Clustering Heart Disease Patient Data

Introduction

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 will 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.

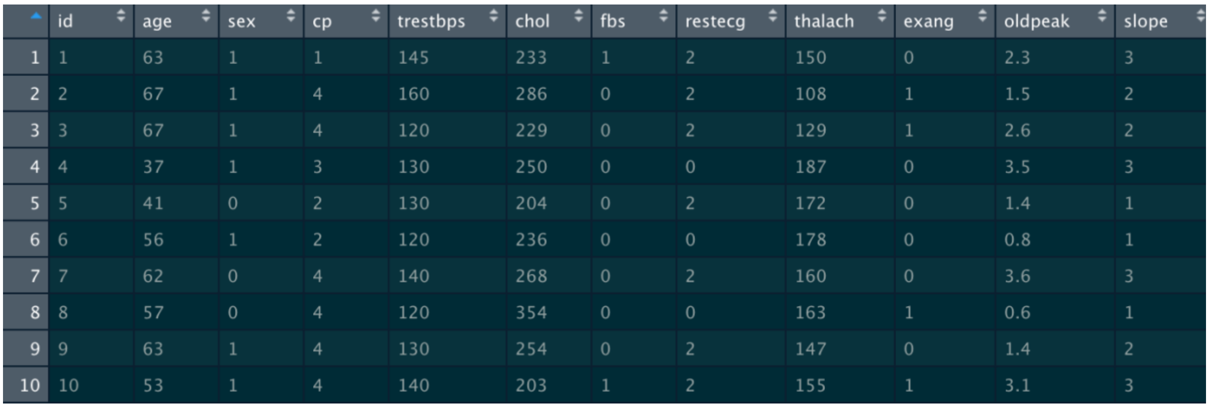
The dataset

The data we are analyzing comes from the V.A. Medical Center in Long Beach, CA. It has the following columns:

- id: patient identification number
- age: patient's age in years
- sex: patient gender(1 = male; 0 = female)
- cp: chest pain type(1 = typical angina, 2 = atypical angina, 3 = non-anginal lain, 4 = asymptomatic)
- testbps: resting blood pressure(mm Hg)
- chol: serum cholestoral(mg/dl)
- fbs: fasting blood sugar > 120 mg/dl (1 = true, 0 = false)
- restecg: resing electrocariographi results (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy)
- thalach: maximum heart rate achieved
- exang: exercise induced angina(1 = yes, 0 = no)
- oldpeak: ST depression induced by exercise relative to rest
- slope: the slope of the peak exercise ST segment

Let's load the datasets and packages and take a look at the first few rows.

```{r}
library(tidyverse)
library(ggplot2)

heart_disease <- read_csv("heart_disease_patients.csv")
```

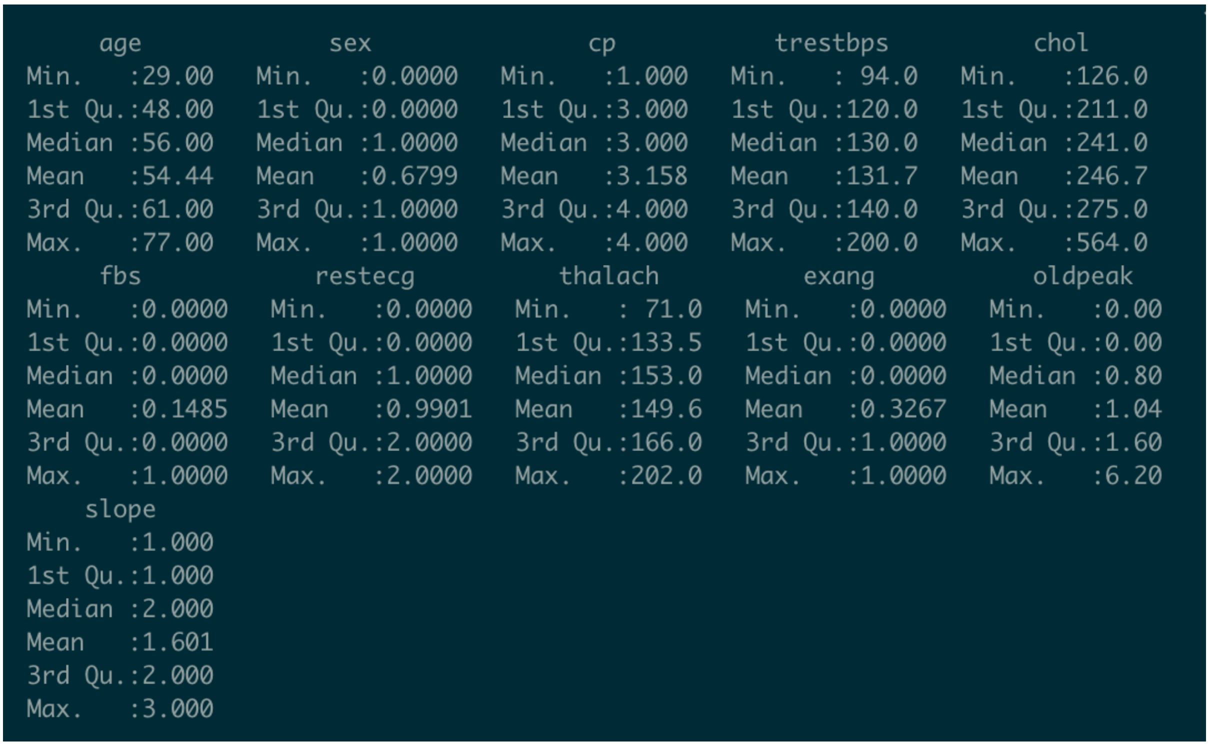


	id	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
1	1	63	1	1	145	233	1	2	150	0	2.3	3
2	2	67	1	4	160	286	0	2	108	1	1.5	2
3	3	67	1	4	120	229	0	2	129	1	2.6	2
4	4	37	1	3	130	250	0	0	187	0	3.5	3
5	5	41	0	2	130	204	0	2	172	0	1.4	1
6	6	56	1	2	120	236	0	0	178	0	0.8	1
7	7	62	0	4	140	268	0	2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

It is essential 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 decide whether we should scale the data. Because k-means and hierarchical clustering measure similarity between points using distance formula, it can emphasize certain variables with a larger scale and thus more significant differences between points.

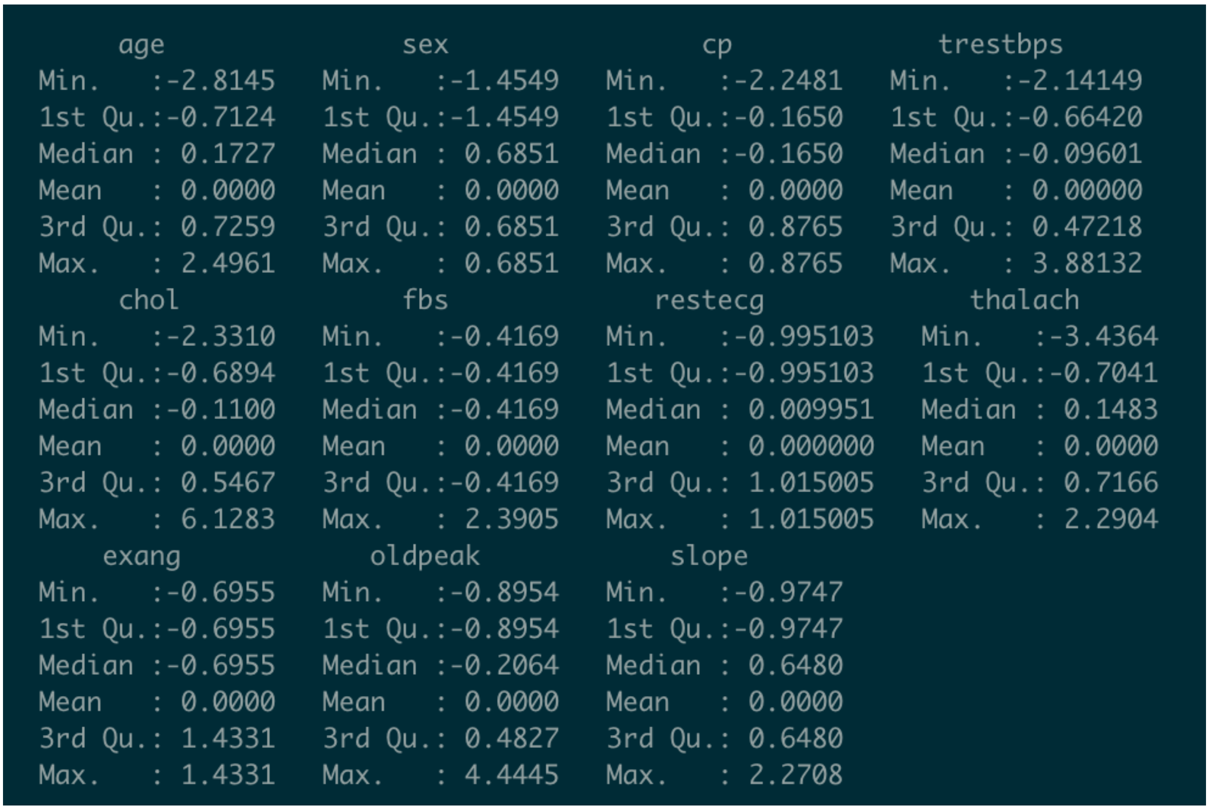
Exploratory data analysis helps us to understand the characteristics of the patients in the data. We need to get an idea of the value ranges of the variables and their distributions. This will also be helpful when we evaluate the clusters of patients from the algorithms.

~~~
```{r}
# Evidence that we should scale the data
summary(heart_disease)

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



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

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

# Let's see the data after we scaled it
summary(scaled)
```



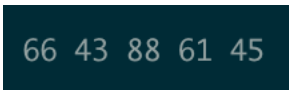
```
      age               sex                cp             trestbps       
 Min.   :-2.8145   Min.   :-1.4549   Min.   :-2.2481   Min.   :-2.14149  
 1st Qu.:-0.7124   1st Qu.:-1.4549   1st Qu.:-0.1650   1st Qu.:-0.66420  
 Median : 0.1727   Median : 0.6851   Median :-0.1650   Median :-0.09601  
 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  
```

Empirical Analysis

Now that we have scaled 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 essential to make sure that our results are reproducible when conducting a statistical analysis. This means that they will get the same results when someone runs our code on the same data. Because we are doing an analysis with a random aspect, it is necessary to set seed to ensure reproducibility. Reproducibility is essential because doctors will potentially use our results to treat patients. Other analysts must see where the groups come from and can verify the results.

```{r}
# Set the seed so that results are reproducible
set.seed(10)

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

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

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



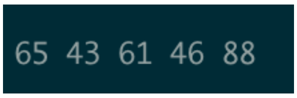
```
66 43 88 61 45
```

Because the k-means algorithm initially selects the cluster centers by randomly selecting points, different algorithm iterations can result in different clusters. If the algorithm genuinely groups similar observations (as opposed to clustering noise), then cluster assignments will be somewhat robust between various iterations of the algorithm.

Regarding 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, the clustering method does not pick 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.

~~~
```{r}
# Set the seed
set.seed(38)

# Select a number of clusters and runt the k-means algorithm
k <- 5
second_clust <- kmeans(scaled, centers = k, nstart = 1)

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



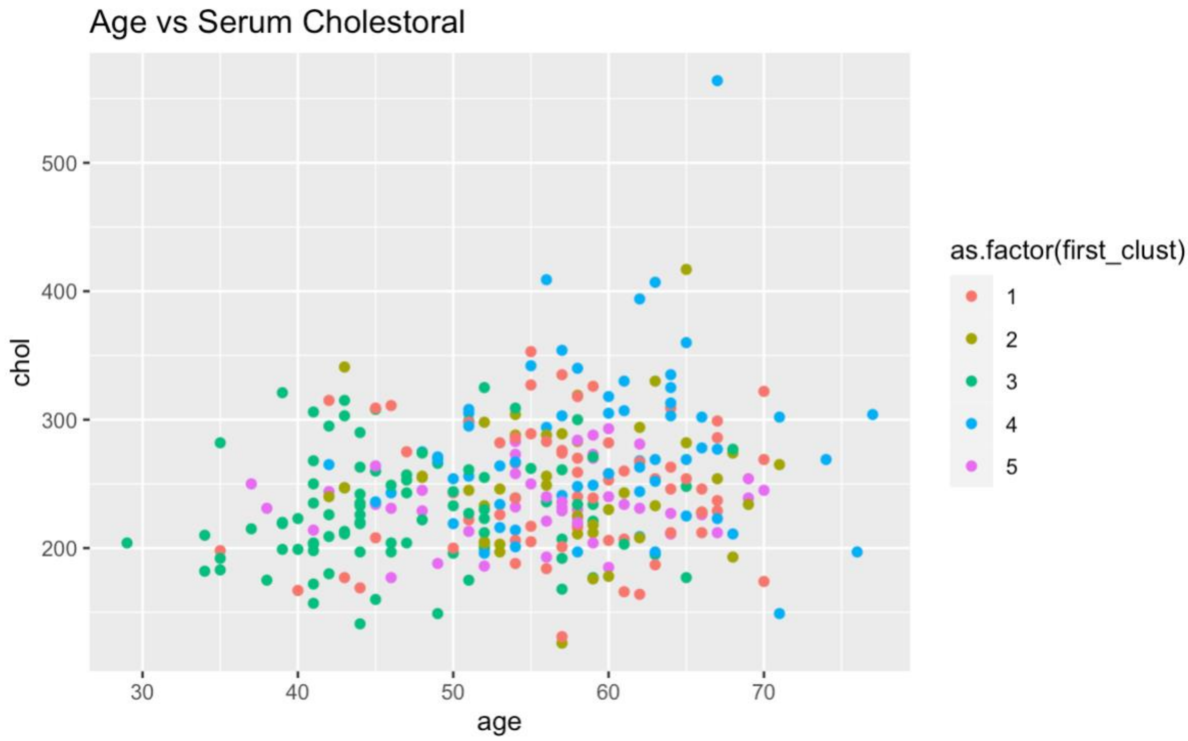
```
65 43 61 46 88
```

The clusters must be 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 initializations of the algorithm will result in similar clusters.

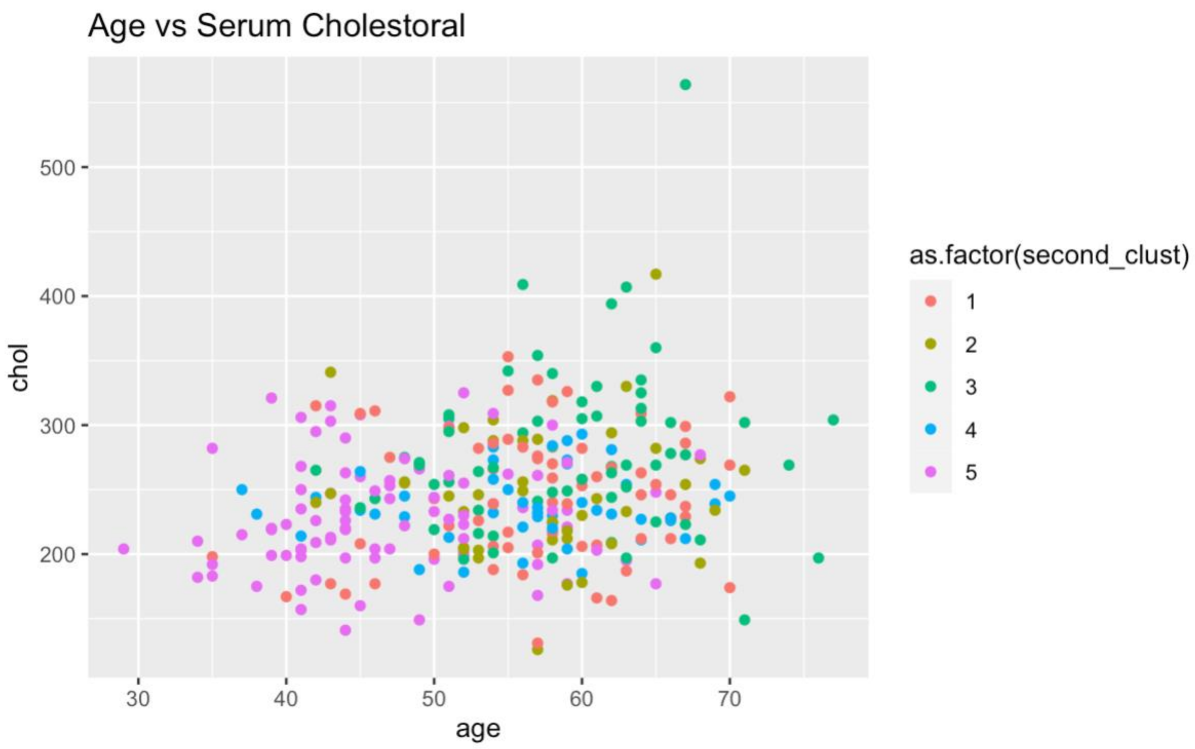
The clusters from different iterations may not be the same, but the clusters should be roughly the same size and have similar distributions of variables. If there is a lot of change in clusters between different algorithm iterations, then k-means clustering is not the right choice for the data.

It is impossible 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 specific patient characteristics may have been used to group patients together.

~~~
```{r}
# Add cluster assignments to the data
heart_disease["first_clust"] <- first_clust$cluster
heart_disease["second_clust"] <- second_clust$cluster

# Create and print the plot 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() + labs(title = "Age vs Serum Cholestoral")
plot_one
```
~~~



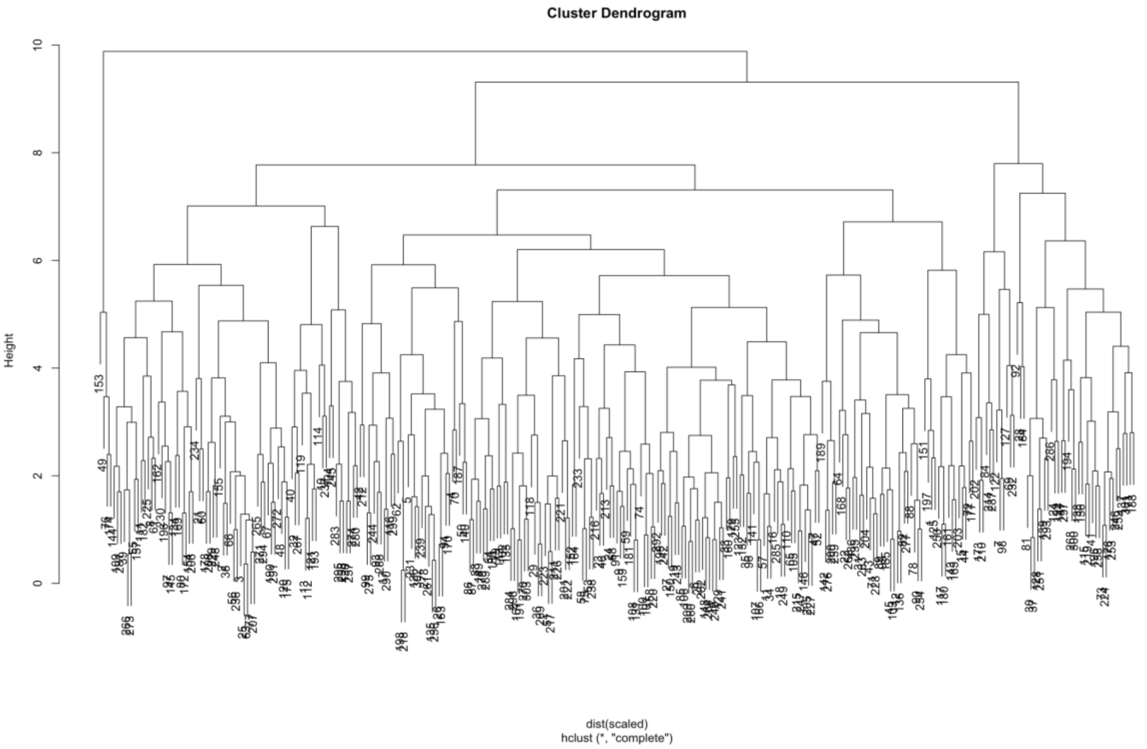
~~~
```{r}
# Create and print the plot of age and chol for the second clustering algorithm
plot_two <- ggplot(heart_disease, aes(x=age, y=chol, color=as.factor(second_clust))) +
  geom_point() + labs(title = "Age vs Serum Cholestoral")
plot_two
```
~~~



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.

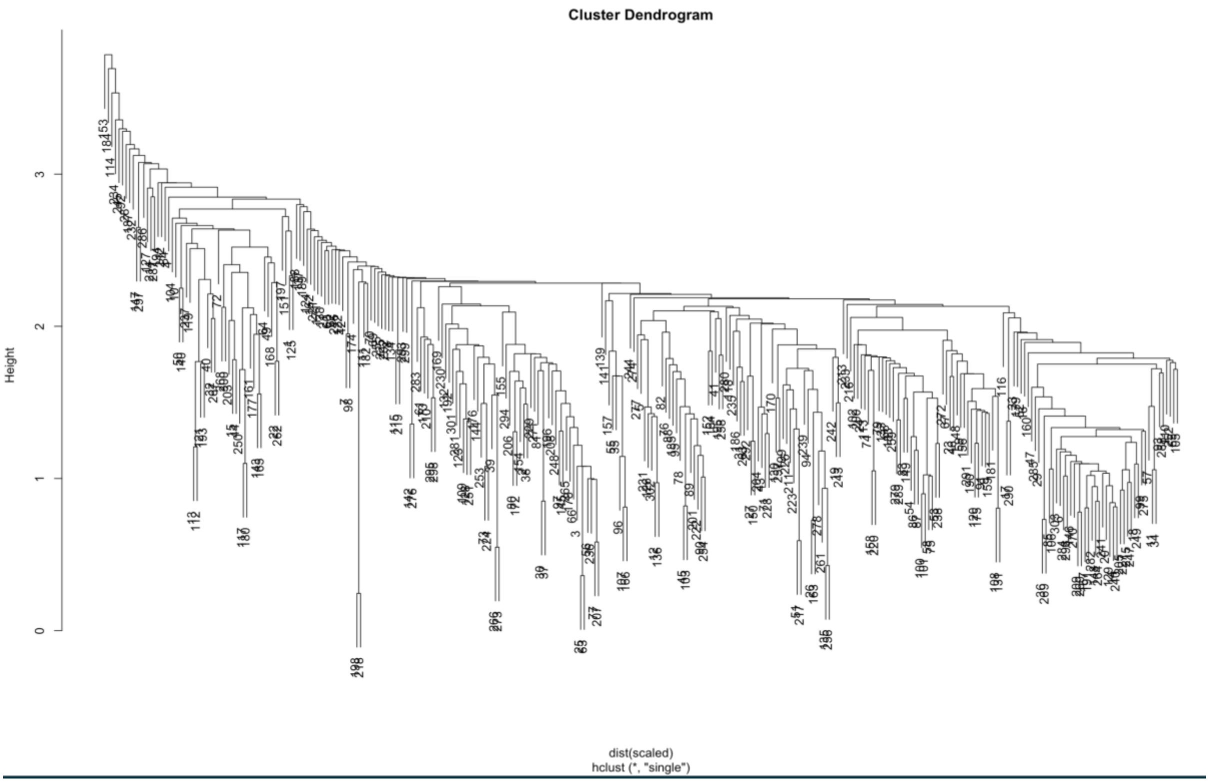
~~~
```{r}
# Execute hierarchical clustering with complete linkage
hier_clust_1 <- hclust(dist(scaled), method = "complete")

# Print the dendogram
plot(hier_clust_1)

# Get cluster assignments based on number os selecter cluster
hc_1_assign <- cutree(hier_clust_1, 5)
```
~~~



In hierarchical clustering, there are multiple ways to measure the dissimilarity between clusters of observations. Complete linkage records the largest distinction 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 clusters. It is always a good idea to explore different dissimilarity measures.

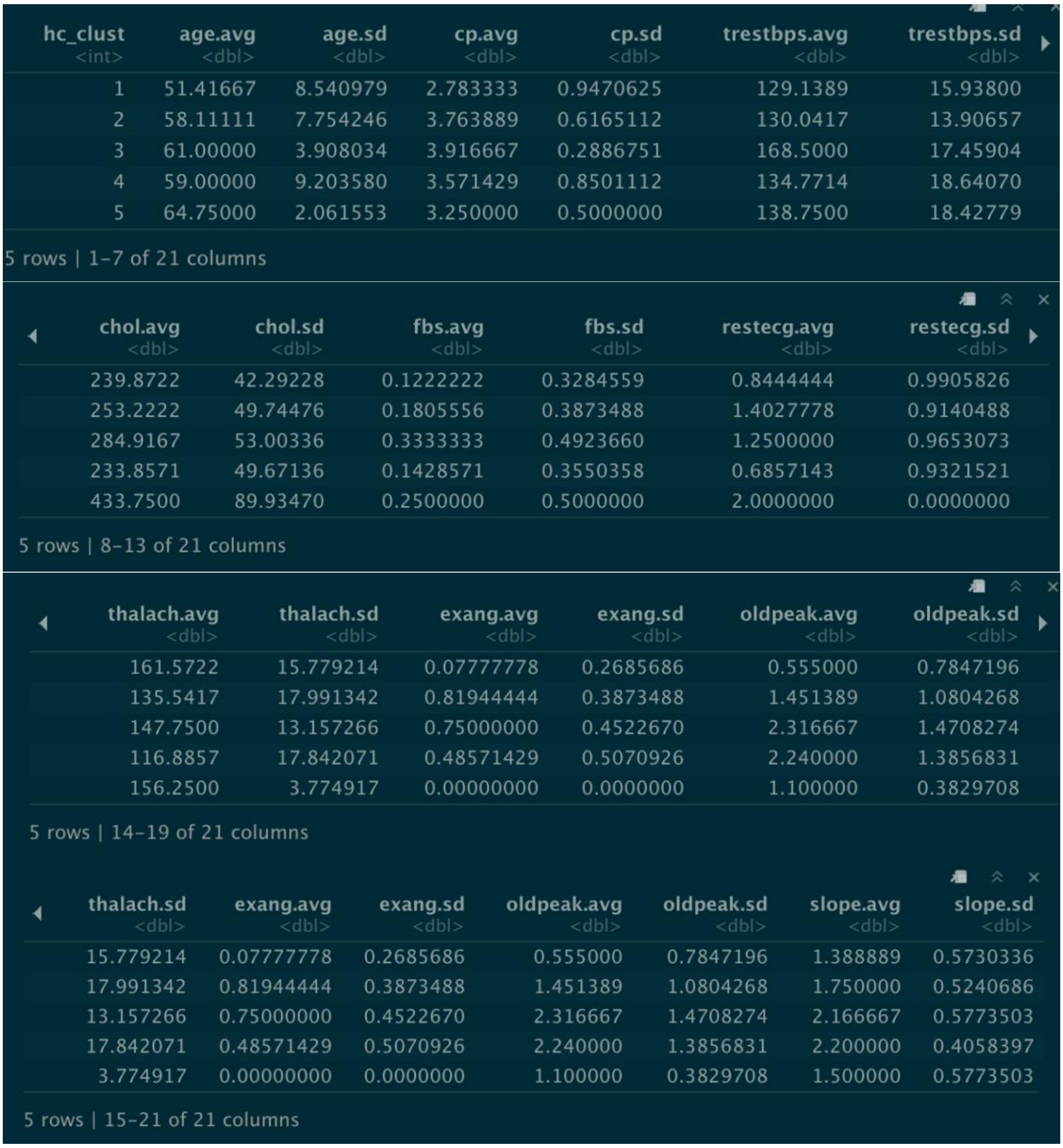
~~~
```{r}
# Execute hierarchical clustering with single linkage
hier_clust_2 <- hclust(dist(scaled), method = "single")

# Print the dendogram
plot(hier_clust_2)

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



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, 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. We're going to examine the clusters resulting from the two hierarchical algorithms.

```{r}
# Add assignment of chosen hierarchical linkage
heart_disease["hc_clust"] <- hc_1_assign

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

# 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
```



```
 hc_clust   age.avg    age.sd    cp.avg      cp.sd   trestbps.avg  trestbps.sd
    <int>     <dbl>     <dbl>     <dbl>      <dbl>          <dbl>        <dbl>
        1  51.41667  8.540979  2.783333  0.9470625       129.1389     15.93800
        2  58.11111  7.754246  3.763889  0.6165112       130.0417     13.90657
        3  61.00000  3.908034  3.916667  0.2886751       168.5000     17.45904
        4  59.00000  9.203580  3.571429  0.8501112       134.7714     18.64070
        5  64.75000  2.061553  3.250000  0.5000000       138.7500     18.42779
5 rows | 1-7 of 21 columns
```

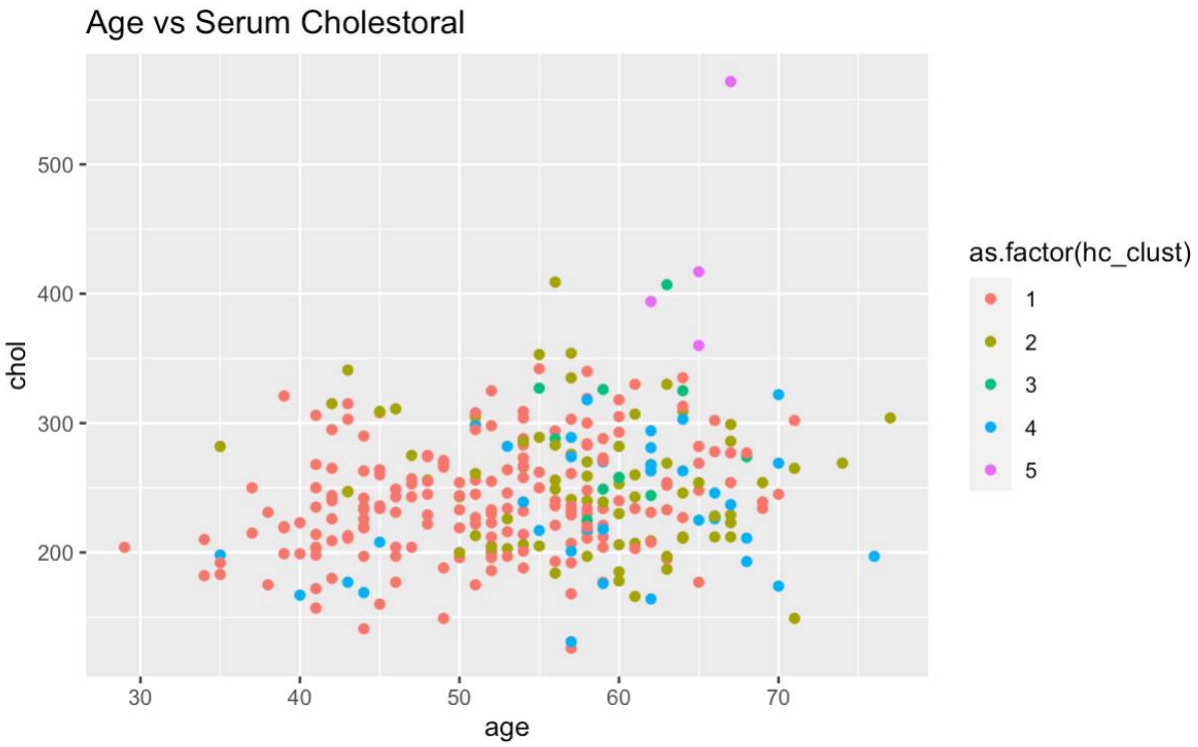
```
   chol.avg    chol.sd     fbs.avg      fbs.sd  restecg.avg  restecg.sd
      <dbl>      <dbl>       <dbl>       <dbl>        <dbl>       <dbl>
   239.8722   42.29228   0.1222222   0.3284559    0.8444444   0.9905826
   253.2222   49.74476   0.1805556   0.3873488    1.4027778   0.9140488
   284.9167   53.00336   0.3333333   0.4923660    1.2500000   0.9653073
   233.8571   49.67136   0.1428571   0.3550358    0.6857143   0.9321521
   433.7500   89.93470   0.2500000   0.5000000    2.0000000   0.0000000
5 rows | 8-13 of 21 columns
```

```
 thalach.avg  thalach.sd   exang.avg   exang.sd  oldpeak.avg  oldpeak.sd
       <dbl>       <dbl>       <dbl>      <dbl>        <dbl>       <dbl>
    161.5722   15.779214  0.07777778  0.2685686     0.555000   0.7847196
    135.5417   17.991342  0.81944444  0.3873488     1.451389   1.0804268
    147.7500   13.157266  0.75000000  0.4522670     2.316667   1.4708274
    116.8857   17.842071  0.48571429  0.5070926     2.240000   1.3856831
    156.2500    3.774917  0.00000000  0.0000000     1.100000   0.3829708
5 rows | 14-19 of 21 columns
```

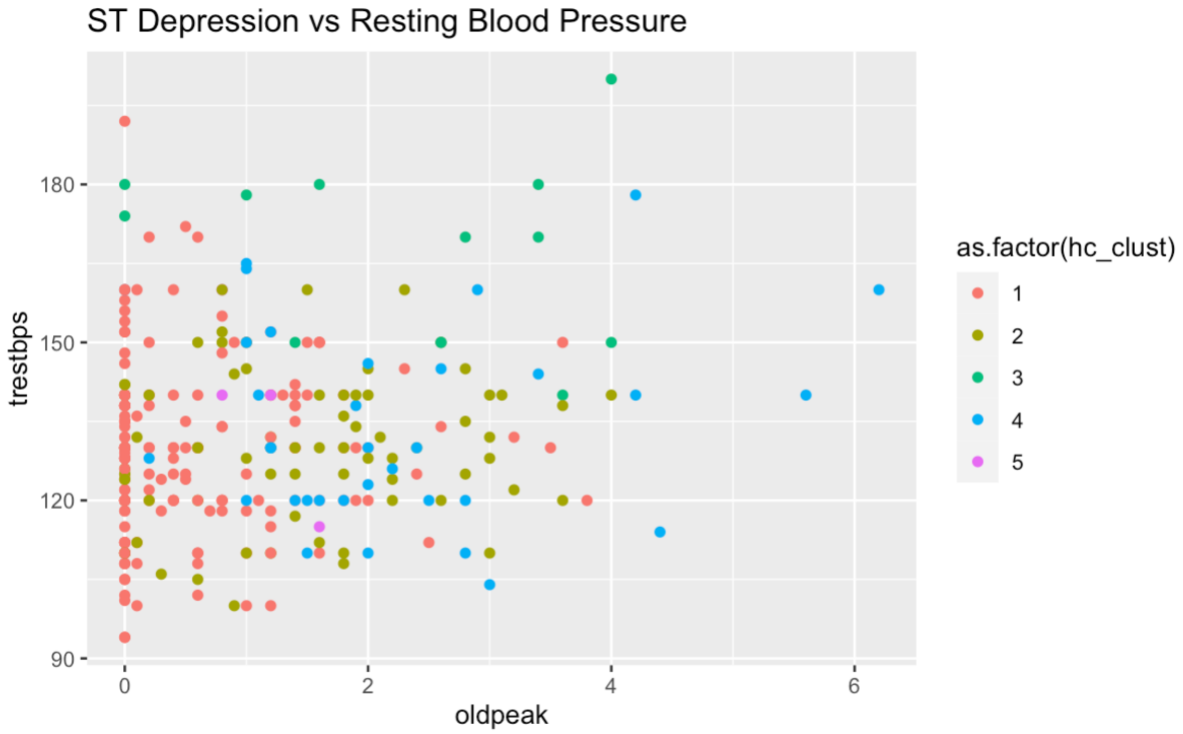
```
 thalach.sd   exang.avg   exang.sd  oldpeak.avg  oldpeak.sd  slope.avg   slope.sd
      <dbl>       <dbl>      <dbl>        <dbl>       <dbl>      <dbl>      <dbl>
  15.779214  0.07777778  0.2685686     0.555000   0.7847196   1.388889  0.5730336
  17.991342  0.81944444  0.3873488     1.451389   1.0804268   1.750000  0.5240686
  13.157266  0.75000000  0.4522670     2.316667   1.4708274   2.166667  0.5773503
  17.842071  0.48571429  0.5070926     2.240000   1.3856831   2.200000  0.4058397
   3.774917  0.00000000  0.0000000     1.100000   0.3829708   1.500000  0.5773503
5 rows | 15-21 of 21 columns
```

In addition to looking at the distributions of variables in each hierarchical clustering run, we will make visualizations to evaluate the algorithms. Even though the data has more than two dimensions, we can understand 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.

```{r}
# Plot age and chol
plot_one <- ggplot(heart_disease, aes(x=age, y = chol, color =as.factor(hc_clust))) +
geom_point() + labs(title = "Age vs Serum Cholestoral")
plot_one
```



~~~
```{r}
# Plot oldpeak and trestbps
plot_two <- ggplot(heart_disease, aes(x = oldpeak, y = trestbps, color = as.factor(hc_clust)))
+ geom_point() + labs(title = "ST Depression vs Resting Blood Pressure")
plot_two
```
~~~

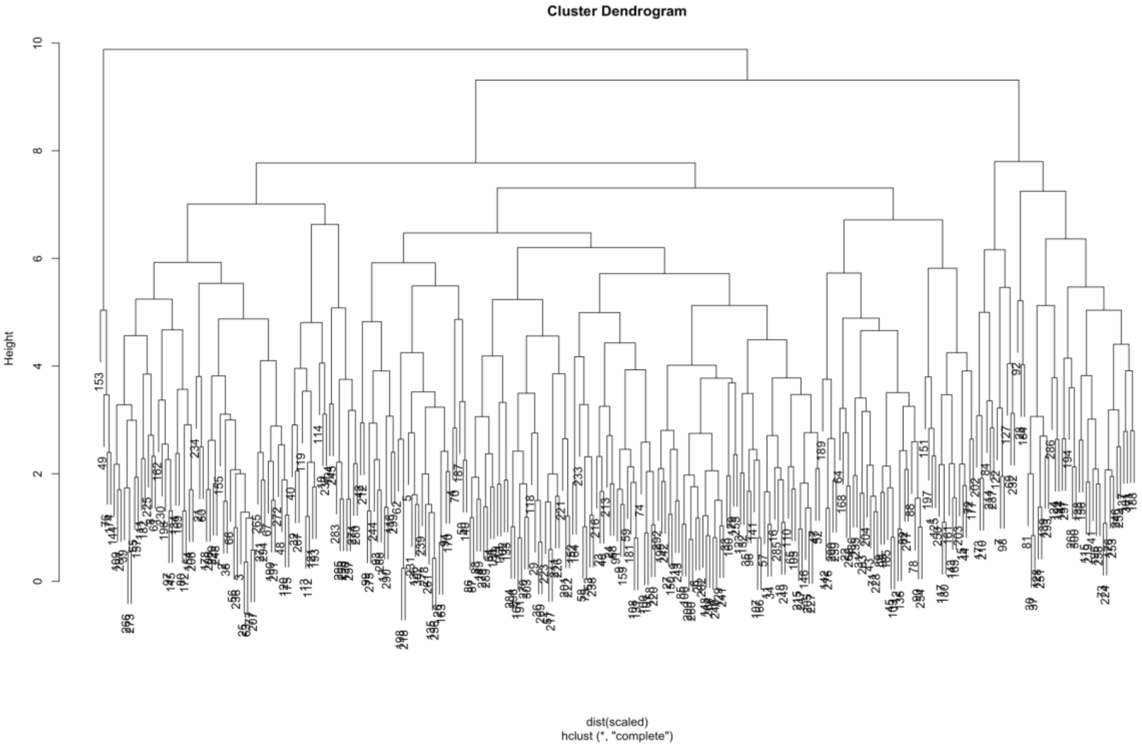


Conclusion

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 algorithm iteration to make sure that the algorithm clusters the signal, not the noise.

It’s also important for the k-mean algorithm to seem 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.

So the best algorithm that shows promise for this data is Hierarchical clustering with complete linkage.



Source:

- https://archive.ics.uci.edu/ml/datasets/heart+Disease