Using K Means Clustering to Establish Patterns in Heart Disease Patients

Sahibzada Ali Mahmud

June 10, 2019

Executive Summary

The main purpose of Clustering algorithms is to group items together based on certain similarities. Clustering algorithms are beneficial in unsupervised learning when data sets are not labelled and the input variables are available without the corresponding output variables. In our project, we will analyze a dataset with anonymized patients with underlying heart conditions. If we are able to classify patients with similar characteristics, then the chances of them responding to the same treatment regime will be higher.The data set we will use is from V.A. Medical Center in Long Beach, CA and can be downloaded here. We shall make use of K-Means Clustering and Hierarchical Clustering to establish patters. Furthermore, we shall also determine the optimal options for clustering for our scenario.

Methodology Used

The methodology used for this project consists of loading the required libraries, exploratory data analysis, scaling the data through z-score standardization, using K-Means and Hierarchical clustering, Data visualization, Comparison of results, and using internal and stability validation to determine the optimal parameters to be used for a clustering algorithm.

1. Loading Libraries and Checking Data

At the outset, we will load the required libraries and have a look at the patient data.Also it is important that the data we use for the clustering algorithm is numeric.

```
#Loading the required libraries
library(dplyr)
library(tidyverse)
```

```
# Loading the data
heart_disease = read.csv("datasets/heart_disease_patients.csv")

# Print the first twenty rows of the data set
head(heart_disease, n=20)
```

```
##    id age sex cp trestbps chol fbs restecg thalach exang oldpeak slope
## 1   1  63   1  1      145  233   1       2     150     0     2.3     3
## 2   2  67   1  4      160  286   0       2     108     1     1.5     2
## 3   3  67   1  4      120  229   0       2     129     1     2.6     2
## 4   4  37   1  3      130  250   0       0     187     0     3.5     3
## 5   5  41   0  2      130  204   0       2     172     0     1.4     1
## 6   6  56   1  2      120  236   0       0     178     0     0.8     1
## 7   7  62   0  4      140  268   0       2     160     0     3.6     3
## 8   8  57   0  4      120  354   0       0     163     1     0.6     1
```

```
## 9   9  63   1  4      130  254   0       2     147     0     1.4     2
## 10 10  53   1  4      140  203   1       2     155     1     3.1     3
## 11 11  57   1  4      140  192   0       0     148     0     0.4     2
## 12 12  56   0  2      140  294   0       2     153     0     1.3     2
## 13 13  56   1  3      130  256   1       2     142     1     0.6     2
## 14 14  44   1  2      120  263   0       0     173     0     0.0     1
## 15 15  52   1  3      172  199   1       0     162     0     0.5     1
## 16 16  57   1  3      150  168   0       0     174     0     1.6     1
## 17 17  48   1  2      110  229   0       0     168     0     1.0     3
## 18 18  54   1  4      140  239   0       0     160     0     1.2     1
## 19 19  48   0  3      130  275   0       0     139     0     0.2     1
## 20 20  49   1  2      130  266   0       0     171     0     0.6     1
```

```
# Let's check general information  about the data!
str(heart_disease)
```

```
## 'data.frame':    303 obs. of  12 variables:
##  $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : int  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : int  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : int  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : int  3 2 2 3 1 1 3 1 2 3 ...
```

```
# Checking for only numeric variables
lapply(heart_disease, class)
```

```
## $id
## [1] "integer"
##
## $age
## [1] "integer"
##
## $sex
## [1] "integer"
##
## $cp
## [1] "integer"
##
## $trestbps
## [1] "integer"
##
## $chol
## [1] "integer"
##
## $fbs
## [1] "integer"
```

```
##
## $restecg
## [1] "integer"
##
## $thalach
## [1] "integer"
##
## $exang
## [1] "integer"
##
## $oldpeak
## [1] "numeric"
##
## $slope
## [1] "integer"
```

2. Exploratory Data Analysis and Z-Score Standardization

The popular clustering algorithms such as K Means clustering and Hierarchical clustering measure the similarity between points using a distance formula. Therefore, we will carry out some prelimenary data exploratory analysis to check whether we require scaling through z-score transformation and reduce the the distance related bias. To ensure that the features contribute relatively equally to the distance formule, we will use z-score standardization approach. In Z-score standardization, the mean of each feature X is subtracted from each value of feature X and divided by the standard deviation of feature value X:

$$X_{new} = \frac{X - Mean(X)}{Std(X)}$$

In R, the scale() function makes it convenient to carry out the z-score standardization for us.

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

```
##        id              age             sex              cp
##  Min.   :  1.0   Min.   :29.00   Min.   :0.0000   Min.   :1.000
##  1st Qu.: 76.5   1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000
##  Median :152.0   Median :56.00   Median :1.0000   Median :3.000
##  Mean   :152.0   Mean   :54.44   Mean   :0.6799   Mean   :3.158
##  3rd Qu.:227.5   3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000
##  Max.   :303.0   Max.   :77.00   Max.   :1.0000   Max.   :4.000
##     trestbps          chol            fbs            restecg
##  Min.   : 94.0   Min.   :126.0   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:120.0   1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :130.0   Median :241.0   Median :0.0000   Median :1.0000
##  Mean   :131.7   Mean   :246.7   Mean   :0.1485   Mean   :0.9901
##  3rd Qu.:140.0   3rd Qu.:275.0   3rd Qu.:0.0000   3rd Qu.:2.0000
##  Max.   :200.0   Max.   :564.0   Max.   :1.0000   Max.   :2.0000
##     thalach          exang           oldpeak         slope
##  Min.   : 71.0   Min.   :0.0000   Min.   :0.00   Min.   :1.000
##  1st Qu.:133.5   1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000
##  Median :153.0   Median :0.0000   Median :0.80   Median :2.000
##  Mean   :149.6   Mean   :0.3267   Mean   :1.04   Mean   :1.601
##  3rd Qu.:166.0   3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000
##  Max.   :202.0   Max.   :1.0000   Max.   :6.20   Max.   :3.000
```

```r
# Removing the unnecessary id column
heart_disease = heart_disease[ , !(names(heart_disease) %in% c("id"))]

# Scaling data through Z-Score Standardization and saving as a data frame
scaled = scale(heart_disease)

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

```
##       age                sex                cp              trestbps
##  Min.   :-2.8145   Min.   :-1.4549   Min.   :-2.2481   Min.   :-2.14149
##  1st Qu.:-0.7124   1st Qu.:-1.4549   1st Qu.:-0.1650   1st Qu.:-0.66420
##  Median : 0.1727   Median : 0.6851   Median :-0.1650   Median :-0.09601
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.7259   3rd Qu.: 0.6851   3rd Qu.: 0.8765   3rd Qu.: 0.47218
##  Max.   : 2.4961   Max.   : 0.6851   Max.   : 0.8765   Max.   : 3.88132
##       chol               fbs             restecg             thalach
##  Min.   :-2.3310   Min.   :-0.4169   Min.   :-0.995103   Min.   :-3.4364
##  1st Qu.:-0.6894   1st Qu.:-0.4169   1st Qu.:-0.995103   1st Qu.:-0.7041
##  Median :-0.1100   Median :-0.4169   Median : 0.009951   Median : 0.1483
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.0000
##  3rd Qu.: 0.5467   3rd Qu.:-0.4169   3rd Qu.: 1.015005   3rd Qu.: 0.7166
##  Max.   : 6.1283   Max.   : 2.3905   Max.   : 1.015005   Max.   : 2.2904
##      exang            oldpeak            slope
##  Min.   :-0.6955   Min.   :-0.8954   Min.   :-0.9747
##  1st Qu.:-0.6955   1st Qu.:-0.8954   1st Qu.:-0.9747
##  Median :-0.6955   Median :-0.2064   Median : 0.6480
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.4331   3rd Qu.: 0.4827   3rd Qu.: 0.6480
##  Max.   : 1.4331   Max.   : 4.4445   Max.   : 2.2708
```

3. Applying K-Means Algorithms to Group Patients

After carrying out the required scaling of data, we can now apply the k-means algorithm. In order to ensure reproducibility, we will set a seed value for convenience of anyone who would like to verify the results.

```r
# Set the seed so that results are reproducible
seed_val = 10
set.seed(seed_val, kind = "Mersenne-Twister", normal.kind = "Inversion")

# Select a number of clusters
k = 5

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

# Checking the Number of patients in each group
first_clust$size
```

```
## [1] 24 53 87 72 67
```

```r
# Extracting the Cluster centroids for more insight
Cluster_centroids_1 <- as.data.frame(first_clust$centers)

# Checking the location of each cluster centroid
Cluster_centroids_1
```

```
##          age        sex         cp   trestbps        chol        fbs
## 1  0.7120585 -1.1873943  0.7029471  1.3505060  1.25970414  0.5188646
## 2  0.4294674  0.5639394 -0.6759343  0.2020207 -0.16935232  0.2716697
## 3 -0.5801236  0.6850692 -0.1889381 -0.1874438 -0.26557279 -0.1587905
## 4  0.2464660  0.5067393  0.7463442 -0.1457276 -0.03779596 -0.0270242
## 5 -0.1063578 -1.4548891 -0.2738115 -0.2435708  0.06819277 -0.1655333
##      restecg     thalach      exang    oldpeak      slope
## 1  0.4706008 -0.27244852  0.3688152  0.9061109  0.7156571
## 2  0.3323269 -0.07851103 -0.5749938  0.5362790  0.7398967
## 3 -0.3250670  0.70657276 -0.4508145 -0.6350194 -0.8068658
## 4  0.1076646 -0.95611569  1.3148554  0.5675836  0.5804253
## 5 -0.1250562  0.26967684 -0.5048600 -0.5341610 -0.4176656
```

4. An Additional Iteration of K-Means Algorithm

Different iterations of K-Means may yield different cluster assignments since the cluster centers are selected through random selection of points. For consistency, it is important to check whether cluster assignments show similarity among different iterations of the algorithms. Therefore, we are going to check the patient grouping through application of another iteration of K-Means clustering algorithm.

```r
# Set the seed
seed_val = 38
set.seed(seed_val, kind = "Mersenne-Twister", normal.kind = "Inversion")

# Run the k-means algorithms
k = 5
second_clust = kmeans(scaled, centers = k, nstart = 1)

# Checking the number of patients in each group for Comparison
second_clust$size
```

```
## [1] 51 48 42 67 95
```

```r
# Checking the location of each cluster centroid
Cluster_centroids_2 <- as.data.frame(first_clust$centers)

# Checking the location of each cluster centroid
Cluster_centroids_2
```

```
##          age        sex         cp   trestbps        chol        fbs
## 1  0.7120585 -1.1873943  0.7029471  1.3505060  1.25970414  0.5188646
## 2  0.4294674  0.5639394 -0.6759343  0.2020207 -0.16935232  0.2716697
## 3 -0.5801236  0.6850692 -0.1889381 -0.1874438 -0.26557279 -0.1587905
## 4  0.2464660  0.5067393  0.7463442 -0.1457276 -0.03779596 -0.0270242
## 5 -0.1063578 -1.4548891 -0.2738115 -0.2435708  0.06819277 -0.1655333
```

```
##      restecg     thalach      exang    oldpeak      slope
## 1  0.4706008 -0.27244852  0.3688152  0.9061109  0.7156571
## 2  0.3323269 -0.07851103 -0.5749938  0.5362790  0.7398967
## 3 -0.3250670  0.70657276 -0.4508145 -0.6350194 -0.8068658
## 4  0.1076646 -0.95611569  1.3148554  0.5675836  0.5804253
## 5 -0.1250562  0.26967684 -0.5048600 -0.5341610 -0.4176656
```

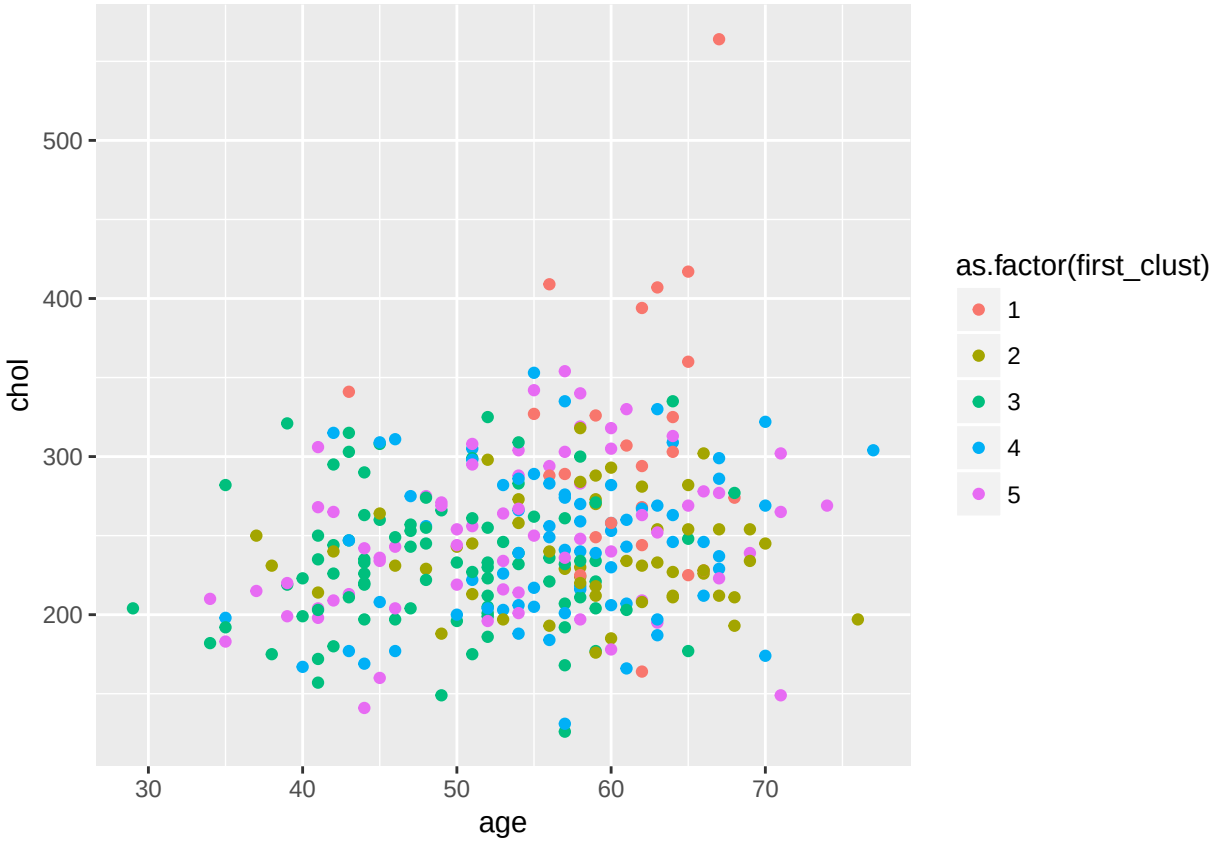
5. Comparing patient clusters

To ensure stability, the clusters resulting from different iterations of K-Means algorithm should roughly have similar sizes and similar distribution of variables. In case of large variations of aforementioned parameters among different iterations of K-Means algorithm, then K-Means cannot be termed as a good choice and other clustering methods may be considered. Visualization can help to an extent to determine the stability. We are going to use ggplot for the purpose.

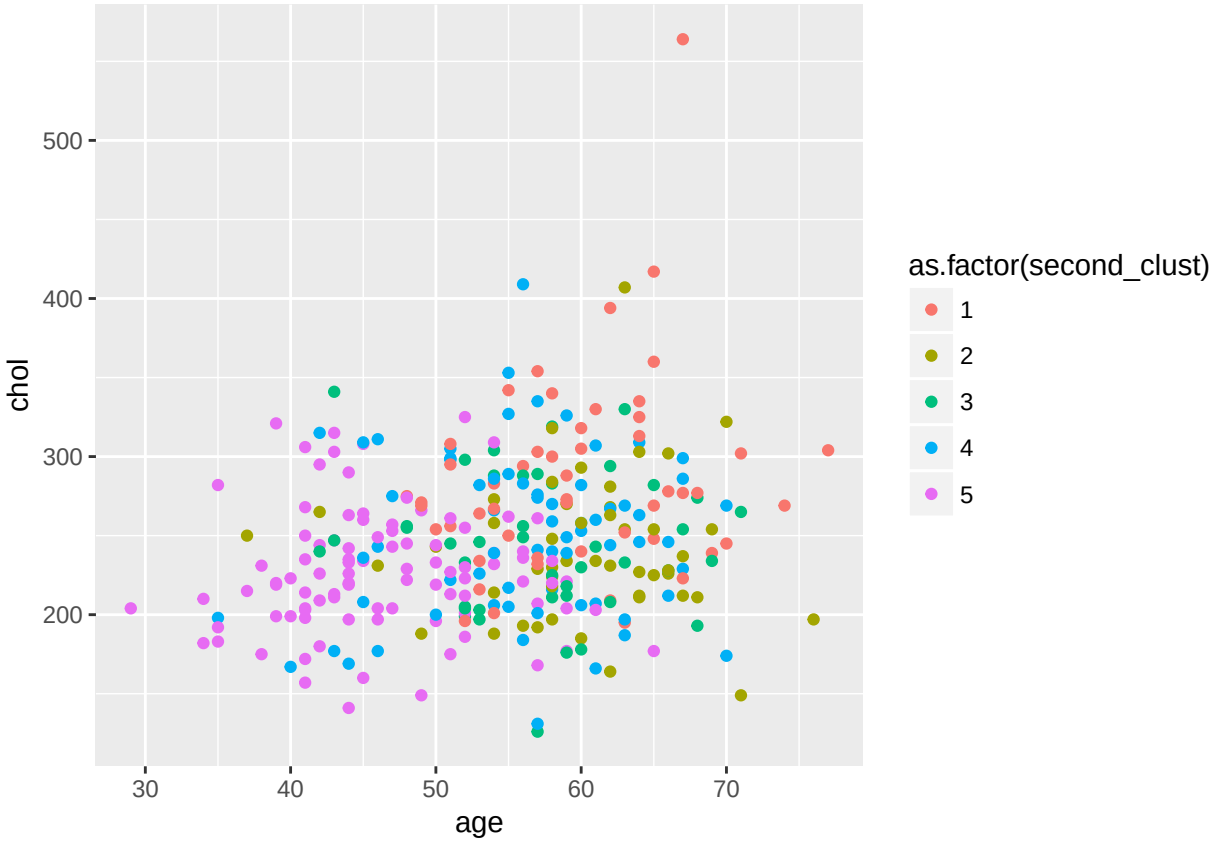
```
# Adding cluster assignments to the data
heart_disease[ , "first_clust"] = first_clust$cluster
heart_disease[ , "second_clust"] = second_clust$cluster

# Check and Load ggplot2
if("ggplot2" %in% rownames(installed.packages()) == FALSE) {install.packages("ggplot2")}
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
# Creating the plots of age and chol for the first clustering algorithm
plot_one = ggplot(heart_disease, aes(x =age, y = chol, color = as.factor(first_clust))) +
  geom_point()
plot_one
```




```
# Creating the plots of age and chol for the second clustering algorithm
plot_two = ggplot(heart_disease, aes(x = age, y = chol, color = as.factor(second_clust))) + geom_point()
plot_two
```

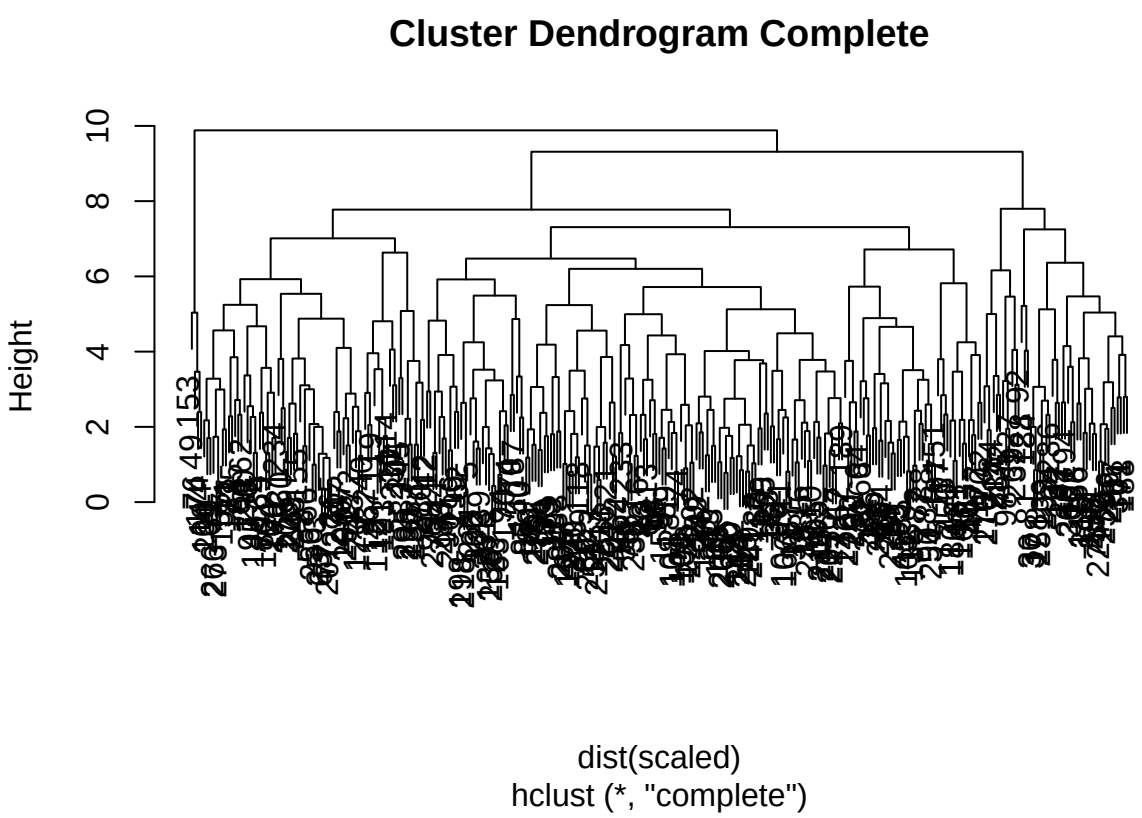


6. Hierarchical clustering: An Alternative

An alternative is Hierarchical clustering in which it is not necessary to specify the number of clusters when running the algorithm. It gives good results when the data has a nested structure. Clusters can be selected by using the dendrogram. When the algorithm is applied on data, the distance matrix is automatically calculated. There are two types of Hierarchical Clustering. - Agglomerative - Divisive

In Agglomerative approach, all the data points are initially considered as individual clusters and the algorithm works its way from the bottom up. It is the commonly used approach. In the Divisive approach, the whole data set is considered to be a single cluster and the algorithm works its way from top to bottom to create a dendrogram. The dendrogram allows one to see how similar observations are to one another and are useful in selecting the number of clusters to group the data

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

# Printing the dendrogram
plot(hier_clust_1, main = "Cluster Dendrogram Complete")
```

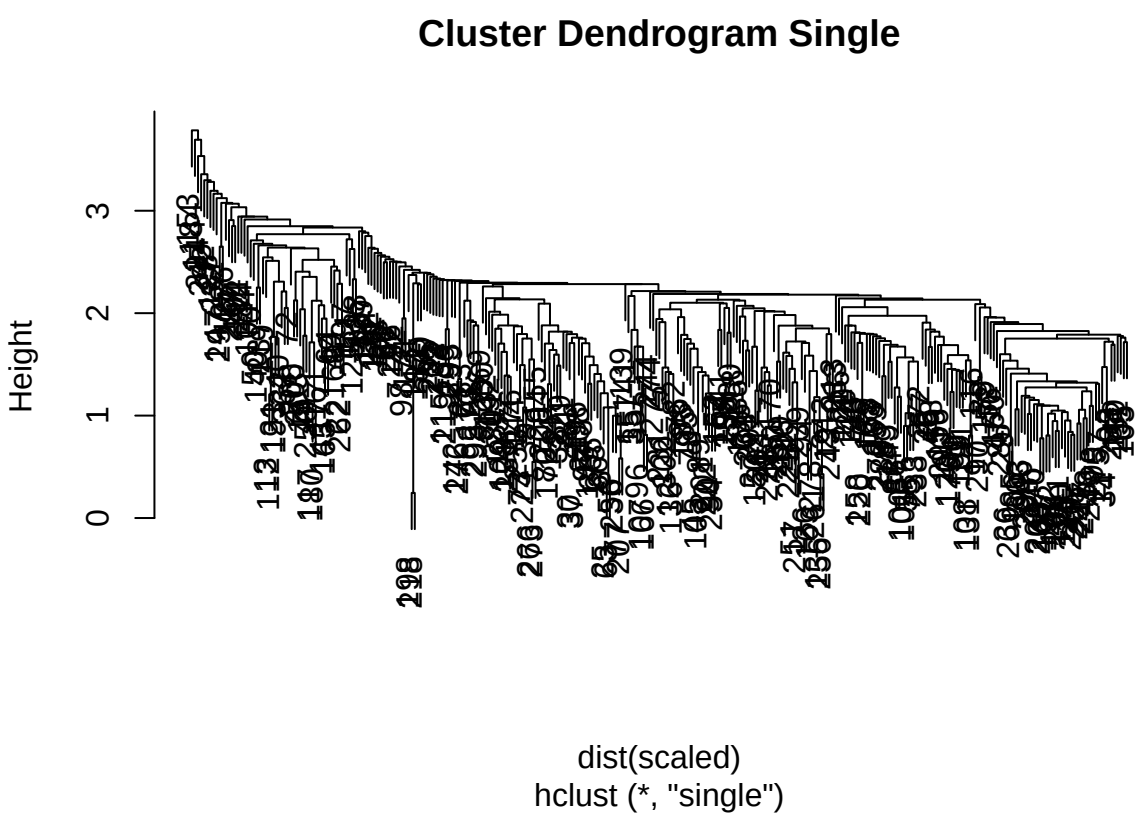



```
# Getting cluster assignments based on number of selected clusters
hc_1_assign <- cutree(hier_clust_1, k = 5)
```

7. 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. In our case, we intend to look at the dissimilarity between the patients through the smallest difference between patients and minimize that difference when grouping together clusters.

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

# Printing the dendrogram
plot(hier_clust_2, main = "Cluster Dendrogram Single")
```




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

8. Comparing clustering results

The idea behind grouping patients with similar characteristics in clusters is to increase the probability of them responding to the same treatment options. If the grouping is not based on certain similar characteristics, then some patients may not respond to the same treatment that is being administered to the group and hence indicative of noise. Therefore, similar characteristiscs or patterns need to be evident from the clusters formed. We are going to observe that from the results of the two Hierachical clustering algorithms through the distribution of variables.

```
# Adding assignments of chosen hierarchical linkage
heart_disease['hc_clust'] = hc_1_assign

# Remove 'sex', 'first_clust', and 'second_clust' variables
hd_simple = heart_disease[, !(names(heart_disease) %in% c("sex", "first_clust", "second_clust"))]

# Getting mean and standard deviation summary statistics

clust_summary = do.call(data.frame, aggregate(. ~ hc_clust, data = hd_simple, function(x) c(avg = mean(
clust_summary
```

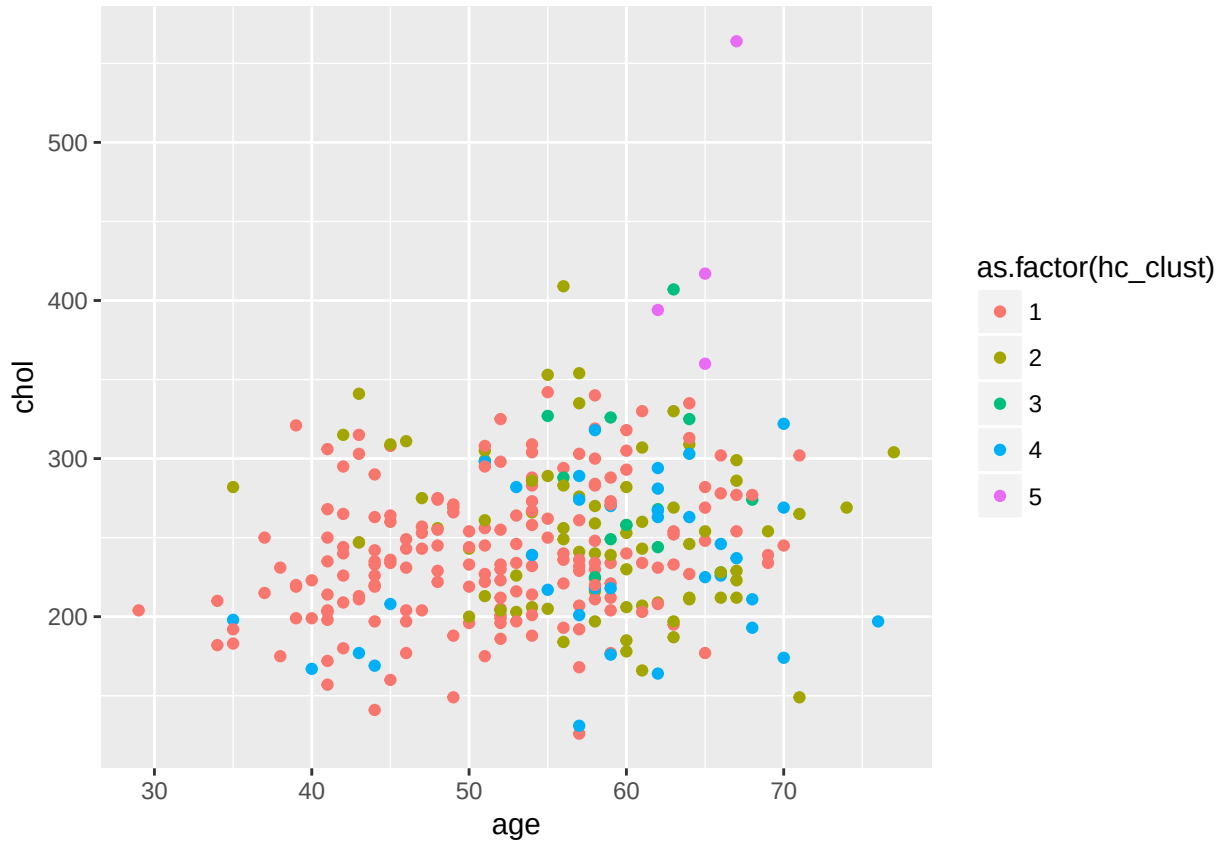
```
##   hc_clust  age.avg   age.sd   cp.avg     cp.sd trestbps.avg trestbps.sd
```

```
## 1        1 51.41667 8.540979 2.783333 0.9470625     129.1389    15.93800
## 2        2 58.11111 7.754246 3.763889 0.6165112     130.0417    13.90657
## 3        3 61.00000 3.908034 3.916667 0.2886751     168.5000    17.45904
## 4        4 59.00000 9.203580 3.571429 0.8501112     134.7714    18.64070
## 5        5 64.75000 2.061553 3.250000 0.5000000     138.7500    18.42779
##   chol.avg  chol.sd   fbs.avg    fbs.sd restecg.avg restecg.sd thalach.avg
## 1 239.8722 42.29228 0.1222222 0.3284559   0.8444444  0.9905826    161.5722
## 2 253.2222 49.74476 0.1805556 0.3873488   1.4027778  0.9140488    135.5417
## 3 284.9167 53.00336 0.3333333 0.4923660   1.2500000  0.9653073    147.7500
## 4 233.8571 49.67136 0.1428571 0.3550358   0.6857143  0.9321521    116.8857
## 5 433.7500 89.93470 0.2500000 0.5000000   2.0000000  0.0000000    156.2500
##   thalach.sd  exang.avg  exang.sd oldpeak.avg oldpeak.sd slope.avg
## 1  15.779214 0.07777778 0.2685686    0.555000  0.7847196  1.388889
## 2  17.991342 0.81944444 0.3873488    1.451389  1.0804268  1.750000
## 3  13.157266 0.75000000 0.4522670    2.316667  1.4708274  2.166667
## 4  17.842071 0.48571429 0.5070926    2.240000  1.3856831  2.200000
## 5   3.774917 0.00000000 0.0000000    1.100000  0.3829708  1.500000
##    slope.sd
## 1 0.5730336
## 2 0.5240686
## 3 0.5773503
## 4 0.4058397
## 5 0.5773503
```

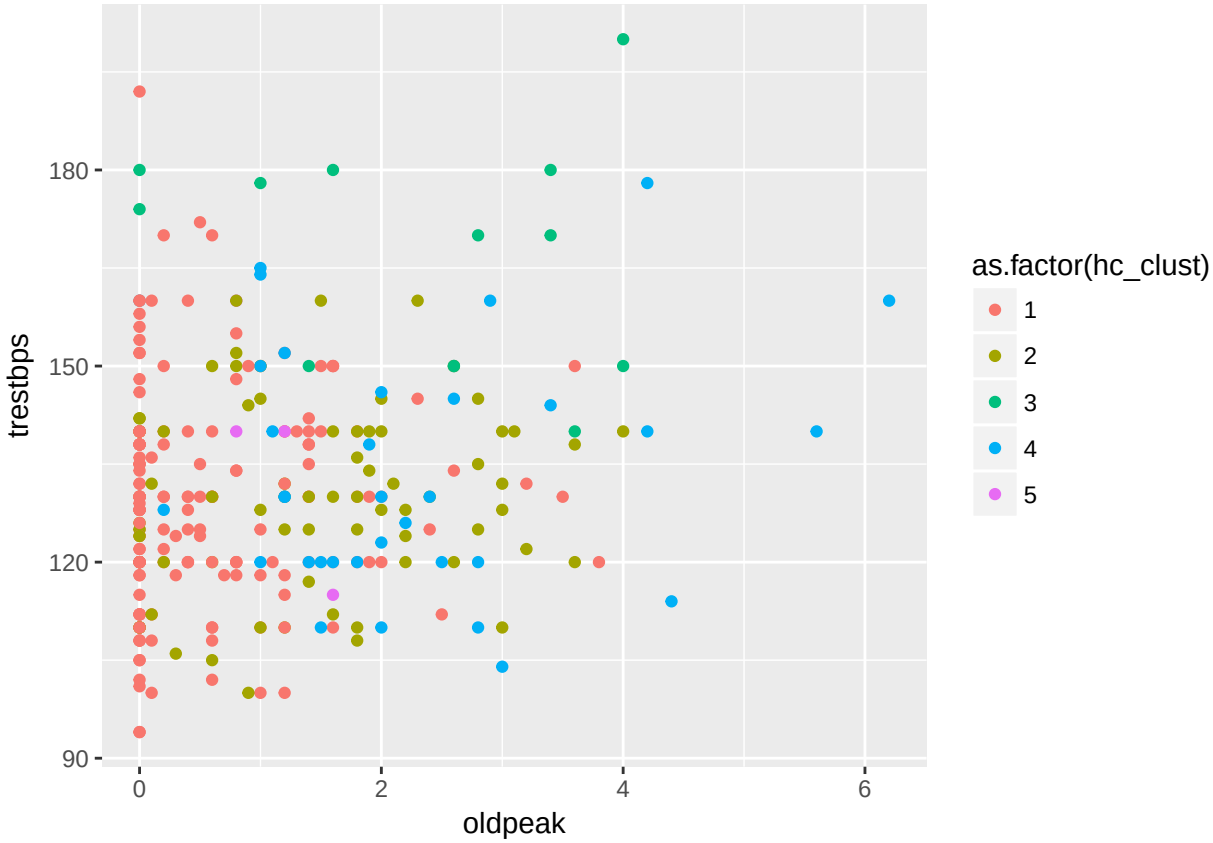
9. Visualizing the cluster contents

We are going to use visualizations to evalute the hierarchical clustering algorithms through scatter plots. The idea to to observe the patterns that appear in the data of grouped patients.

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




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



10. Using Validation Measures to Determine the Optimal Options

When using clustering algorithms, a good idea is to determine which algorithm would suit a particular situation and perform comparatively well. Also, disovering and selecting the optimal parameters for a clustering algorithm also improves its performance. However, determining the appropriate algorithm and selecting the optimal features requires additional work. Thankfully, R has a package called "clvalid" which makes it easy to carry out the aforementioned tasks.

The package makes use of several methods to determine quality of clustering through Internal measures and sTability of clustering through stability measures. Internal measures check the compactness, connectedness, and separation of the cluster partitions while the stability measures make use of Average Proportion of Non-overlap (APN), Average Distance (AD), Average Distance between Means (ADM), and Figure of Merit (FOM).

```
#Installing and loading the clValid Package
if("clValid" %in% rownames(installed.packages()) == FALSE) {install.packages("clValid")}
library(clValid)
```

```
## Warning: package 'clValid' was built under R version 3.4.4
```

```
## Loading required package: cluster
```

```
#Checking Quality of Clustering through Internal Validation
intern <- clValid(heart_disease, 2:6, clMethods = c("hierarchical", "kmeans"), validation = "internal")
```

```
## Warning in clValid(heart_disease, 2:6, clMethods = c("hierarchical",
## "kmeans"), : rownames for data not specified, using 1:nrow(data)
```

```
#Check the Summary of Internal Validation
summary(intern)
```

```
##
## Clustering Methods:
##  hierarchical kmeans
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2        3        4        5        6
##
## hierarchical Connectivity    5.4282   7.5115  37.2357  56.7944  57.1167
##              Dunn            0.1701   0.1701   0.0756   0.0773   0.0773
##              Silhouette      0.6218   0.5426   0.3600   0.2797   0.2566
## kmeans       Connectivity   41.5052  57.6881  75.5865 108.5512 118.1817
##              Dunn            0.0238   0.0397   0.0771   0.0485   0.0419
##              Silhouette      0.3890   0.3558   0.2827   0.2381   0.2377
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 5.4282 hierarchical 2
## Dunn         0.1701 hierarchical 2
## Silhouette   0.6218 hierarchical 2
```

```
#Checking Stability of Clustering through Internal Validation
stab <- clValid(heart_disease, 2:6, clMethods=c("hierarchical","kmeans"), validation="stability")
```

```
## Warning in clValid(heart_disease, 2:6, clMethods = c("hierarchical",
## "kmeans"), : rownames for data not specified, using 1:nrow(data)
```

```
#Check Optimal Scores
optimalScores(stab)
```

```
##            Score       Method Clusters
## APN 9.366229e-04 hierarchical        2
## AD  4.518131e+01       kmeans        6
## ADM 4.713387e-01 hierarchical        2
## FOM 7.772795e+00       kmeans        6
```

11. Conclusion

Based on internal validation, the quality of clustering is better for hierarchical clustering for a cluster size of 2. When considering stability measures, the optimal scores show that K-Means clustering performs better for a cluster size of 6 based on Figure of Merit (FOM) and Average Distance (AD). The Hierarchical Clustering show good stability results for cluster size of 2 and based on Average Distance between Means (ADM) and

Average Proportion of Non-Overlap (APN). Therefore, as future work, the internal validation and stabilty validation measures can be used as a yardstick to initially select parameters for a given data set to possibly get optimal results.