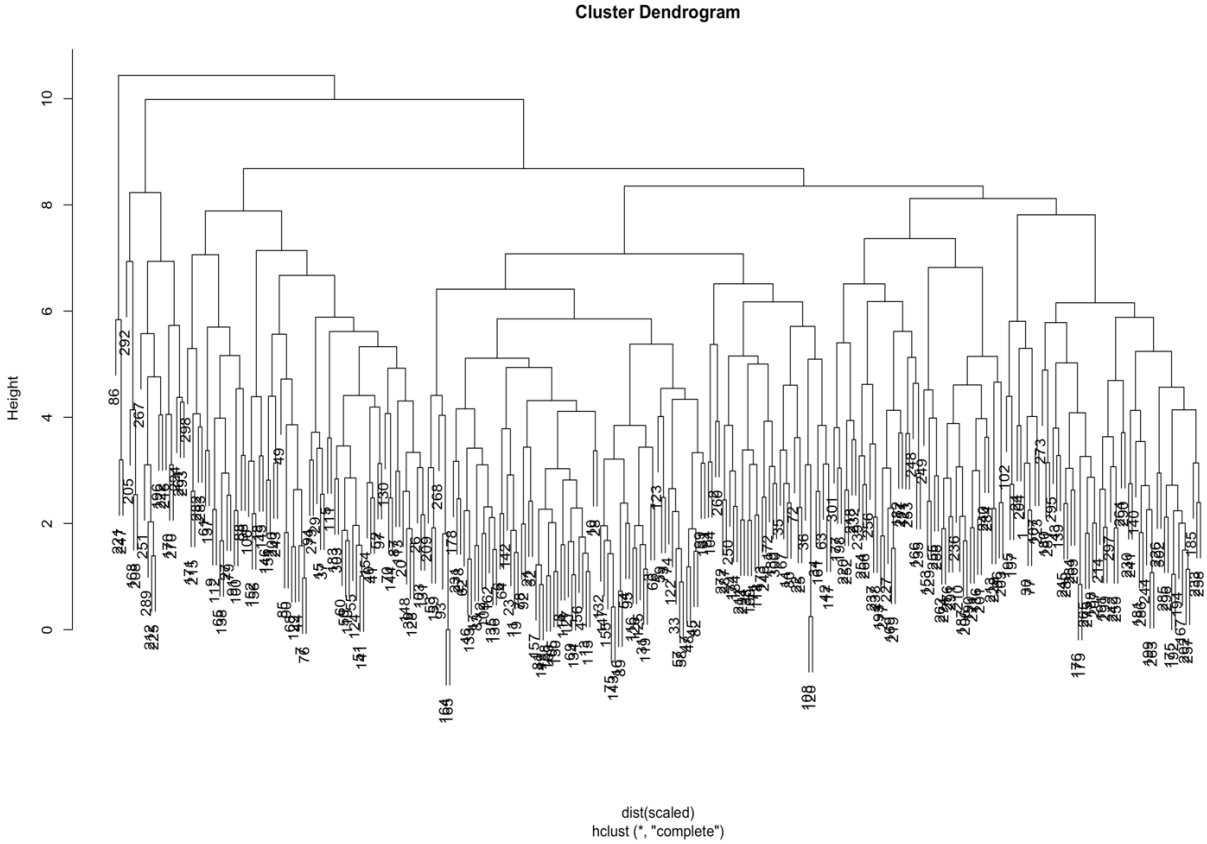
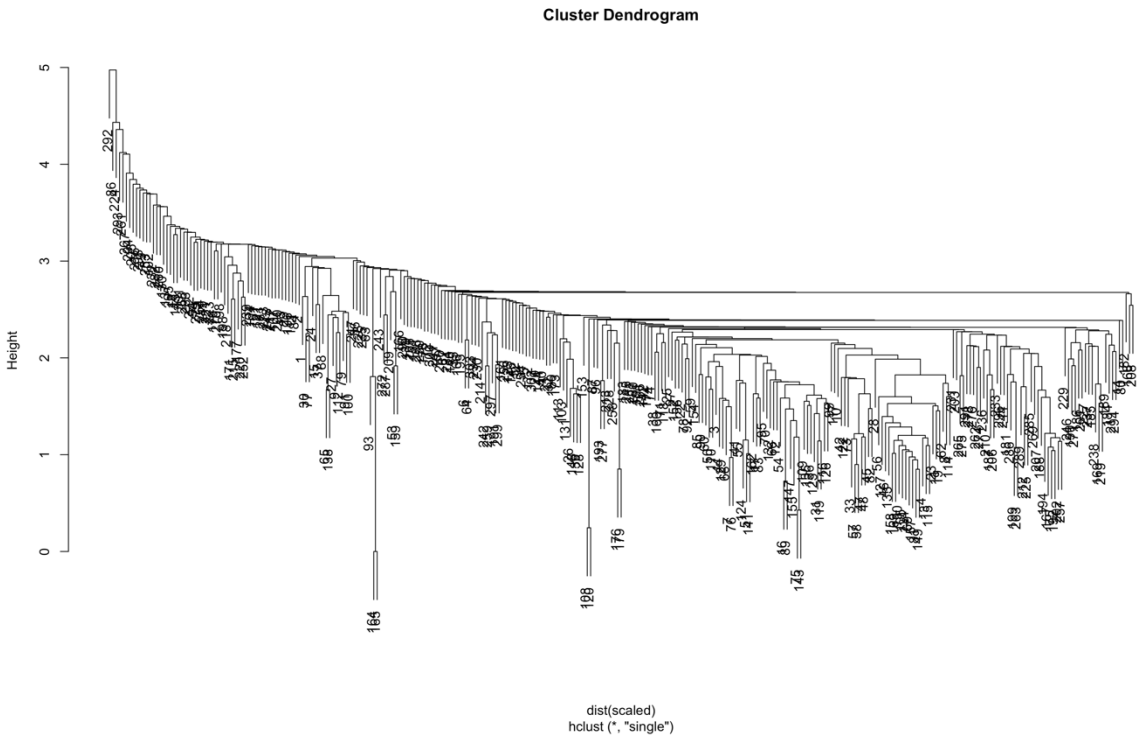

It is very important to check the correlation between the variables as they tell how each variable are related to each other.

## *6.8 LET'S START GROUPING PATIENTS*

Now that we have cleaned, transformed, scaled and visualized the data, we can start the clustering process.
Clustering can be done in hierarchical agglomerative, partitioning(K-Means), and model-based ways

For the k-means algorithm, it is necessary to select the number of clusters in advance.
It is also important to make sure that our results are reproducible when conducting a statistical analysis.
This means that when someone runs our code on the same data, they will get the same results. Because we are doing an analysis that has a random aspect, it is necessary to set a seed to ensure reproducibility.
Reproducibility is especially important because doctors will potentially use our results to treat patients. It is vital that other analysts see where the groups come from and can verify the results.
```

```
# Set the seed so that results are reproducible
seed_val <- 10
set.seed(seed_val)
```

Setting the seed value as 10



We can see that seed value is set to be 10 in our environment

Now we need to set the number of clusters
Let's say we need to put number of clusters as 5

```
# Select a number of clusters
k <- 5
```

Value of k will be loaded up in our environment

Now we will run our k means algo and name it as first_clust

```
# Run the k-means algorithm
first_clust <- kmeans(heart_disease2, centers = k, nstart = 1)
```

Here we use:
kmeans(dataset_name, centre = here we set centre as number of clusters, nstart=from where the clustering starts from)

```
Error in do_one(nmeth) : NA/NaN/Inf in foreign function call (arg 1)
```

We will get this error
Since we used our heart_disease2 dataset with non-numerical values so that's why it is showing this error
K- means cannot handle non-numerical value
So for this part we will use the numerical dataset which we scaled before named scaled

```
# Run the k-means algorithm
first_clust <- kmeans(scaled, centers = k, nstart = 1)
```

first_clust | List of 9

So , lets now find the center of all the clusters

```
# Find the centres of the clusters
first_clust$centers
```

```
        age        sex          cp    trestbps        chol        fbs     restecg    thalach
1  0.1205890  0.5973470  1.22452350  0.12891063 -0.33753699  0.4468793 -0.19962721  0.2253129
2  0.6256750  0.1041585 -0.84243363  0.68898713  0.36430719  0.7471108 -0.72588072 -0.4807964
3  0.1945581  0.3222350 -0.80777225 -0.08593246 -0.03081877 -0.3701543  0.29528895 -1.0491172
4 -0.7648653  0.4944347  0.02001606 -0.32488254 -0.27853862 -0.2783064  0.33402623  0.7791677
5  0.2660477 -1.3726936  0.25666074 -0.05044046  0.39168236 -0.1321332 -0.06712663  0.1134902
       exang    oldpeak      slope          ca        thal     target
1 -0.3680046  0.2756165 -0.5544192 -0.09266278  0.08468929  0.1419912
2  0.6024410  0.9342947 -0.6876212  1.26638886  0.64315014 -1.0916529
3  1.1492983  0.5486835 -0.5669022  0.02017107  0.27732892 -0.9580082
4 -0.5115278 -0.6752780  0.8344993 -0.27862969 -0.12896698  0.4675362
5 -0.4795361 -0.4472701  0.3397385 -0.37311214 -0.53574505  0.8258592
```

Here we can see the center of clusters for each attributes

Now to see which part lies in which cluster number

```
# Show the clusters
first_clust$cluster
```

first_clust$cluster

Here if we see the observation of assigned cluster, we see clusters have been assigned from 1 to 5 and we can note what observation has been classified to which cluster

```
# How many patients are in each cluster?
first_clust$size
```

```
[1] 52 41 60 81 69
```

We can see that there are 5 clusters and 52 , 41, 60, 81, 69 represent the number of patients in each clusters or the size of the cluster

Because the k-means algorithm initially selects the cluster centers by randomly selecting points, different iterations of the algorithm can result in different clusters.

If the algorithm is genuinely grouping similar observations (as opposed to clustering noise), then cluster assignments will be somewhat robust between various iterations of the algorithm. With regards to the heart disease data, this would mean that the same patients would be grouped even when the algorithm is initialized at different random points.

If patients are not in similar clusters with various algorithm runs, then the clustering method is not picking up on meaningful relationships between patients.

We're going to explore how the patients are grouped with another iteration of the k-means algorithm. We will then be able to compare the resulting groups of patients.

```
# Set the seed so that results are reproducible
seed_val <- 38
set.seed(seed_val)

# Select a number of clusters
k <- 5
# Run the k-means algorithm
second_clust <- kmeans(scaled, centers = k, nstart = 1)
# Find the centres of the clusters
second_clust$centers
# Show the clusters
second_clust$cluster
# How many patients are in each cluster?
second_clust$size
```

```
         age        sex         cp     trestbps       chol         fbs     restecg     thalach
1  0.4332974 -1.3297465  0.2319189 -0.001173886  0.4962325 -0.14957067 -0.06843902  0.07709327
2  0.1798773  0.6185699  0.0596624  0.032856251 -0.2096058  0.38517758  0.08248545  0.38214686
3  0.3688103  0.1677972 -0.8488789  0.225553134  0.1367575 -0.06601626 -0.11817549 -0.95994844
4 -0.9297693  0.3175903  0.1074804 -0.450987128 -0.3538816 -0.34402458  0.32945276  0.76497897
5  0.1385872  0.5725869  1.2431570  0.345032977 -0.2178056  0.70602653 -0.33859381  0.18350167
       exang    oldpeak      slope         ca        thal     target
1 -0.4927571 -0.4264664  0.3822959 -0.3717624 -0.53799936  0.8175583
2 -0.3305788 -0.4229120  0.5110880  1.4101767  0.56120315 -0.5188896
3  0.9735284  0.8467354 -0.7033633  0.4535556  0.39734797 -1.0005315
4 -0.4466838 -0.6425920  0.6375384 -0.5989497 -0.23633143  0.6526718
5 -0.2165472  0.4159043 -0.7291802 -0.4932230 -0.06293129  0.1111501
```

```
# Find the centres of the clusters
second_clust$centers
# Show the clusters
second_clust$cluster
# How many patients are in each cluster?
second_clust$size
```

```
  [1] 5 5 4 4 1 4 1 4 5 4 4 1 4 5 1 1 1 5 4 1 4 4 4 5 4 1 5 4 1 5 4 4 4 5 5 5 1 5 1 1 1 4 3 1 4 4 4 4
 [49] 1 1 1 1 2 4 1 4 4 4 4 1 1 4 4 4 5 4 4 1 4 1 5 4 4 4 4 1 5 4 4 5 4 4 1 5 1 1 2 4 1 1 2 4 2 1 4 3
 [97] 1 2 4 2 4 5 1 5 4 1 5 1 1 1 1 2 1 4 4 4 5 5 4 1 1 4 4 1 4 4 4 1 1 1 1 1 4 4 1 1 1 5 3 3 1 4 4 1
[145] 1 1 1 1 4 4 1 1 5 1 4 1 4 4 2 4 4 1 4 2 2 3 3 3 3 3 5 5 5 2 3 3 2 1 3 3 3 3 1 5 3 4 3 3 2 4 3 3
[193] 3 3 5 3 5 2 3 2 4 3 3 5 3 2 3 3 2 2 2 3 4 3 3 3 3 3 3 2 3 3 5 3 3 3 3 3 5 3 4 2 3 3 3 3 2 2 2 4
[241] 3 3 3 3 3 4 3 3 2 2 3 2 3 3 5 3 3 3 3 5 3 4 3 3 3 3 3 2 3 3 4 5 3 2 3 2 3 2 2 3 3 5 5 4 3 3 2 2
[289] 3 3 2 3 3 5 3 3 3 3 3 5 3 3 1
```

```
# How many patients are in each cluster?
second_clust$size
```

```
[1] 63 35 88 77 40
```

6.9COMPARING CLUSTERS (K-MEANS)

Even though our algorithm initially starts by randomly initializing the cluster centers ,

- If the k-means is the right choice for the data, then different initialization of the algorithm will result in similar clusters.
- The clusters from different iterations may not be the same, but the clusters should be roughly the same size and have similar distributions of variables.
- If there is a lot of change in clusters between different iterations of the algorithm, then k-means clustering is not the right choice for the data.
- It is not possible to validate that the clusters obtained from the algorithm are accurate because there is no patient labeling.
- Thus, it is necessary to examine how the clusters change between different iterations of the algorithm. We're going to use some visualizations to get an idea of the cluster stabilities. That way we can see how certain patient characteristics may have been used to group patients together.

The first thing to do is adding our clusters to the data set

```
heart_disease2["first_clust"] <- first_clust$cluster
heart_disease2["second_clust"] <- second_clust$cluster

heart_disease2
```

We can see that first_clust and secod_clus have been added to our dataset heart_disease2

slope <fctr>	ca <fctr>	thal <fctr>	age <int>	trestbps <int>	chol <int>	thalach <int>	oldpeak <dbl>	first_clust <int>	second_clust <int>
0	0	1	63	145	233	150	2.3	1	5
0	0	2	37	130	250	187	3.5	1	5
2	0	2	41	130	204	172	1.4	5	4
2	0	2	56	120	236	178	0.8	4	4
2	0	2	57	120	354	163	0.6	5	1
1	0	1	57	140	192	148	0.4	4	4
1	0	2	56	140	294	153	1.3	5	1
2	0	3	44	120	263	173	0.0	4	4
2	0	3	52	172	199	162	0.5	1	5
2	0	2	57	150	168	174	1.6	4	4

Now, choosing certain characteristics/attributes from the dataset to visualize as there are no labels of patients and no other way to cluster them

Choosing age and chol with respect to the first cluster

```
# Create and print the plot of age and chol for the first clustering algorithm
plot_one <- ggplot(heart_disease2, aes(x=age, y=chol, color=as.factor(first_clust))) +
  geom_point()
plot_one
```

Here, we can observe our 5 clusters as plotted with their respective different colors

Now, doing the same for second cluster with respect to the variables age and chol :

```
# Create and print the plot of age and chol for the second clustering algorithm
plot_two <- ggplot(heart_disease2, aes(x=age, y=chol, color=as.factor(second_clust))) +
  geom_point()
plot_two
```



If, we observe both the cluster plots we can easily observe that they are not similar



FIRST CLUSTER PLOT

SECOND CLUSTER PLOT

NOT SIMILAR → WILL NOT USE K-MEANS CLUSTERING

6.10 HIERARICHAL CLUSTERING

- Another really important and interesting method for clustering is hierarchal clustering.
- This works well with data having a nested structure. Our data might follow this structure. For example, if men are more likely to exhibit specific characteristics, those characteristics might be nested inside the gender variable
- Hierarchical clustering also does not require the number of clusters to be selected before running the algorithm, they can be selected using the dendrogram
- Dendrogram allows us to see how similar observations are to one another, and they are useful in helping us choose the number of clusters to group the data

Since clustering requires numeric data, we will use the scaled dataset here

```
# Execute hierarchical clustering with complete linkage
hier_clust_1 <- hclust(dist(scaled), method = "complete")
```

Here we are using the hclust() method , all the hierarchal clusters uses distance as a means to form clusters , here we are choosing our method to be complete ,
So, our dendrogram will be complete.

Plotting our dendrogram

```
# Print the dendrogram
plot(hier_clust_1)
```


Cluster Dendrogram
Height
0
2
4
6
8
10
dist(scaled)
hclust (*, "complete")

Now, assigning the cluster with a desired number of groups, here 5

```
# Get cluster assignments based on number of selected clusters
hc_1_assign <- cutree(hier_clust_1, 5)
```

We are using **cutree()** method here to mention desired number of groups to cut height

cutree(tree, k = **NULL**, h = **NULL**)

`tree`	a tree as produced by `hclust`. `cutree()` only expects a list with components `merge`, `height`, and`labels`, of appropriate content each.
`k`	an integer scalar or vector with the desired number of groups
`h`	numeric scalar or vector with heights where the tree should be cut.

At least one of `k` or `h` must be specified, `k` overrides `h` if both are given.

6.11 ROUND 2 HIERARICHAL CLUSTERING

- In hierarchical clustering, there are multiple ways to measure the dissimilarity between clusters of observations.
- Complete linkage records the largest dissimilarity between any two points in the two clusters being compared
- On the other hand, single linkage is the smallest dissimilarity between any two points in the clusters
- Different linkages will result in different clusters being formed.
- We want to explore different algorithms to group our heart disease patients. The best way to measure dissimilarity between patients could be to look at the smallest difference between patients and minimize that difference when grouping together clusters.
- It is always a good idea to explore different dissimilarity measures. Let's implement hierarchical clustering using a new linkage function.

```
# Execute hierarchical clustering with single linkage
hier_clust_2 <- hclust(dist(scaled), method = "single")
```

Here we are using single linkage as our method and naming it as hier_clust_2

```
# Print the dendrogram
plot(hier_clust_2)
```



This is our single linked tree like structure

Now assigning this cluster to our dataset

```
# Get cluster assignments based on number of selected clusters
hc_2_assign <- cutree(hier_clust_2, 5)
```

6.12 COMPARING CLUSTERS (HIERARCHICAL)

The doctors are interested in grouping similar patients together to determine appropriate treatments. Therefore, they want clusters with more than a few patients to see different treatment options.

While a patient can be in a cluster by themselves, this means that the treatment they received might not be recommended for someone else in the group.

Like the k-means algorithm, the way to evaluate hierarchical clusters is to investigate which patients are grouped together.

Are there patterns evident in the cluster assignments, or do they seem to be groups of noise? We're going to examine the clusters resulting from the two hierarchical algorithms.

```
```{r}
heart_disease["hc_clust"] <- hc_1_assign
```

We will put our hc_1_assign data in heart_disease[“hc_clust”] making a new attribute/column in our dataset representing the hierarichal clusters

```
heart_disease
```

fbs <int>	restecg <int>	thalach <int>	exang <int>	oldpeak <dbl>	slope <int>	ca <int>	thal <int>	target <int>	hc_clust <int>
1	0	150	0	2.3	0	0	1	1	1
0	1	187	0	3.5	0	0	2	1	2
0	0	172	0	1.4	2	0	2	1	2
0	1	178	0	0.8	2	0	2	1	2
0	1	163	1	0.6	2	0	2	1	3
0	1	148	0	0.4	1	0	1	1	2
0	0	153	0	1.3	1	0	2	1	3
0	1	173	0	0.0	2	0	3	1	2
1	1	162	0	0.5	2	0	3	1	3
0	1	174	0	1.6	2	0	2	1	2

1–10 of 303 rows | 6–15 of 15 columns Previous 1 2 3 4 5 6 … 31 Next

We can see here that how hc_clust column have been assigned

```
# Remove the sex, first_clust, and second_clust variables
hd_simple <- heart_disease[, !(names(heart_disease) %in% c("sex", "first_clust", "second_clust"))]
```

Removing these variables so as to get the mean and standard deviation

```
# Get the mean and standard deviation summary statistics
clust_summary <- do.call(data.frame, aggregate(. ~hc_clust, data = hd_simple, function(x) c(avg = mean(x),
sd = sd(x))))
```
```

```
clust_summary
```

slope.avg <dbl>	slope.sd <dbl>	ca.avg <dbl>	ca.sd <dbl>	thal.avg <dbl>	thal.sd <dbl>	target.avg <dbl>	target.sd <dbl>
1.2038835	0.5306161	1.0485437	1.0134260	2.572816	0.6199611	0.1165049	0.3223982
1.5663717	0.5957604	0.4159292	0.9794830	2.230088	0.4819760	0.8407080	0.3675783
1.6567164	0.5090764	0.5820896	0.8375515	1.940299	0.5471856	0.8507463	0.3590278
1.0000000	0.0000000	1.6666667	1.5275252	3.000000	0.0000000	0.3333333	0.5773503
0.5294118	0.5144958	1.2941176	1.1599949	2.647059	0.7018882	0.0000000	0.0000000

Here we get the mean and standard deviation of all the attributes, This we can see the distribution of variables in each of the hierarchical clustering runs.

We will make final visualization, to evaluate the algorithms

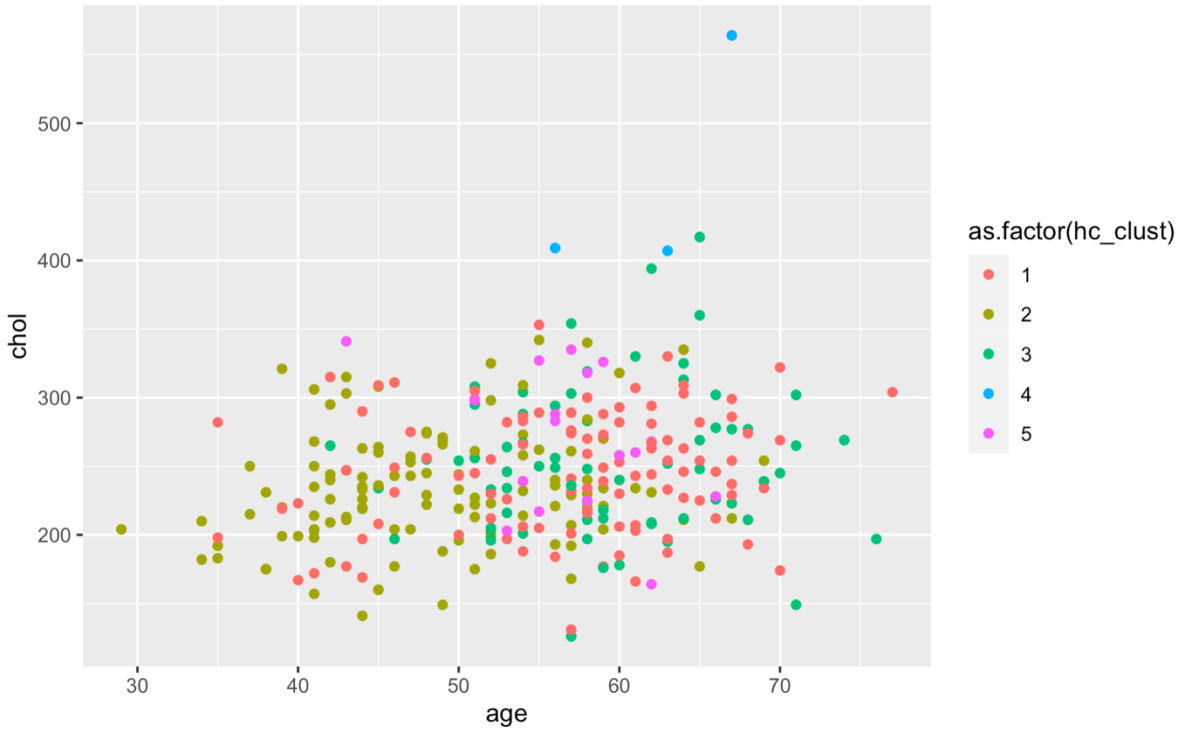
6.13 VISUALIZING THE CLUSTER CONTENT (HIERARCHICAL)

- Even though the data has more than two dimensions, we can get an idea of how the data clusters by looking at a scatter plot of two variables.
- We want to look for patterns that appear in the data and see what patients get clustered together.

```
# Plot age and chol
plot_one <- ggplot(heart_disease, aes(x = age, y = chol,
                                      color = as.factor(hc_clust))) +
  geom_point()
```

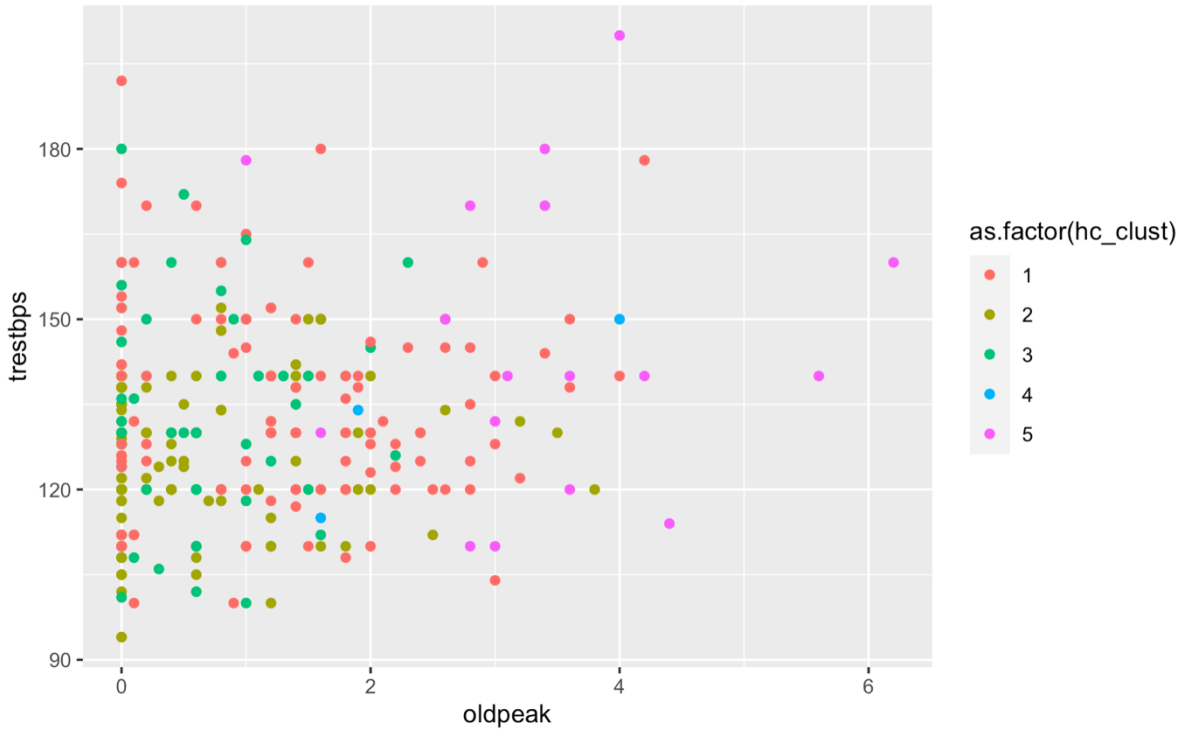
Checking with respect to age and chol in hierarchical clustering

```
plot_one
```



Now plotting with respect to oldpeak and trestbps

```
# Plot oldpeak and trestbps
plot_two <- ggplot(heart_disease, aes(x = oldpeak, y = trestbps,
                                      color = as.factor(hc_clust))) +
  geom_point()
plot_two
```



So we can visualize how the clustering works in hierarchical

CONCLUSION

Now that we've tried multiple clustering algorithms, it is necessary to determine if we think any of them will work for clustering our patients

- For the k-means algorithm, similar clusters must be produced for each iteration of the algorithm to make sure that the algorithm clusters the signal, not the noise.
- For the sake of the doctors, we also want to have multiple patients in each group so they can compare treatments.
- Remember that it is important the k-mean algorithm seems stable when running multiple iterations. This means that we would see similar groups of patients showing up in the plots from the different iterations of the algorithm.
- For the hierarchical clustering, we need a method that puts a balanced number of patients in each group.

```
#Add TRUE if the algorithm shows promise, add FALSE if it does not
explore_kmeans <- FALSE
explore_hierarch_complete <- TRUE
explore_hierarch_single <- FALSE
```

REFERENCES

- https://archive.ics.uci.edu/ml/datasets/heart+Disease
  (Last Accessed 8th Dec 2020)
- https://rstudio.com
  (Last Accessed 8th Dec 2020)
- https://learn.datacamp.com/projects/552
  (Last Accessed 8th Dec 2020)

- **Data Science With R course** , Datacamp , (Last Accessed 8th Dec 2020)
-
- **Machine Learning Nanodegree Program** , Udacity Last Accessed 8th Dec 2020)
-
- **Data Science for all** channel , Youtube (Last Accessed 8th Dec 2020)

NEXT PAGE PROJECT 2

INTRODUCTION

1.1 Basic Introduction

Worldwide, road traffic injuries claim more than 1.2 million lives each year and are the leading cause of death among young people aged between 15 and 29 years old. Out of the above-mentioned statistics the USA contributes around 12% of road traffic mortality world-wide. These deaths have a huge impact on health and development, and represent an economic burden in every country. It's estimated that road traffic injuries cost governments approximately 3% of GDP, and up to 5% in low- and middle-income countries. The rise in global road traffic deaths has been largely driven by the escalating death toll on roads in low- and middle-income countries, particularly in emerging economies. In many middle-income countries, the risk of suffering road traffic injuries depends on various social determinants, such as the use of alcohol while driving, excess speed, traffic flow, and urban and infrastructural developments. Even though all the countries worldwide have been incorporating and implementing various interventions to reduce road traffic injuries, the pace of change has been slow. In addition to road deaths, over 50 million people sustain non-fatal injuries each year as a result of road traffic crashes. Road traffic deaths and injuries also can lead to indirect and direct consequences, such as emotional distress and loss of family income; moreover, an increase in obesity due a lack of proper road infrastructure (unsafe spaces for walking or cycling) can help increase the level of obesity in a country. So, in order to device new strategies to overcome problems related to road mortality we must first have a detailed insight to the core problems like fatality caused due to excessive speeding, alcohol consumption and amateur driving skills and then we must focus on to group of states in the US facing similar kind of traits or patterns. Such kind of grouping will help us to prioritize area of focus among 51 states of the USA and all these can be achieved using Data Mining, Data Wrangling, Data Visualization, Dimensionality Reduction and Unsupervised Clustering

1.2 Scope of Work

In the United States, the PAHO (Pan American Health Organization) is an international public health agency working to improve health and living standards of the people of United States. As per them:

- Road traffic injuries in the Region of the America kill some 154,089 people each year, representing 12% of the road traffic mortalities world-wide.
- The road traffic death rate for the Region as a whole is 15.9 per 100,000 population, lower than the global rate of 17.4. This regional rate masks wide variations from country to country, however—national rates range from a low of 6.0 in Canada to a high of 29.3 in the Dominican Republic.
- A statistical pie distribution shows that in these death rates car occupants, motorcyclists, and pedestrians accounts for 34%, 23% and 22% respectively, while cyclists represent 3% and remaining 18% of the deaths are from other categories which are not specified.
- Considering number of million miles driven with respect to number of driver fatal rate annually sums up to 2.95 million miles driven with respect to 100K number of driver fatal rate. These statistics demonstrates that road traffic death rate for the region as a whole might be 29 drivers per 100K population.

So, taking all these into consideration, we will design a clustering-based model along with Dimensionality Reduction which will group the states with similar kind of traits or patterns in a cluster in an accurate manner. And these clusters will eventually show the states or provinces with high priority focus among 51 states.

REQUIREMENT ANALYSIS

2.1 Functional Requirements

Providing the details of the project, all requirements and specifications will be made. Some of the functionalities we are going to use in our project are the basic principles of Data science along with Python3 Programming and we are going to implement it on software called as Jupyter Notebook. Unsupervised Clustering of data grouping techniques from Machine Learning and Data Visualizations using matplotlib. Most of the important data science related libraries for Python have been implemented in our code in the best possible way to enhance the quality of our project. All the work has been done in the Jupyter Notebook environment of WinPy using base language as Python3. This provides a better work environments and faster run time for Data Science related work as compared to other IDEs' of Python available out there.

2.2Dataset

We are going to use two datasets first road-accidents.csv and second miles-driven.csv. The first dataset road-accidents.csv consists of 51 records/instances (rows) and 5 attributes/variables (columns) and the first dataset road-accidents.csv consists of 51 records/instances (rows) and 2 attributes/variables (columns). To download the data, visit here.

First dataset road-accidents.csv uses 5 attributes used:

1. **state**: Consists of all 51 states of USA and it is stored as object type.
2. **drvr_fatl_col_bmiles**: Driver fatal column per billion miles consists of driver death rates per billion miles and it is stored as float64 type.
3. **perc_fatl_speed**: Percentage fatal Speed consists of driver death rates due to excessive speeding and it is stored as int64 type.
4. **perc_fatl_alcohol**: Percentage fatal Alcohol consists of driver death rates due to alcohol consumption and it is stored as int64 type.
5. **perc_fatl_1st_time**: Percentage fatal 1st Time consists of driver death rates due to first time driving on road or amateur driving skill of the driver and it is stored as int64 type.

Second dataset miles-driven.csv uses 2 attributes used:

1. **state**: Consists of all 51 states of USA and it is stored as object type.
2. **million_miles_annually**: Million Miles Driven to that particular state annually and it is stored as int64 type.

BACKGROUND ANALYSIS

3.1 A Look at Data

First dataset road-accidents.csv consists of 51 records/instances (rows) or 5 attributes/variables (columns):

Out[1]:

	state	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
0	Alabama	18.8	39	30	80
1	Alaska	18.1	41	25	94
2	Arizona	18.6	35	28	96
3	Arkansas	22.4	18	26	95
4	California	12.0	35	28	89
5	Colorado	13.6	37	28	95
6	Connecticut	10.8	46	36	82
7	Delaware	16.2	38	30	99
8	District of Columbia	5.9	34	27	100
9	Florida	17.9	21	29	94
10	Georgia	15.6	19	25	93
11	Hawaii	17.5	54	41	87
12	Idaho	15.3	36	29	98
13	Illinois	12.8	36	34	96
14	Indiana	14.5	25	29	95
15	Iowa	15.7	17	25	87
16	Kansas	17.8	27	24	85
17	Kentucky	21.4	19	23	76
18	Louisiana	20.5	35	33	98
19	Maine	15.1	38	30	84
20	Maryland	12.5	34	32	99

21	Massachusetts	8.2	23	35	80
22	Michigan	14.1	24	28	77
23	Minnesota	9.6	23	29	88
24	Mississippi	17.6	15	31	100
25	Missouri	16.1	43	34	84
26	Montana	21.4	39	44	85
27	Nebraska	14.9	13	35	90
28	Nevada	14.7	37	32	99
29	New Hampshire	11.6	35	30	83
30	New Jersey	11.2	16	28	78
31	New Mexico	18.4	19	27	98
32	New York	12.3	32	29	80
33	North Carolina	16.8	39	31	81
34	North Dakota	23.9	23	42	86
35	Ohio	14.1	28	34	82
36	Oklahoma	19.9	32	29	94
37	Oregon	12.8	33	26	90
38	Pennsylvania	18.2	50	31	88
39	Rhode Island	11.1	34	38	79
40	South Carolina	23.9	38	41	81
41	South Dakota	19.4	31	33	86
42	Tennessee	19.5	21	29	81
43	Texas	19.4	40	38	87
44	Utah	11.3	43	16	96
45	Vermont	13.6	30	30	95
46	Virginia	12.7	19	27	88
47	Washington	10.6	42	33	86
48	West Virginia	23.8	34	28	87
49	Wisconsin	13.8	36	33	84
50	Wyoming	17.4	42	32	90

First dataset road-accidents.csv consists of 51 records/instances (rows) or 5 attributes/variables (columns):

Out[2]:

	state	million_miles_annually
0	Alabama	64914
1	Alaska	4593
2	Arizona	59575
3	Arkansas	32953
4	California	320784
5	Colorado	46606
6	Connecticut	31197
7	Delaware	9028
8	District of Columbia	3568
9	Florida	191855
10	Georgia	108454
11	Hawaii	10066
12	Idaho	15937
13	Illinois	103234
14	Indiana	76485
15	Iowa	31274
16	Kansas	30021
17	Kentucky	48061
18	Louisiana	46513
19	Maine	14248
20	Maryland	56221
21	Massachusetts	54792
22	Michigan	94754
23	Minnesota	56685
24	Mississippi	38851
25	Missouri	68789
26	Montana	11660
27	Nebraska	19093
28	Nevada	24189
29	New Hampshire	12720
30	New Jersey	73094
31	New Mexico	25650
32	New York	127726
33	North Carolina	103772
34	North Dakota	9131
35	Ohio	111990
36	Oklahoma	47464
37	Oregon	33373
38	Pennsylvania	99204
39	Rhode Island	7901
40	South Carolina	48730

41	South Dakota	9002
42	Tennessee	70751
43	Texas	237440
44	Utah	26222
45	Vermont	7141
46	Virginia	80974
47	Washington	56955
48	West Virginia	18963
49	Wisconsin	58554
50	Wyoming	9245

3.2 Methodology

- **Data storage locations**: Both the datasets road-accidents.csv and miles-driven.csv are stored in the local drive of our machines and will be used as the bas for our whole project. Since both the datasets are originally collected by the National Highway Traffic Safety Administration and the National Association of Insurance Commissioners they consist of real-time values and they are huge in terms of values per cells. However, it is said that the bigger the data the excessive amount of training is required which leads to more accurate results.

- **Data ingestion**: Importing our set of data from the stored location to our main code written in jupyter notebook which will further be checked for Nan values and outliers and then it will get clean and processed. This includes steps like understanding our data, collecting it, describing the data, exploring and verifying the quality of it.

- **Data Preparation**: Consists of steps like selecting the data, constructing the Data Frame Object, Data Cleaning, integrating and formatting the data using technique like Normalization, Transformation, Scaling, Fitting, Validation and Featurization

- **Model Building & Training:** In this stage we will build and train following models:

  - **Multivariate Linear Regression (MLR)** which will help us to compare one target (like: drvr_fatl_col_bmiles) with multiple features (like: perc_fatl_speed, perc_fatl_alcohol, perc_fatl_1st_time), where, it estimates a single regression model with more than one predictor variable.
  - **Principal Component Analysis (PCA)** will help us to use results generated using **Multivariate Linear Regression (MLR)** and then we can implement Dimensionality Reduction so that we can select important feature as a component out of these features (perc_fatl_speed, perc_fatl_alcohol, perc_fatl_1st_time) and then do component analysis by pairing and fitting it with our target (drvr_fatl_col_bmiles).
  - **Data Visualization using Scree Plot** this is not any kind of model but it is a helpful plotting function to determine the number of centroid points for **Unsupervised K-Means Clustering**. **Scree Plot** is often called as **Elbow Plot** where the points residing in the elbow always give the accurate number of data points which can be used to define centroid points initially to generate clusters later in **Unsupervised K-Means Clustering** algorithm.
  - **K-Means Clustering** is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes. A cluster refers to a collection of data points aggregated together because of certain similarities. Every data point is allocated to each of the nearest cluster depending on the centroid point which is initially defined to generate the clusters.

- **Model Deployment:** After training all the models we have to merge the two Data Frame Objects imported from datasets road-accidents.csv and miles-driven.csv in such a way that which state of the US lies under which cluster.

States corresponding to highest priority cluster are the actual area of focus to work on to reduce the road traffic mortality.

3.3 How to overcome the problem?

The very important problem which can be taken into consideration for reduction of traffic mortality is to implement costly nation-wide plan without identifying the are of focus among 51 states which can be achieved using clustering or grouping of data points. Before implementation if we figure out the clusters or groups then we are good to imply certain rules nation-wide to reduce traffic mortality. This project helps us to investigate the same and group states showing similar traits or patterns into cluster and then select those states which lies under highest priority cluster to implement action rules. To accomplish this in this project we are using Data Wrangling, Data Visualization, and Machine Learning algorithms (like, **Multivariate Linear Regression (MLR), Principal Component Analysis (PCA)** and **Unsupervised Learning** such as **K-Means Clustering**). The datasets for this project can be found here.

APPROACH

This is just a short overview of what strategies have been used to solve the problem. The detailed description will be in the code and explanation section.

4.1 Importing libraries

Firstly, we imported some of the basic important libraries used in R

- os - helps python shell to interact with operating system
- pandas - data manipulation and analysis in Data Frame and Time Series Object
- numpy - numerical python library used to work with arrays and matrices
- sklearn - machine learning algorithms like supervised and unsupervised learning
- matplotlib - data visualization for basic plotting
- seaborn - data visualization for complex plotting

4.2 Loading into Python3

Some pre-processing steps included downloading the data and loading it., pre-processed to make it smaller and faster to load. All the CSV (comma separated) files were loaded into R using the pandas.read_csv() function to load the dataset into data frame object.

4.3 Tidying the Data

Now, to check the basic statistics of the data and separate the unwanted data from the important data we use methods like info() and describe() and all Nan values if present can be identified using dataframeobject.isna().sum() and later can be removed to clean the data. Since the dataset is entirely numerical therefore it has to be Scaled and Fitted before training any model using StandardScalar() module of sklearn library residing inside preprocessing module (eg: from sklearn.preprocessing import StandardScalar()) to scale the data frame object and .fit() function is used to adjust weight according to the data values to obtain better accuracy. **Note:** Scaling and Fitting is always performed before training the models.

4.4 Data Visualization

Here we are doing the both basic plotting and complex plotting to demonstrate Data Visualization using matplotlib and seaborn libraries respectively.

4.6 Correlation

We can check the correlation analysis from different variables, for this we have to use the dataset with numerical values. We will print the output of correlation which will represent how each attribute/variable (column) is related to each other.

4.7 Using various Machine Learning Algorithms

- **Multivariate Linear Regression (MLR)** – to compare one target with multiple features, where, it estimates a single regression model with more than one predictor variable.
- **Principal Component Analysis (PCA)** – to use results generated using **Multivariate Linear Regression (MLR)** and then we can implement Dimensionality Reduction so that we can select important feature as a component out of the above specified features and then do component analysis by pairing and fitting it with our target.
- **Data Visualization using Scree Plot** – this is not any kind of model but it is a helpful plotting function to determine the number of centroid points for **Unsupervised K-Means Clustering**. **Scree Plot** is often called as

Elbow Plot where the points residing in the elbow always give the accurate number of data points which can be used to define centroid points initially to generate clusters later in **Unsupervised K-Means Clustering** algorithm.

- **K-Means Clustering** – is one of the simplest and popular unsupervised machine learning algorithms to bring various data points under appropriate clusters. A cluster refers to a collection of data points aggregated together because of certain similarities. Every data point is allocated to each of the nearest cluster depending on the centroid point which is initially defined to generate the clusters.

ANALYSIS OF CODE

In [1]:
```
import os

current_dir = os.getcwd()
print("Present Working Directory:\n", current_dir, sep="")
```

```
Present Working Directory:
C:\myProject\WinPY\WPy64-3830\notebooks
```

In [2]:
```
# Change the location of current directory to the required locations where datasets resides
os.chdir(r"D:/NYIT (Fall2020-ACADEMICS)/INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROFF Helen Gu]/Final Projects/PROJECT1 PYTHON/datasets")
current_dir = os.getcwd()
print("Changed Directory:\n", current_dir, sep="")

# List all files in the current directory
ls = os.listdir()
print("\nList Of Files:")
for element in range(len(ls)):
    print(ls[element], sep="")

# View the first 20 lines of datasets/miles-driven.csv
with open("miles-driven.csv") as myfirst_dataset:
    head = [next(myfirst_dataset) for x in range(20)]
print("\nFirst 20 Lines of Dataset miles-driven.csv:\n", str(head).strip("[]"), sep="")

# View the first 20 lines of datasets/road-accidents.csv
with open("road-accidents.csv") as mysecond_dataset:
    head = [next(mysecond_dataset) for x in range(20)]
print("\nFirst 20 Lines of Dataset road-accidents.csv:\n", str(head).strip("[]"), sep="")
```

```
Changed Directory:
D:\NYIT (Fall2020-ACADEMICS)\INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROFF Helen Gu]\Final Projects
\PROJECT1 PYTHON\datasets

List Of Files:
DataSet
Link.txtmiles-
driven.csv
road-accidents.csv

First 20 Lines of Dataset miles-driven.csv:
'state|million_miles_annually\n', 'Alabama|64914\n', 'Alaska|4593\n', 'Arizona|59575\n', 'Arkansas|32953\n', 'California|320784\n', 'Colorado|46606\n', 'Connecticut|31197\n', 'Delaware|9028\n', 'District of Columbia|3568\n', 'Florida|191855\n', 'Georgia|108454\n', 'Hawaii|10066\n', 'Idaho|15937\n', 'Illinois|103234\n', 'Indiana|76485\n', 'Iowa|31274\n', 'Kansas|30021\n', 'Kentucky|48061\n', 'Louisiana|46513\n'

First 20 Lines of Dataset road-accidents.csv:
'##### LICENSE #####\n', '# This data set is modified from the original at fivethirtyeight (https://github.com/fivethirtyeight/data/tree/master/bad-drivers)\n', '# and it is released under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/)\n', '##### COLUMN ABBREVIATIONS #####\n', '# drvr_fatl_col_bmiles = Number of drivers involved in fatal collisions per billion miles (2011)\n', '# perc_fatl_speed = Percentage Of Drivers Involved In Fatal Collisions Who Were Speeding (2009)\n', '# perc_fatl_alcohol = Percentage Of Drivers Involved In Fatal Collisions Who Were Alcohol-Impaired (2011)\n', '# perc_fatl_1st_time = Percentage Of Drivers Involved InFatal Collisions Who Had Not Been Involved In Any Previous Accidents (2011)\n', '##### DATA BEGIN #####\n', 's
tate|drvr_fatl_col_bmiles|perc_fatl_speed|perc_fatl_alcohol|perc_fatl_1st_time\n', 'Alabama|18.8|39|30|80\n', 'Alaska|18.1|41|25|94\n', 'Arizona|18.6|35|28|96\n', 'Arkansas|22.4|18|26|95\n', 'California|12|35|28|89\n', 'Colorado|13.6|37|28|95\n', 'Connecticut|10.8|46|36|82\n', 'Delaware|16.2|38|30|99\n', 'District of Columbia|5.9|34|27|100\n', 'Florida|17.9|21|29|94\n'
```

```
# Import the `pandas` module as "pd"
import pandas as pd

# Read in `road-accidents.csv`
car_acc = pd.read_csv("D:/NYIT (Fall2020-ACADEMICS)/INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROFF Hel
en Gu]/Final Projects/PROJECT1 PYTHON/datasets/road-accidents.csv", comment='#', sep='|')

# Save the number of rows columns as a tuple
rows_and_cols = car_acc.shape
print("Brief Description Of Dataset road-accidents.csv Stored In DataFrame Object car_acc:\n")
print('There are {} rows and {} columns.\n'.format(rows_and_cols[0], rows_and_cols[1]))

# Generate an overview of the DataFrame
car_acc_information = car_acc.info()
print(car_acc_information)

# Check whether there are Nan values or not
print("\nBrief Description Whether DataFrame Object car_acc Consists Any NaN Values:\n", car_acc.isna().sum(),
sep="")

# Display the first five rows of the DataFrame
print("\nFollowing Is The Generic Overview Of DataFrame Object car_acc:\n")
car_acc.head()
```

```
Brief Description Of Dataset road-accidents.csv Stored In DataFrame Object

car_acc:There are 51 rows and 5 columns.

<class
'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   state                 51 non-null     object
 1   drvr_fatl_col_bmiles  51 non-null     float64
 2   perc_fatl_speed       51 non-null     int64
 3   perc_fatl_alcohol     51 non-null     int64
 4   perc_fatl_1st_time    51 non-null     int64
dtypes: float64(1), int64(3), object(1)
memory usage: 2.1+ KB
None

Brief Description Whether DataFrame Object car_acc Consists Any NaN
Values:state                0
drvr_fatl_col_bmiles    0
perc_fatl_speed         0
perc_fatl_alcohol       0
perc_fatl_1st_time      0
dtype: int64
Following Is The Generic Overview Of DataFrame Object car_acc:
```

Out[3]:

	state	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
0	Alabama	18.8	39	30	80
1	Alaska	18.1	41	25	94
2	Arizona	18.6	35	28	96
3	Arkansas	22.4	18	26	95
4	California	12.0	35	28	89

```
# Compute the summary statistics of all columns in the `car_acc` DataFrame
sum_stat_car = car_acc.describe()
print("Summary Statistics Of DataFrame Object car_acc:")
sum_stat_car
```

Summary Statistics Of DataFrame Object car_acc:

Out[4]:

	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
count	51.000000	51.000000	51.000000	51.00000
mean	15.790196	31.725490	30.686275	88.72549
std	4.122002	9.633438	5.132213	6.96011
min	5.900000	13.000000	16.000000	76.00000
25%	12.750000	23.000000	28.000000	83.50000
50%	15.600000	34.000000	30.000000	88.00000
75%	18.500000	38.000000	33.000000	95.00000
max	23.900000	54.000000	44.000000	100.00000

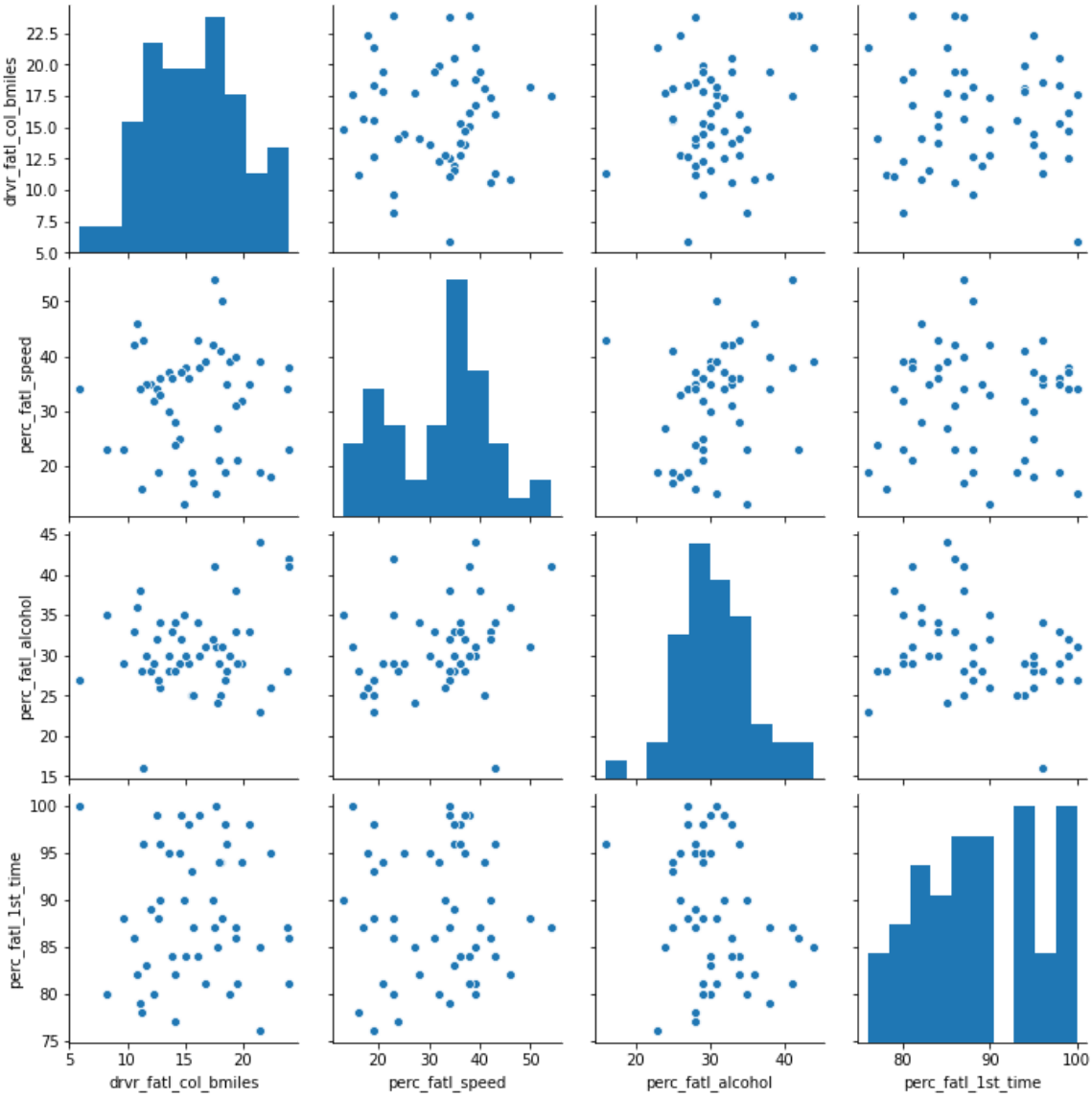
In [5]:

```
# import seaborn and make plots appear inline
import seaborn as sns
%matplotlib inline

# Create a pairwise scatter plot to explore the data
print("Data Visualization And Exploration:\n\n", sns.pairplot(car_acc), sep="")
```

Data Visualization And Exploration:

<seaborn.axisgrid.PairGrid object at 0x00000258A9616CD0>


drvr_fatl_col_bmiles
perc_fatl_speed
perc_fatl_alcohol
perc_fatl_1st_time
drvr_fatl_col_bmiles
perc_fatl_speed
perc_fatl_alcohol
perc_fatl_1st_time

```
# Compute the correlation coefficent for all column pairs
print("Pearson Correlation Coefficient Matrix:\n")
corr_columns = car_acc.corr()
corr_columns
```

Pearson Correlation Coefficient Matrix:

Out[6]:

	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
drvr_fatl_col_bmiles	1.000000	-0.029080	0.199426	-0.017942
perc_fatl_speed	-0.029080	1.000000	0.286244	0.014066
perc_fatl_alcohol	0.199426	0.286244	1.000000	-0.245455
perc_fatl_1st_time	-0.017942	0.014066	-0.245455	1.000000

In [7]:

```
# Import the linear model function from sklearn
import numpy as np
from sklearn import linear_model

# Create the features and target DataFrames
features = car_acc[['perc_fatl_speed', 'perc_fatl_alcohol', 'perc_fatl_1st_time']]
target = car_acc['drvr_fatl_col_bmiles']

# Create a linear regression object reg =
linear_model.LinearRegression()print(type(reg))

# Fit a multivariate linear regression model
reg.fit(features, target)

# Retrieve the regression coefficients
fit_coef = reg.coef_
print(type(fit_coef))

fit_coef
```

=

<class 'sklearn.linear_model._base.LinearRegression'>
<class 'numpy.ndarray'>

Out[7]: array([-0.04180041, 0.19086404, 0.02473301])

```
# Standardize and center the feature columns
from sklearn.preprocessing import StandardScalerscaler =
StandardScaler()
features_scaled = scaler.fit_transform(features)

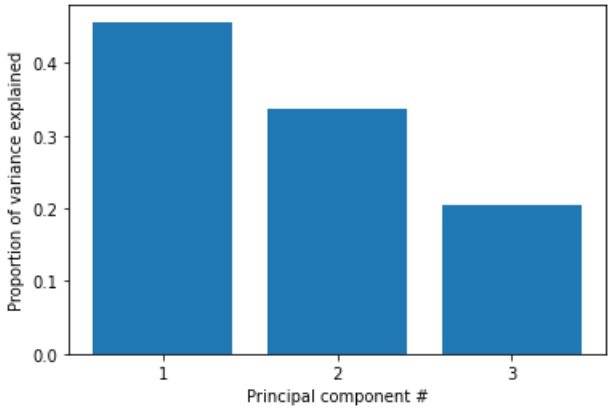
# Import the PCA class from sklearn from
sklearn.decomposition import PCApca = PCA()

# Fit the standardized data to the pca
pca.fit(features_scaled)

# Plot the proportion of variance explained on the y-axis of the bar plot
import matplotlib.pyplot as plt
plt.bar(range(1, pca.n_components_ + 1),  pca.explained_variance_ratio_)
plt.xlabel('Principal component #')
plt.ylabel('Proportion of variance explained')
plt.xticks([1, 2, 3])

# Compute the cumulative proportion of variance explained by the first two principal components
two_first_comp_var_exp = pca.explained_variance_ratio_.cumsum()[1]
print("The Cumulative Variance Of The First Two Principal Componenets Is {}".format(round(two_first_comp_var_ex
p, 5)))
```

The Cumulative Variance Of The First Two Principal Componenets Is 0.7947



In [9]:

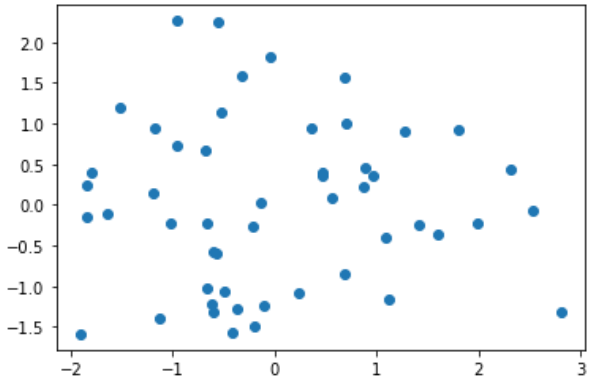
```
# Transform the scaled features using two principal components
pca = PCA(n_components=2)
p_comps = pca.fit_transform(features_scaled)

# Extract the first and second component to use for the scatter plot
p_comp1 = p_comps[:, 0]
p_comp2 = p_comps[:, 1]

# Plot the first two principal components in a scatter plot
print("Scatter Plot Of First Two Principal Component:\n")
plt.scatter(p_comp1, p_comp2)
```

Scatter Plot Of First Two Principal Component:

Out[9]: <matplotlib.collections.PathCollection at 0x258ae817610>



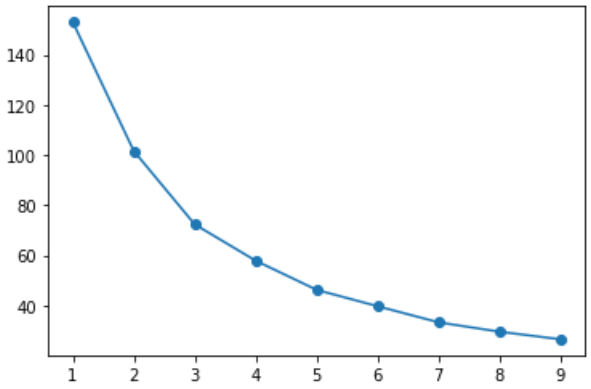
```
# Import KMeans from sklearn
from sklearn.cluster import KMeans

# A loop will be used to plot the explantory power for up to 10 KMeans clusters
ks = range(1, 10)
inertias = []
for k in ks:
    # Initialize the KMeans object using the current number of clusers (k)
    km = KMeans(n_clusters=k, random_state=8)
    # Fit the scaled scaled features to the KMeans object
    km.fit(features_scaled)
    # Append the inertia for `km` to the list of inertias
    inertias.append(km.inertia_)

# Plot the results in a line plot
print("Scree Plot Or Line Plot Using KMeans Along With Principal Components:\n")
plt.plot(ks, inertias, marker='o')
```

Scree Plot Or Line Plot Using KMeans Along With Principal Components:

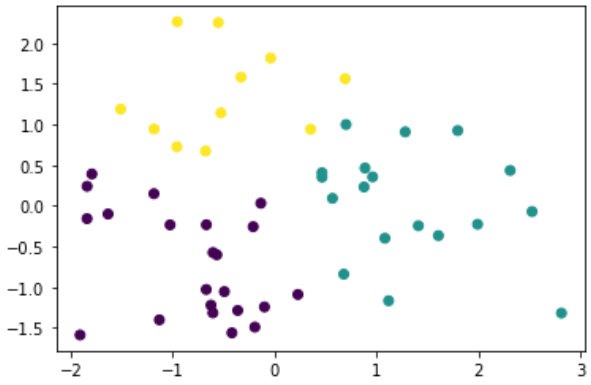
Out[10]: [<matplotlib.lines.Line2D at 0x258aeee9e80>]



In [11]:
```
# Create a KMeans object with 3 clusters km =
KMeans(n_clusters=3, random_state=8)# Fit the data to
the `km` object
km.fit(features_scaled)
# Create a scatter plot of the first two principal components# and color it
according to the KMeans cluster assignment
print("Visualize Clusters Using Scatter Plot, KMeans And PCA:\n")
plt.scatter(p_comps[:, 0], p_comps[:, 1], c=km.labels_)
```

Visualize Clusters Using Scatter Plot, KMeans And PCA:

Out[11]: <matplotlib.collections.PathCollection at 0x258ae029520>

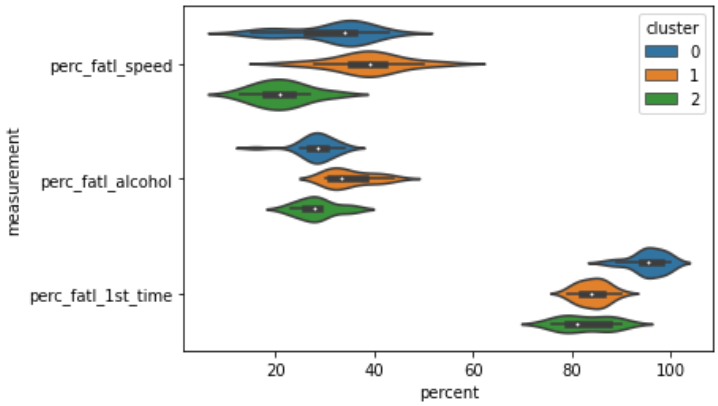


```
# Create a new column with the labels from the KMeans clustering
car_acc['cluster'] = km.labels_

# Reshape the DataFrame to the long format
melt_car = pd.melt(car_acc, id_vars='cluster', var_name='measurement', value_name='percent',
                   value_vars=['perc_fatl_speed', 'perc_fatl_alcohol', 'perc_fatl_1st_time'])

# Create a violin plot splitting and coloring the results according to the km-clusters
sns.violinplot(y='measurement', x='percent', data=melt_car, hue='cluster')
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x258adf4b310>



```
# Read in the `miles-drives.csv`
miles_driven = pd.read_csv("D:/NYIT (Fall2020-ACADEMICS)/INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROF
F Helen Gu]/Final Projects/PROJECT1 PYTHON/datasets/miles-driven.csv", sep='|')

# Save the number of rows columns as a tuple
rows_and_cols = miles_driven.shape
print("Brief Description Of Dataset miles-driven.csv Stored In DataFrame Object miles_driven:\n")
print('There are {} rows and {} columns.\n'.format(rows_and_cols[0], rows_and_cols[1]))

# Generate an overview of the DataFrame
miles_driven_information = miles_driven.info()
print(miles_driven_information)

# Check whether there are Nan values or not
print("\nBrief Description Whether DataFrame Object miles_driven Consists Any NaN Values:\n", miles_driven.isna
().sum(), sep="")

# Display the first five rows of the DataFrame
print("\nFollowing Is The Generic Overview Of DataFrame Object miles_driven:\n")
miles_driven.head()
```

Brief Description Of Dataset miles-driven.csv Stored In DataFrame Object

miles_driven:There are 51 rows and 2 columns.

<class
'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 2 columns):

#	Column	Non-Null Count	Dtype
0	state	51 non-null	object
1	million_miles_annually	51 non-null	

int64dtypes: int64(1), object(1)
memory usage: 944.0+
bytesNone

Brief Description Whether DataFrame Object miles_driven Consists Any NaN
Values:state 0
million_miles_annually 0
dtype: int64

Following Is The Generic Overview Of DataFrame Object miles_driven:

Out[13]:

	state	million_miles_annually
0	Alabama	64914
1	Alaska	4593
2	Arizona	59575
3	Arkansas	32953
4	California	320784

```
# Update the DataFrame `miles_driven` by converting `million_miles_annually` to `billion_miles_annually`
miles_driven['million_miles_annually'] = (miles_driven['million_miles_annually'] / 1000)
miles_driven.rename(columns={'million_miles_annually':'billion_miles_annually'}, inplace=True)
miles_driven.head()
```

Out[14]:

	state	billion_miles_annually
0	Alabama	64.914
1	Alaska	4.593
2	Arizona	59.575
3	Arkansas	32.953
4	California	320.784

In [15]:
```
# Merge the `car_acc` DataFrame with the `miles_driven` DataFrame
car_acc_miles = car_acc.merge(miles_driven, on='state')
print("After Merging We Get DataFrame Object car_acc_miles As:\n")
car_acc_miles.head()
```

After Merging We Get DataFrame Object car_acc_miles As:

Out[15]:

	state	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time	cluster	billion_miles_annually
0	Alabama	18.8	39	30	80	1	64.914
1	Alaska	18.1	41	25	94	0	4.593
2	Arizona	18.6	35	28	96	0	59.575
3	Arkansas	22.4	18	26	95	0	32.953
4	California	12.0	35	28	89	0	320.784

In [16]:
```
# Create a new column for the number of drivers involved in fatal accidents
car_acc_miles['num_drvr_fatl_col'] = car_acc_miles['drvr_fatl_col_bmiles'] * car_acc_miles['billion_miles_annua
lly']
print("\nFollowing Is The Generic Overview Of DataFrame Object car_acc_miles After Adding New Column:\n")
car_acc_miles.head()
```

Following Is The Generic Overview Of DataFrame Object car_acc_miles After Adding New Column:

Out[16]:

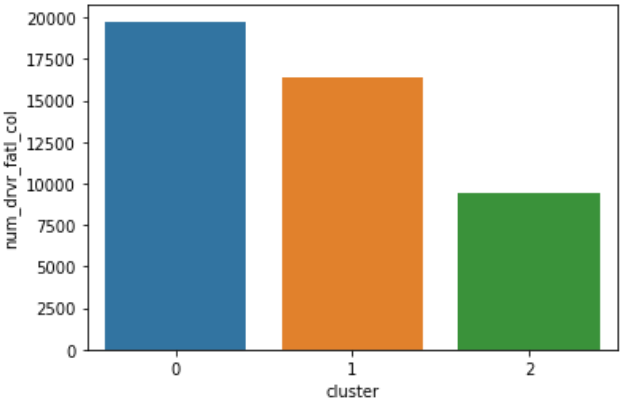
	state	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time	cluster	billion_miles_annually	num_drvr_fatl_col
0	Alabama	18.8	39	30	80	1	64.914	1220.3832
1	Alaska	18.1	41	25	94	0	4.593	83.1333
2	Arizona	18.6	35	28	96	0	59.575	1108.0950
3	Arkansas	22.4	18	26	95	0	32.953	738.1472
4	California	12.0	35	28	89	0	320.784	3849.4080

```
# Create a barplot of the total number of accidents per cluster
sns.barplot(x='cluster', y='num_drvr_fatl_col', data=car_acc_miles, estimator=sum, ci=None)

# Calculate the number of states in each cluster and their 'num_drvr_fatl_col' mean and sum.
count_mean_sum = car_acc_miles.groupby('cluster')['num_drvr_fatl_col'].agg(['count', 'mean', 'sum'])
count_mean_sum
```

Out[17]:

cluster	count	mean	sum
0	22	898.378595	19764.3291
1	18	911.406439	16405.3159
2	11	860.505945	9465.5654



‘

```
# Directing corresponding states to cluster 0
print("Following Are The Respective States Corresponding To Cluster 0:\n")
for i in range(len(car_acc_miles)):
    if car_acc_miles.loc[i].cluster == 0:
        print(car_acc_miles.loc[i].state)
```

Following Are The Respective States Corresponding To Cluster 0:

Alaska
Arizona
Arkansas
California
Colorado
Delaware
District of Columbia
Florida
Georgia
Idaho
Illinois
Indiana
Louisiana
Maryland
Mississippi
Nevada
New Mexico
Oklahoma
Oregon
Utah
Vermont
West Virgin

EXPLANATION OF CODE STEP BY STEP

In this section the code is explained line by line and why that method was applied and its description in detail.

cell 1. Importing OS Library for File System Structure Manipulation

In cell 1, we have imported os library using “import os” to show current directory after opening the Jupyter Notebook. To get the current directory we used “os.getcwd().

In [1]:
```
import os

current_dir = os.getcwd()
print("Present Working Directory:\n", current_dir, sep="")
```

```
Present Working Directory:
C:\myProject\WinPY\WPy64-3830\notebooks
```

cell 2. File Manipulation In Raw File Format

In cell 2, we have changed the directory to the directory where the actual datasets are residing using “os.chdir()” and then we used “os.listdir()” to see all the files residing in that directory. After we got that our datasets are in proper directory then we used file open code line using with “open()” to open the file and see the first 20 lines of the file to achieve this we “for loop”. Therefore, in the last two output lines of this cell we can see the first 20 lines of both the datasets in raw format.

In [2]:
```
# Change the location of current directory to the required locations where datasets resides
os.chdir(r"D:/NYIT (Fall2020-ACADEMICS)/INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROFF Helen Gu]/Final
Projects/PROJECT1 PYTHON/datasets")
current_dir = os.getcwd()
print("Changed Directory:\n", current_dir, sep="")

# List all files in the current directory
ls = os.listdir()
print("\nList Of Files:")
for element in range(len(ls)):
    print(ls[element], sep="")

# View the first 20 lines of datasets/miles-driven.csv
with open("miles-driven.csv") as myfirst_dataset:
    head = [next(myfirst_dataset) for x in range(20)]
print("\nFirst 20 Lines of Dataset miles-driven.csv:\n", str(head).strip("[]"), sep="")

# View the first 20 lines of datasets/road-accidents.csv
with open("road-accidents.csv") as mysecond_dataset:
    head = [next(mysecond_dataset) for x in range(20)]
print("\nFirst 20 Lines of Dataset road-accidents.csv:\n", str(head).strip("[]"), sep="")
```

```
Changed Directory:
D:\NYIT (Fall2020-ACADEMICS)\INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROFF Helen Gu]\Final Projects
\PROJECT1 PYTHON\datasets

List Of Files:
DataSet Link.txt
miles-driven.csv
road-accidents.csv

First 20 Lines of Dataset miles-driven.csv:
'state|million_miles_annually\n', 'Alabama|64914\n', 'Alaska|4593\n', 'Arizona|59575\n', 'Arkansas|32953\n',
'California|320784\n', 'Colorado|46606\n', 'Connecticut|31197\n', 'Delaware|9028\n', 'District of Columbia|356
8\n', 'Florida|191855\n', 'Georgia|108454\n', 'Hawaii|10066\n', 'Idaho|15937\n', 'Illinois|103234\n', 'Indiana
|76485\n', 'Iowa|31274\n', 'Kansas|30021\n', 'Kentucky|48061\n', 'Louisiana|46513\n'

First 20 Lines of Dataset road-accidents.csv:
'##### LICENSE #####\n', '# This data set is modified from the original at fivethirtyeight (https://github.co
m/fivethirtyeight/data/tree/master/bad-drivers)\n', '# and it is released under CC BY 4.0 (https://creativecom
mons.org/licenses/by/4.0/)\n', '##### COLUMN ABBREVIATIONS #####\n', '# drvr_fatl_col_bmiles = Number of drive
rs involved in fatal collisions per billion miles (2011)\n', '# perc_fatl_speed = Percentage Of Drivers Involv
ed In Fatal Collisions Who Were Speeding (2009)\n', '# perc_fatl_alcohol = Percentage Of Drivers Involved In F
atal Collisions Who Were Alcohol-Impaired (2011)\n', '# perc_fatl_1st_time = Percentage Of Drivers Involved In
Fatal Collisions Who Had Not Been Involved In Any Previous Accidents (2011)\n', '##### DATA BEGIN #####\n', 's
tate|drvr_fatl_col_bmiles|perc_fatl_speed|perc_fatl_alcohol|perc_fatl_1st_time\n', 'Alabama|18.8|39|30|80\n',
'Alaska|18.1|41|25|94\n', 'Arizona|18.6|35|28|96\n', 'Arkansas|22.4|18|26|95\n', 'California|12|35|28|89\n',
'Colorado|13.6|37|28|95\n', 'Connecticut|10.8|46|36|82\n', 'Delaware|16.2|38|30|99\n', 'District of Columbia|
5.9|34|27|100\n', 'Florida|17.9|21|29|94\n'
```

cell 3. Importing Dataset “road-accidents.csv” as a Data Frame Object

In cell 3, first we have imported pandas library using “import pandas as pd”. And then we have used “.info()” and “.describe()” to get some useful insights from the data frame object where “.describe()” gives statistical summary of the entire data frame object and “.info()” gives details about attributes/variables (columns) of the data frame object. Lastly we have used “.head()” to see the first 5 records of the data frame object.

In [3]:
```python
# Import the `pandas` module as "pd"
import pandas as pd

# Read in `road-accidents.csv`
car_acc = pd.read_csv("D:/NYIT (Fall2020-ACADEMICS)/INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROFF Hel
en Gu]/Final Projects/PROJECT1 PYTHON/datasets/road-accidents.csv", comment='#', sep='|')

# Save the number of rows columns as a tuple
rows_and_cols = car_acc.shape
print("Brief Description Of Dataset road-accidents.csv Stored In DataFrame Object car_acc:\n")
print('There are {} rows and {} columns.\n'.format(rows_and_cols[0], rows_and_cols[1]))

# Generate an overview of the DataFrame
car_acc_information = car_acc.info()
print(car_acc_information)

# Check whether there are Nan values or not
print("\nBrief Description Whether DataFrame Object car_acc Consists Any NaN Values:\n", car_acc.isna().sum(),
sep="")

# Display the first five rows of the DataFrame
print("\nFollowing Is The Generic Overview Of DataFrame Object car_acc:\n")
car_acc.head()
```

```
Brief Description Of Dataset road-accidents.csv Stored In DataFrame Object car_acc:

There are 51 rows and 5 columns.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   state                 51 non-null     object
 1   drvr_fatl_col_bmiles  51 non-null     float64
 2   perc_fatl_speed       51 non-null     int64
 3   perc_fatl_alcohol     51 non-null     int64
 4   perc_fatl_1st_time    51 non-null     int64
dtypes: float64(1), int64(3), object(1)
memory usage: 2.1+ KB
None

Brief Description Whether DataFrame Object car_acc Consists Any NaN Values:
state                   0
drvr_fatl_col_bmiles    0
perc_fatl_speed         0
perc_fatl_alcohol       0
perc_fatl_1st_time      0
dtype: int64

Following Is The Generic Overview Of DataFrame Object car_acc:
```

Out[3]:

	state	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
0	Alabama	18.8	39	30	80
1	Alaska	18.1	41	25	94
2	Arizona	18.6	35	28	96
3	Arkansas	22.4	18	26	95
4	California	12.0	35	28	89

cell 4. Shows Statistical Summary of Data Frame Object

In cell 4, as discussed in cell 3 it is just showing statistical summary of entire data frame object using ".describe()".

In [4]:
```
# Compute the summary statistics of all columns in the `car_acc` DataFrame
sum_stat_car = car_acc.describe()
print("Summary Statistics Of DataFrame Object car_acc:")
sum_stat_car
```

```
Summary Statistics Of DataFrame Object car_acc:
```

Out[4]:

	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
count	51.000000	51.000000	51.000000	51.00000
mean	15.790196	31.725490	30.686275	88.72549
std	4.122002	9.633438	5.132213	6.96011
min	5.900000	13.000000	16.000000	76.00000
25%	12.750000	23.000000	28.000000	83.50000
50%	15.600000	34.000000	30.000000	88.00000
75%	18.500000	38.000000	33.000000	95.00000
max	23.900000	54.000000	44.000000	100.00000

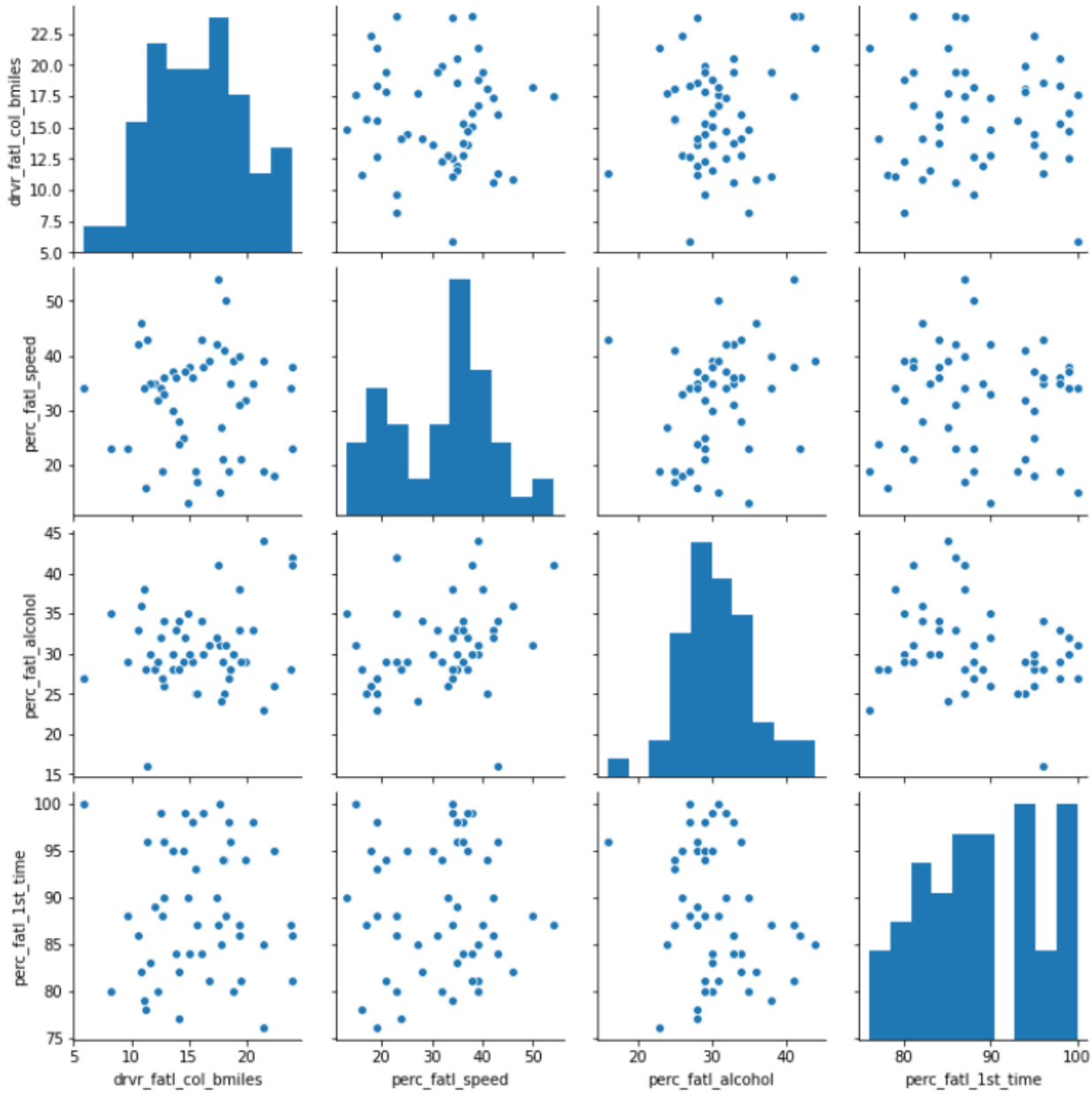
cell 5. Data Visualization Using Pair Plot Of Seaborn Library

In cell 5, we first imported the seaborn library using "import seaborn as sns" this library is used for Data Visualization for complex plots. After importing the library we used Pair Plot using "sns.pairplot(car_acc)" here we passed our data frame object "car_acc" inside "sns.pairplot()". Basically pair plot is used to pair all the attributes/values (columns) with each other and understand the relationship between them.

In [5]:
```
# import seaborn and make plots appear inline
import seaborn as sns
%matplotlib inline

# Create a pairwise scatter plot to explore the data
print("Data Visualization And Exploration:\n\n", sns.pairplot(car_acc), sep="")
```

Data Visualization And Exploration:

<seaborn.axisgrid.PairGrid object at 0x000002F78960CFD0>



cell 6. Correlating Attributes With Each Other

In cell 6, we are correlating all attributes/variables (columns) with each other to get insights of how each attribute is correlated with each other. To do this we have used "car_acc.corr()" function with our data frame object. So, after this we can realize that fatal rates due to alcohol consumption shows high correlation and in next step, we will see that the same attribute will show very low association. Hence, for till this step we can state that fatal rate due to alcohol consumption can be a factor to proceed with our analysis.

In [6]:
```
# Compute the correlation coefficent for all column pairs
print("Pearson Correlation Coefficient Matrix:\n")
corr_columns = car_acc.corr()
corr_columns
```

```
Pearson Correlation Coefficient Matrix:
```

Out[6]:

	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time
drvr_fatl_col_bmiles	1.000000	-0.029080	0.199426	-0.017942
perc_fatl_speed	-0.029080	1.000000	0.286244	0.014066
perc_fatl_alcohol	0.199426	0.286244	1.000000	-0.245455
perc_fatl_1st_time	-0.017942	0.014066	-0.245455	1.000000

cell 7. Implementing Multivariate Linear Regression (MLR)

From previous cell we came to know that fatal rate due to alcohol consumption can be a best approach to go forward with but that can not be only depending criteria. So to deduce more we used Multivariate Linear Regression (MLR) which generates single regression model using the target and pairing that target with possible predictor outcomes. To do this we have used sklearn library and linear_model module of that library using "from sklearn import linear_model". Then after this we passed all the features to feature variable and target to target variable. Then we build the regression model and before training the model we have to fit the weights of the feature and target using "reg.fit(features, target)"

In [7]:
```
# Import the linear model function from sklearn
import numpy as np
from sklearn import linear_model

# Create the features and target DataFrames
features = car_acc[['perc_fatl_speed', 'perc_fatl_alcohol', 'perc_fatl_1st_time']]
target = car_acc['drvr_fatl_col_bmiles']

# Create a linear regression object
reg = linear_model.LinearRegression()
print(type(reg))

# Fit a multivariate linear regression model
reg.fit(features, target)

# Retrieve the regression coefficients
fit_coef = reg.coef_
print(type(fit_coef))

fit_coef
```

```
<class 'sklearn.linear_model._base.LinearRegression'>
<class 'numpy.ndarray'>
```

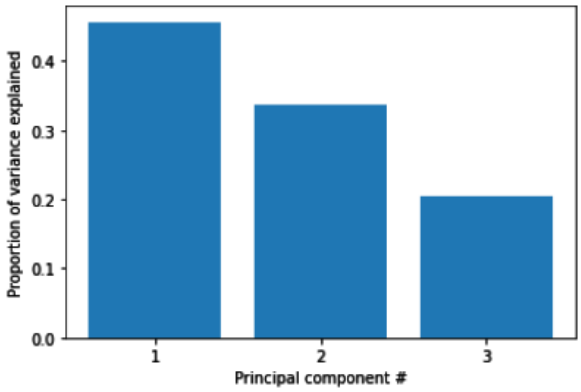
Out[7]:
```
array([-0.04180041,  0.19086404,  0.02473301])
```

cell 8. Implementing Principal Component Analysis (PCA)

After doing Multivariate Linear Regression (MLR) we have to use Principal Component Analysis (PCA) to group all the features together and then see which feature has maximum variance ratio with respect to that of mean value, this is how PCA works after we get the two highest features, we showed them using box plot. So, the lowest feature gets out of focus automatically this helps in Dimensionality Reduction.

In [8]:
```
# Standardize and center the feature columns
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Import the PCA class from sklearn
from sklearn.decomposition import PCA
pca = PCA()

# Fit the standardized data to the pca
pca.fit(features_scaled)

# Plot the proportion of variance explained on the y-axis of the bar plot
import matplotlib.pyplot as plt
plt.bar(range(1, pca.n_components_ + 1),  pca.explained_variance_ratio_)
plt.xlabel('Principal component #')
plt.ylabel('Proportion of variance explained')
plt.xticks([1, 2, 3])

# Compute the cumulative proportion of variance explained by the first two principal components
two_first_comp_var_exp = pca.explained_variance_ratio_.cumsum()[1]
print("The Cumulative Variance Of The First Two Principal Componenets Is {}".format(round(two_first_comp_var_ex
p, 5)))
```

```
The Cumulative Variance Of The First Two Principal Componenets Is 0.7947
```



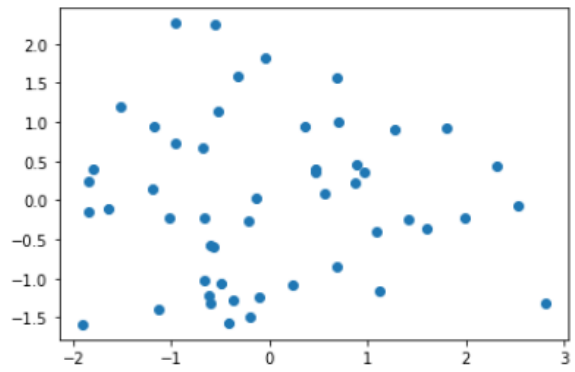
cell 9. Dimensionality Reduction

After executing cell 9 we get two components “fatal death rate due to speeding” and “fatal death rate due to alcohol” on the basis of which we create a scatter plot to see the relationship between the selected components so that we can further do our analysis.

In [9]:
```
# Transform the scaled features using two principal components
pca = PCA(n_components=2)
p_comps = pca.fit_transform(features_scaled)

# Extract the first and second component to use for the scatter plot
p_comp1 = p_comps[:, 0]
p_comp2 = p_comps[:, 1]

# Plot the first two principal components in a scatter plot
print("Scatter Plot Of First Two Principal Component:\n")
plt.scatter(p_comp1, p_comp2)
```

```
Scatter Plot Of First Two Principal Component:
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x2f78e815850>
```



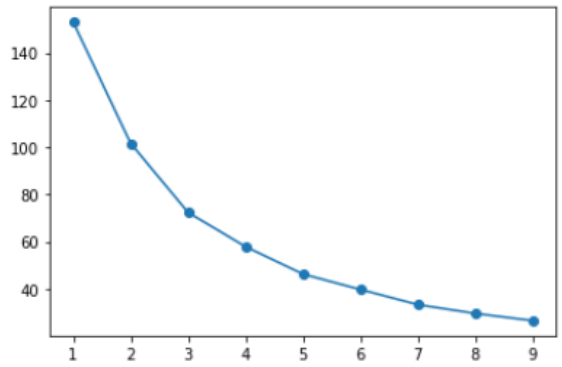
cell 10. Unsupervised Learning Using K-Means Clustering

In cell 10, we are performing K-Means clustering using sklearn library where we are giving range of 1 to 10 which means it will perform clustering using 9 centroid points and then we are taking 8 random records from our data frame object as data points around these clusters until all 9 clusters are formed and we get to know which data point is closer to which cluster. So we pass these results to matplotlib library to demonstrate Data Visualization on this K-Means Model using Scree Plot or Elbow Plot which later helps us to select the number of clusters. If you see the plot given below an elbow is forming at 1, 2, 3. So these will be our clusters we have to select while training our model. Since we have to give the number of clusters at initial stage itself in K-Means we get those points using Scree Plot and we pass number of clusters as 3.

In [10]:
```
# Import KMeans from sklearn
from sklearn.cluster import KMeans

# A loop will be used to plot the explantory power for up to 10 KMeans clusters
ks = range(1, 10)
inertias = []
for k in ks:
    # Initialize the KMeans object using the current number of clusers (k)
    km = KMeans(n_clusters=k, random_state=8)
    # Fit the scaled scaled features to the KMeans object
    km.fit(features_scaled)
    # Append the inertia for `km` to the list of inertias
    inertias.append(km.inertia_)

# Plot the results in a line plot
print("Scree Plot Or Line Plot Using KMeans Along With Principal Components:\n")
plt.plot(ks, inertias, marker='o')
```

```
Scree Plot Or Line Plot Using KMeans Along With Principal Components:
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x2f78eef9e80>]
```



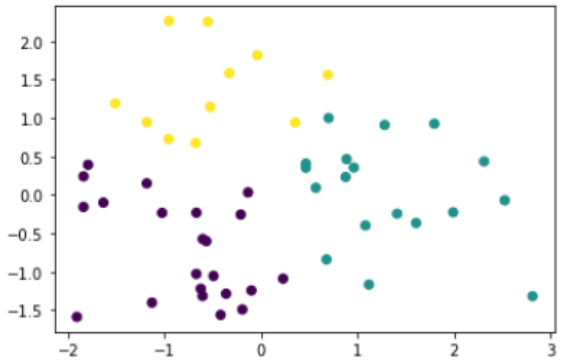
cell 11. Visualizing Clusters Using Scatter Plot

Since in previous steps, we performed Principal Component Analysis (PCA) and in cell 10 we build and train our K-Means Clustering for 8 random records from our data frame and 9 clusters, here, we will be performing K-Means again with three clusters along with Principal Components we generated in cell 9. And together we will plot a scatter plot to get more insights on our data analysis. So we get to know that on x-axis of scatter plot we have component 1 and on y-axis of scatter plot we have component 2 where data points are getting grouped with 3 clusters. Following image shows in brief:

In [11]:
```
# Create a KMeans object with 3 clusters
km = KMeans(n_clusters=3, random_state=8)
# Fit the data to the `km` object
km.fit(features_scaled)
# Create a scatter plot of the first two principal components
# and color it according to the KMeans cluster assignment
print("Visualize Clusters Using Scatter Plot, KMeans And PCA:\n")
plt.scatter(p_comps[:, 0], p_comps[:, 1], c=km.labels_)
```

```
Visualize Clusters Using Scatter Plot, KMeans And PCA:
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x2f78e0dd280>
```

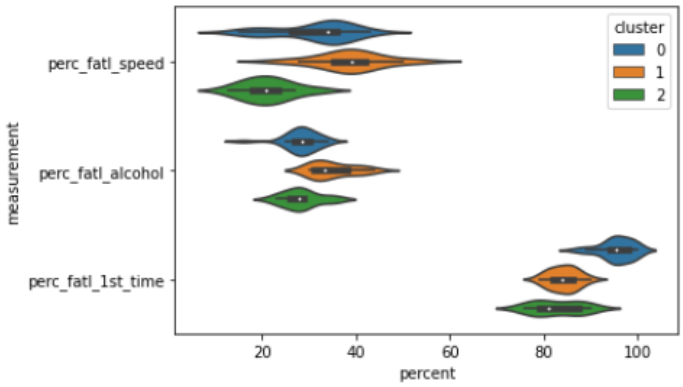


cell 12. Getting the K-Means Clustering result and Plotting it using Violin Plot

In this cell we got all our results and clusters and we added them to our existing data frame object car_acc using a new column called as clusters and then we plot a Violin Plot which gives us more detailed explaination.

In [12]:
```
# Create a new column with the labels from the KMeans clustering
car_acc['cluster'] = km.labels_

# Reshape the DataFrame to the long format
melt_car = pd.melt(car_acc, id_vars='cluster', var_name='measurement', value_name='percent',
                   value_vars=['perc_fatl_speed', 'perc_fatl_alcohol', 'perc_fatl_1st_time'])

# Create a violin plot splitting and coloring the results according to the km-clusters
sns.violinplot(y='measurement', x='percent', data=melt_car, hue='cluster')
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2f78df3c400>
```



cell 13. Dataset “miles-driven.csv” as a Data Frame Object

In cell 3, first we have imported pandas library using “import pandas as pd”. And then we have used “.info()” and “.describe()” to get some useful insights from the data frame object where “.describe()” gives statistical summary of the entire data frame object and “.info()” gives details about attributes/variables (columns) of the data frame object. Lastly we have used “.head()” to see the first 5 records of the data frame object.

In [13]:
```
# Read in the `miles-drives.csv`
miles_driven = pd.read_csv("D:/NYIT (Fall2020-ACADEMICS)/INTRODUCTION TO DATA MINING (CSCI657 or CSCI415) [PROF
F Helen Gu]/Final Projects/PROJECT1 PYTHON/datasets/miles-driven.csv", sep='|')

# Save the number of rows columns as a tuple
rows_and_cols = miles_driven.shape
print("Brief Description Of Dataset miles-driven.csv Stored In DataFrame Object miles_driven:\n")
print('There are {} rows and {} columns.\n'.format(rows_and_cols[0], rows_and_cols[1]))

# Generate an overview of the DataFrame
miles_driven_information = miles_driven.info()
print(miles_driven_information)

# Check whether there are Nan values or not
print("\nBrief Description Whether DataFrame Object miles_driven Consists Any NaN Values:\n", miles_driven.isna
().sum(), sep="")

# Display the first five rows of the DataFrame
print("\nFollowing Is The Generic Overview Of DataFrame Object miles_driven:\n")
miles_driven.head()
```

```
Brief Description Of Dataset miles-driven.csv Stored In DataFrame Object miles_driven:

There are 51 rows and 2 columns.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 2 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   state                  51 non-null     object
 1   million_miles_annually  51 non-null     int64
dtypes: int64(1), object(1)
memory usage: 944.0+ bytes
None

Brief Description Whether DataFrame Object miles_driven Consists Any NaN Values:
state                   0
million_miles_annually    0
dtype: int64

Following Is The Generic Overview Of DataFrame Object miles_driven:
```

Out[13]:

	state	million_miles_annually
0	Alabama	64914
1	Alaska	4593
2	Arizona	59575
3	Arkansas	32953
4	California	320784

cell 14. Updating “miles_driven” Data Frame Object

In cell 14, we have to update our existing data frame “miles-driven” object as it initially consists one column as million miles driven and later we have to merge this data frame object with ca_acc and in that we have one column which displays death rate of number of drivers per billion miles. Hence we have to update “miles-driven” data frame object before heading to next steps.

In [14]:
```
# Update the DataFrame `miles_driven` by converting `million_miles_annually` to `billion_miles_annually`
miles_driven['million_miles_annually'] = (miles_driven['million_miles_annually'] / 1000)
miles_driven.rename(columns={'million_miles_annually':'billion_miles_annually'}, inplace=True)
miles_driven.head()
```

Out[14]:

	state	billion_miles_annually
0	Alabama	64.914
1	Alaska	4.593
2	Arizona	59.575
3	Arkansas	32.953
4	California	320.784

cell 15. Merging Data Frame Objects “miles-driven” and “car_acc”

In this cell we are just merging “car-acc” data frame object with “miles-driven” data frame object using “.merge()” function.

In [15]:
```
# Merge the `car_acc` DataFrame with the `miles_driven` DataFrame
car_acc_miles = car_acc.merge(miles_driven, on='state')
print("After Merging We Get DataFrame Object car_acc_miles As:\n")
car_acc_miles.head()
```

```
After Merging We Get DataFrame Object car_acc_miles As:
```

Out[15]:

	state	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time	cluster	billion_miles_annually
0	Alabama	18.8	39	30	80	1	64.914
1	Alaska	18.1	41	25	94	0	4.593
2	Arizona	18.6	35	28	96	0	59.575
3	Arkansas	22.4	18	26	95	0	32.953
4	California	12.0	35	28	89	0	320.784

cell 16. After Merging Add New Column

Since we understood that in car_acc data frame object we had column “driver death rates per billion miles” but in miles-driven we have column “billion miles annually” so we have to add a new column saying “number of fatal” after we multiply “driver death rates per billion miles” with “billion miles annually”. This intern helps us justify our entire work completely by saying we have clusters, we have each individual states corresponding to respective clusters, and lastly proper death rates.

In [16]:
```
# Create a new column for the number of drivers involved in fatal accidents
car_acc_miles['num_drvr_fatl_col'] = car_acc_miles['drvr_fatl_col_bmiles'] * car_acc_miles['billion_miles_annua
lly']
print("\nFollowing Is The Generic Overview Of DataFrame Object car_acc_miles After Adding New Column:\n")
car_acc_miles.head()
```

```
Following Is The Generic Overview Of DataFrame Object car_acc_miles After Adding New Column:
```

Out[16]:

	state	drvr_fatl_col_bmiles	perc_fatl_speed	perc_fatl_alcohol	perc_fatl_1st_time	cluster	billion_miles_annually	num_drvr_fatl_col
0	Alabama	18.8	39	30	80	1	64.914	1220.3832
1	Alaska	18.1	41	25	94	0	4.593	83.1333
2	Arizona	18.6	35	28	96	0	59.575	1108.0950
3	Arkansas	22.4	18	26	95	0	32.953	738.1472
4	California	12.0	35	28	89	0	320.784	3849.4080

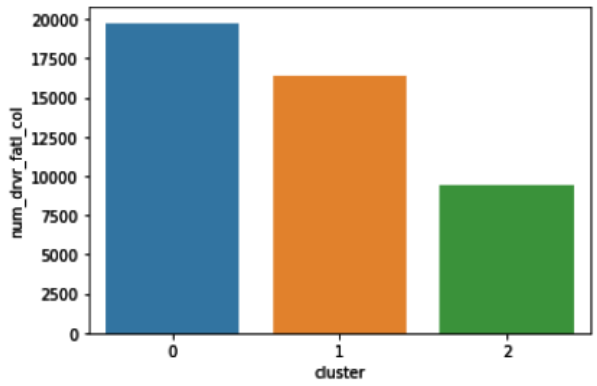
cell 17. Plotting Bar Plot using Data And Info Collected In Previous Cells

In this cell we are plotting Bar Plot using seaborn library where on x-axis we are giving clusters, on y-axis we are giving number of driver fatal column which we generated in our previous cell, data frame object as “car_acc_miles” and also calculating Count, Mean and Sum and grouping using “.groupby()” and“.agg()” functions.

In [17]:
```
# Create a barplot of the total number of accidents per cluster
sns.barplot(x='cluster', y='num_drvr_fatl_col', data=car_acc_miles, estimator=sum, ci=None)

# Calculate the number of states in each cluster and their 'num_drvr_fatl_col' mean and sum.
count_mean_sum = car_acc_miles.groupby('cluster')['num_drvr_fatl_col'].agg(['count', 'mean', 'sum'])
count_mean_sum
```

Out[17]:

cluster	count	mean	sum
0	22	898.378595	19764.3291
1	18	911.406439	16405.3159
2	11	860.505945	9465.5654



cell 18. Looping on “car_acc_miles” Frame Object

Using for loop on car_acc_miles data frame object to get all the individual states which are corresponding to Cluster 0 which shows that high road fatality is due to speeding. Displaying only those states which are grouped under Cluster 0.

In [18]:
```
# Directing corresponding states to cluster 0
print("Following Are The Respective States Corresponding To Cluster 0:\n")
for i in range(len(car_acc_miles)):
    if car_acc_miles.loc[i].cluster == 0:
        print(car_acc_miles.loc[i].state)
```

```
Following Are The Respective States Corresponding To Cluster 0:

Alaska
Arizona
Arkansas
California
Colorado
Delaware
District of Columbia
Florida
Georgia
Idaho
Illinois
Indiana
Louisiana
Maryland
Mississippi
Nevada
New Mexico
Oklahoma
Oregon
Utah
Vermont
West Virginia
```

CONCLUSION

As we can see, there is no other obvious correct choice regarding which cluster is the most important to focus on. Yet, we can conclude from above demonstrated work that all states residing in cluster 0 are of high priority, also to reduce road fatalities in the United States we should focus on cluster 0 that basically shows us the fatality rates caused due to excessive speeding.

REFERENCES

- https://github.com/shubendu/Reducing-Traffic-Mortality-in-the-USA/blob/master/datasets/
  (Last Accessed 8th Dec 2020)
- https://www.paho.org/hq/index.php?option=com_content&view=article&id=5163:about-road-safety&Itemid=39898&lang=en
  (Last Accessed 8th Dec 2020)
- https://learn.datacamp.com/projects/462
  (Last Accessed 8th Dec 2020)

- **Data Science With Python3 course** , Data-camp , (Last Accessed 8th Dec 2020)
- **Machine Learning Nanodegree Program** , Udacity Last Accessed 8th Dec 2020)
- **Data Science for all** channel , Youtube (Last Accessed 8th Dec 2020)

WORK DONE

MEHREET SINGH BAJAJ

- Understanding R and doing crash course (Harvard Data Science certificate in R)
- Datacamp python course
- Installing and running r environment and R-studio software
- Understanding the basic concept of both projects in python and R
- Making the report in a proper structure and explaining it
- Understanding working of Tableau software
- Presentation Contents

RISHABH BADJATYA

- Data Mining Tools/Software - Tableau
- Presentation Content
- Company (website) that provides the software/tool
- The functionalities of tools/software
- How to use it
- Applications
- Demo

MOHAMMED SAFWAN ASLAM KAZI

- Reducing Traffic Mortality in The USA
- Understanding Concepts of Python3
- Machine Learning Algorithms,
- Installation of Python3, WinPY and Jupyter Notebook, Coding, Development and Deployment of Project)
- Report For Project 2 :- Reducing Traffic Mortality in The USA ()

MAITRY JARIWALA