HarvardX: PH125.9x Data Science  
IDV: Hearth disease clustering

Tamás Kovács

May 30, 2019

# I. INTRODUCTION

Clustering algorithms are used to group items that are similar to one another. There are many industries where it would be beneficial. Retailers want to arrange same customers for targeted ad campaigns. However, physicians alsp often inquire about past cases to discover ways to treat their patients best. The medical record describes the systematic documentation of a patient’s medical history and cares across time. These records cover a variety of types of “notes” entered over time by doctors, logging observations and treatment of drugs and therapeutics, etc. Those patients who have comparable health records or symptoms to a former case could profit from the same treatment. This project examines whether physicians might be able to arrange patients mutually to target treatment using some conventional unsupervised learning methods.

The anonymized dataset of this project contains characteristics and measures of patients diagnosed with heart disease comes from the V.A. Medical Center, California.

This project is going to look at patients who have been diagnosed with heart disease, and it will use clustering methods using the k-means and hierarchical clustering algorithms, data visualization (ggplot2), and unsupervised learning. The k-means clustering is a way of vector quantization, recommended for cluster analysis in data mining. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster, while the hierarchical clustering (HCA) is a way of cluster analysis which seeks to establish a hierarchy of clusters.

Before the analysis of the project, let’s see how the patient data looks like.

heart\_dis = read.csv("datasets/heart.csv")  
  
# The first five rows of the data set  
head(heart\_dis, 5)

## 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  
## target  
## 1 1  
## 2 1  
## 3 1  
## 4 1  
## 5 1

# check that only numeric variables  
lapply(heart\_dis, class)

## $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"  
##   
## $ca  
## [1] "integer"  
##   
## $thal  
## [1] "integer"  
##   
## $target  
## [1] "integer"

# II. METHODS

## Quantifying of differences

It is necessary to carry out some exploratory analysis to familiarize ourselves with the data before clustering. Exploratory data analysis helps us to understand the characteristics of the patients in the data. Through this, we are getting an idea of the value ranges of the variables and their distributions.

It will help us see more about the variables and make a wise choice about whether we should scale the data or not. K-means and hierarchical is clustering measures similarity between points using a distance formula. It can place extra weight on specific variables that have a larger scale and thus, larger differences between points. This will be helpful when we decide the clusters of patients from the algorithms.

summary(heart\_dis)

## 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

# Remove id's  
heart\_dis = heart\_dis[ , !(names(heart\_dis) %in% c('id'))]  
  
# Scaling dataset to df  
scaled = scale(heart\_dis)  
summary(scaled)

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

## Grouping patients

Once we’ve decided if we need to change the data and create any significant changes, we can start the clustering process. For the k-means algorithm, it is crucial to decide the number of clusters.

It is also essential to secure that our results are reproducible when carrying a statistical analysis, so when someone runs our code on the same data, they will get the same results. Reproducibility is crucial because doctors will use our results to treat patients. Furthermore, another analyst also can recognize where the groups come from.

# Seed setting so the results are reproducible  
seed\_val = 10  
set.seed(seed\_val)  
  
# Cluster' numbers  
k = 5  
  
# Apply first k-means algorithm on 1st cluster  
cluster\_1 = kmeans(scaled, centers = k, nstart = 1)  
  
# Patients in each group  
cluster\_1$size

## [1] 50 57 33 69 94

Now, we will explore how the patients are grouping with another repetition of the k-means algorithm. The k-means algorithm chooses the cluster cores by randomly picking points; several repetitions of the algorithm can result in many clusters being created. If the algorithm is genuinely classifying similar observations, then cluster assignments will be slightly robust between different repetitions of the algorithm. In concern with the heart disease data, this would suggest that the same patients would be grouped even when the algorithm is initialized at various random points. If patients are not in comparable clusters with multiple algorithm runs, then the clustering method isn’t choosing significant relationships between patients.

# Seed setting  
seed\_val = 38  
set.seed(seed\_val)  
  
# Apply first k-means algorithm on 2nd cluster  
k = 5  
cluster\_2 = kmeans(scaled, k, nstart=1)  
  
# Patients in each group  
cluster\_2$size

## [1] 63 35 88 77 40

We can compare the resulting groups of patients.

## Analyze the 1st and 2nd cluster

The k-means algorithm produced clusters are stable. Even though the algorithm starts by randomly initializing the cluster centers, if the k-means is the best option for the data, then different initializations of the algorithm will end in similar clusters. The clusters from different repetitions may not be the same, but the clusters should be about the same size and have comparable arrangements of variables.

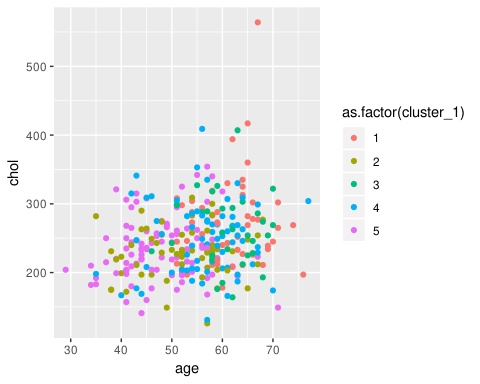
##Analyze patient's 1st and 2nd cluster  
# adding cluster assignments to the data  
heart\_dis['cluster\_1'] = cluster\_1$cluster  
heart\_dis['cluster\_2'] = cluster\_2$cluster

It is not reasonable to confirm that the clusters received from an algorithm are ground truth are reliable since there is no proper labeling for patients. Consequently, it is important to consider how the clusters develop between various repetitions of the algorithm. We will apply visualizations to obtain an impression of the cluster stabilities. We will see how specific patient characteristics may have been applied to collecting patients.

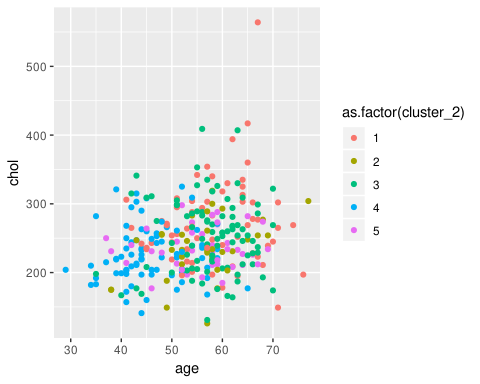
# Loading ggplot2 and   
library(ggplot2)

## Registered S3 methods overwritten by 'ggplot2':  
## method from   
## [.quosures rlang  
## c.quosures rlang  
## print.quosures rlang

# generate the plots of age and chol for the 1st clustering algorithm  
first\_plot = ggplot(heart\_dis, aes(x=age, y=chol, color=as.factor(cluster\_1))) + geom\_point()  
first\_plot



# generate the plots of age and chol for the 2nd clustering algorithm  
second\_plot = ggplot(heart\_dis, aes(x=age, y=chol, color=as.factor(cluster\_2))) + geom\_point()  
second\_plot

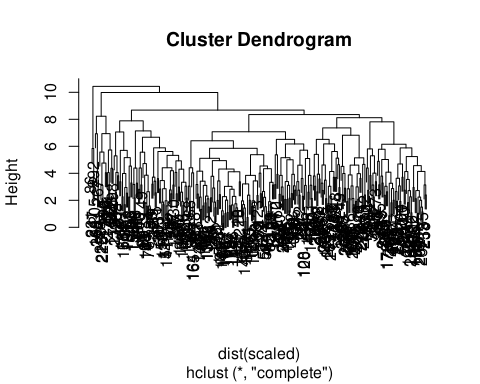


## Hierarchical clustering

Another option could be the hierarchical clustering, that works fine when the data has a nested structure. It is plausible that the data from heart disease patients follow this type of arrangement. For instance, if men are more likely to show specific characteristics, those features might be nested inside the gender variable. Hierarchical clustering also does not need the number of clusters to be chosen before running the algorithm.

The dendrogram enables us to understand how related observations are to one another and are valuable in picking the number of clusters to arrange the data. Now we are going to check how hierarchical clustering groups the data.

# creating hierarchical clustering with complete linkage  
hier\_clust\_1 = hclust(dist(scaled), method= 'complete')  
  
# generate dendrogram  
plot(hier\_clust\_1)

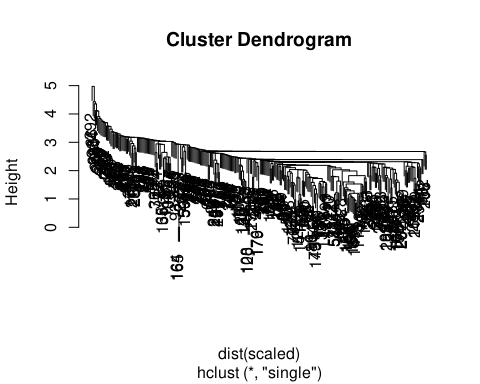


# creating cluster assignments based on the number of selected clusters  
hc\_1\_assign <- cutree(hier\_clust\_1, 5)

We want to investigate different algorithms to arrange our heart disease cases. The best way to cover dissimilarity among patients could be to study at the tiniest difference between patients and decrease that difference when grouping clusters. It makes sense to investigate various dissimilarity measures.

There are many ways to estimate the difference between clusters of observations in hierarchical clustering. Complete linkage reports the most considerable difference between any two points in the two clusters being compared. On the other hand, a single linkage is the smallest difference between any two points in the clusters. Let’s perform hierarchical clustering utilizing a new linkage function.

# creating hierarchical clustering with complete linkage  
hier\_clust\_2 = hclust(dist(scaled), method='single')  
  
# generate dendrogram  
plot(hier\_clust\_2)



# creating cluster assignments based on the number of selected clusters  
hc\_2\_assign <- cutree(hier\_clust\_2,5)

# III. RESULTS

As with the k-means, the way to assess the clusters is to examine which cases are being grouped. Are there patterns visible in the cluster distributions, or do they appear to be only noise?

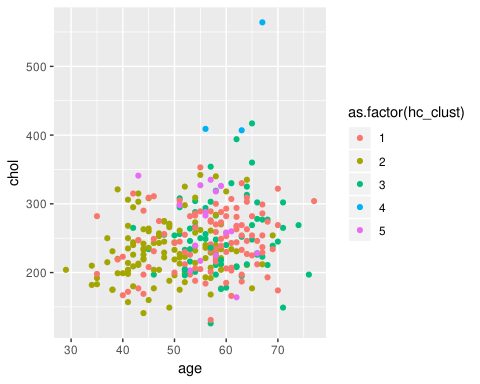
Physicians are engaged in comparable grouping patients to plan proper medicines and treatments. Consequently, they want to have clusters with more than a few cases to detect various therapies. It is reasonable for a patient to be in a cluster by themselves; this suggests that the medicine or therapy they got might not be prescribed for someone else in the group. We will explore the clusters emerging from the two hierarchical algorithms.

# adding assignments of chosen hierarchical linkage  
heart\_dis['hc\_clust'] = hc\_1\_assign  
  
# remove variables ('sex', 'cluster\_1', and 'cluster\_2')  
hd\_simple = heart\_dis[, !(names(heart\_dis) %in% c('sex', 'cluster\_1', 'cluster\_2'))]  
  
# generate mean and standard deviation summary statistics  
clust\_sum = do.call(data.frame, aggregate(. ~hc\_clust, data = hd\_simple, function(x) c(avg = mean(x), sd = sd(x))))  
clust\_sum

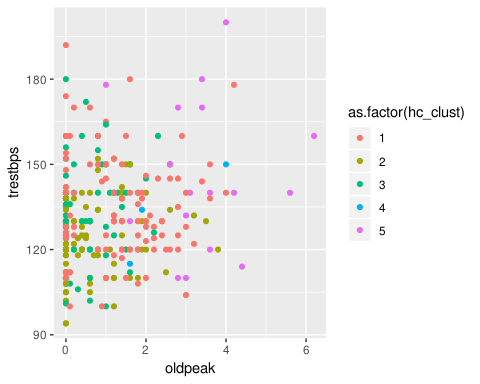
## hc\_clust age.avg age.sd cp.avg cp.sd trestbps.avg trestbps.sd  
## 1 1 56.89320 8.502494 0.4174757 0.9131316 134.5728 18.32849  
## 2 2 48.58407 8.011962 1.4778761 0.8973905 125.5487 12.59109  
## 3 3 59.26866 7.316611 1.2089552 0.9775869 133.5970 17.40956  
## 4 4 62.00000 5.567764 0.6666667 1.1547005 133.0000 17.52142  
## 5 5 56.82353 5.174883 0.0000000 0.0000000 146.1176 26.63147  
## chol.avg chol.sd fbs.avg fbs.sd restecg.avg restecg.sd  
## 1 246.4078 43.80201 0.14563107 0.3544608 0.4271845 0.5164346  
## 2 233.5310 42.10745 0.01769912 0.1324428 0.6814159 0.4680027  
## 3 252.1045 53.84690 0.32835821 0.4731602 0.4328358 0.5286885  
## 4 460.0000 90.07219 0.00000000 0.0000000 0.0000000 0.0000000  
## 5 269.2941 51.69353 0.35294118 0.4925922 0.5882353 0.7122871  
## thalach.avg thalach.sd exang.avg exang.sd oldpeak.avg oldpeak.sd  
## 1 135.6505 23.036079 0.6116505 0.4897580 1.3611650 1.0950184  
## 2 163.8938 15.640699 0.1327434 0.3408085 0.6185841 0.8326537  
## 3 149.9701 19.986336 0.1044776 0.3081877 0.5835821 0.6756808  
## 4 154.6667 5.033223 0.3333333 0.5773503 2.5000000 1.3076697  
## 5 137.5882 17.381955 0.7647059 0.4372373 3.4294118 1.2628166  
## slope.avg slope.sd ca.avg ca.sd thal.avg thal.sd target.avg  
## 1 1.2038835 0.5306161 1.0485437 1.0134260 2.572816 0.6199611 0.1165049  
## 2 1.5663717 0.5957604 0.4159292 0.9794830 2.230088 0.4819760 0.8407080  
## 3 1.6567164 0.5090764 0.5820896 0.8375515 1.940299 0.5471856 0.8507463  
## 4 1.0000000 0.0000000 1.6666667 1.5275252 3.000000 0.0000000 0.3333333  
## 5 0.5294118 0.5144958 1.2941176 1.1599949 2.647059 0.7018882 0.0000000  
## target.sd  
## 1 0.3223982  
## 2 0.3675783  
## 3 0.3590278  
## 4 0.5773503  
## 5 0.0000000

In addition to studying at the distributions of variables in each of hierarchical clustering, we will create visualizations to assess the algorithms, therefore, we can receive an impression of how the data clusters by looking at a scatterplot of two variables. We want to understand what patients get clustered together.

# age and chol  
first\_plot = ggplot(hd\_simple, aes(x=age, y=chol, color=as.factor(hc\_clust))) + geom\_point()  
first\_plot



# oldpeak and trestbps  
second\_plot = ggplot(hd\_simple, aes(oldpeak, trestbps, color=as.factor(hc\_clust))) + geom\_point()  
second\_plot



# IV. CONCLUSION

During the project, we’ve tried out various clustering algorithms, it is essential to decide if we recall any of them will run for clustering our cases. For the k-means, it is powerful that comparable clusters are provided for each repetition of the algorithm. However, we need to be sure that the algorithm is clustering signal as exposed to noise.

For the interest of the physicians, we also need to have many patients in each group so they can contrast therapies. We did a few preparatory works to investigate the execution of the algorithms. It is important to build more visualizations and examine how the algorithms group additional variables.

explore\_kmeans = F  
explore\_hierarch\_complete = T  
explore\_hierarch\_single = F