Lab 8: Clustering Methods

PSTAT 131/231, Fall 2021

Learning Objectives

- k-means clustering - `kmeans()` function - Pros and cons
- Hierachical clustering - `hclust()` function - Different linkages (complete, single, average) - Make a dendrogram - Plotting Dendrogram using `dendextend`

---

1. Pre-processing of data

We will look at cluster analysis performed on a set of cars from 1978-1979. The dataset includes gas mileage (coded as mpg), engine horsepower (horsepower), and other information for 38 vehicles. It is similar to the Auto dataset from ISLR, but has fewer observations and features.

In R, a number of cluster analysis algorithms are available through the library `cluster`, providing us with a large selection of methods to perform cluster analysis.

Load the package `cluster` and look at the first three observations:

```
# install.packages("cluster")
library(cluster)
library(tidyverse)

# Read tab-delimited file by read.delim
car = read_delim("./cars.tab", delim = "\t")

# First 3 observations
head(car, 3)
```

```
## # A tibble: 3 x 8
##   Country Car           MPG Weight Drive_Ratio Horsepower Displacement Cylinders
##   <chr>   <chr>       <dbl>  <dbl>       <dbl>      <dbl>        <dbl>     <dbl>
## 1 U.S.    Buick Esta~  16.9   4.36        2.73        155          350         8
## 2 U.S.    Ford Count~  15.5   4.05        2.26        142          351         8
## 3 U.S.    Chevy Mali~  19.2   3.60        2.56        125          267         8
```

The numerical variables (`MPG`, `Weight`, `Drive_Ratio`, `Horsepower`, `Displacement` and `Cylinders`) are measured on different scales, therefore we want to standardize the data before proceeding. There are multiple methods to re-scale the variables, we do it by subtracting the mean and dividing the standard deviation from each feature.

`scale()` function helps with standardization. It has two more arguments other than passing in the data matrix. By `center=TRUE`, centering is done by subtracting the column means (omitting NAs); by `scale=TRUE`, scaling is done by dividing the (centered) columns by their standard deviations.

```
# Standardize the variables by subtracting mean and divided by standard deviation
scar = scale(car[, -c(1,2)], center=TRUE, scale=TRUE)
```

Note that we didn't standardize predictors `Country` and `Car`, which are both strings.

2. Clustering functions

(a) k-means clustering

k-means clustering is available in R through the `kmeans()` function. The first step (and certainly not a trivial one) when using k-means cluster analysis is to specify the number of clusters (k) that will be formed in the final solution.

Note: Recall from the lecture that the idea behind k-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible. The most common choice for within-cluster variation measure is the squared euclidean distance.

```
set.seed(1)
# 3-means clustering
km = kmeans(scar, centers=3)
km
```

```
## K-means clustering with 3 clusters of sizes 8, 13, 17
##
## Cluster means:
##          MPG     Weight Drive_Ratio Horsepower Displacement  Cylinders
## 1 -1.1203872  1.4864539  -1.2960719  1.3712707   1.66759621  1.6252130
## 2 -0.5602631  0.2238870   0.2578965  0.3473723  -0.02489459  0.1376443
## 3  0.9556775 -0.8707154   0.4127012 -0.9109415  -0.76571412 -0.8700635
##
## Clustering vector:
##  [1] 1 1 1 1 3 3 3 3 2 2 2 2 2 2 2 2 1 1 1 1 3 2 3 3 3 3 3 3 2 2 3 3 3 3 3 2 2 3
##
## Within cluster sum of squares by cluster:
## [1]  3.74590 24.08477 18.91804
##  (between_SS / total_SS =  78.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Looking on the good side, the k-means technique is fast, and doesn't require calculating all of the distances between each observation and every other observation. It can be written to efficiently deal with very large data sets, so it may be useful in cases where other methods fail.

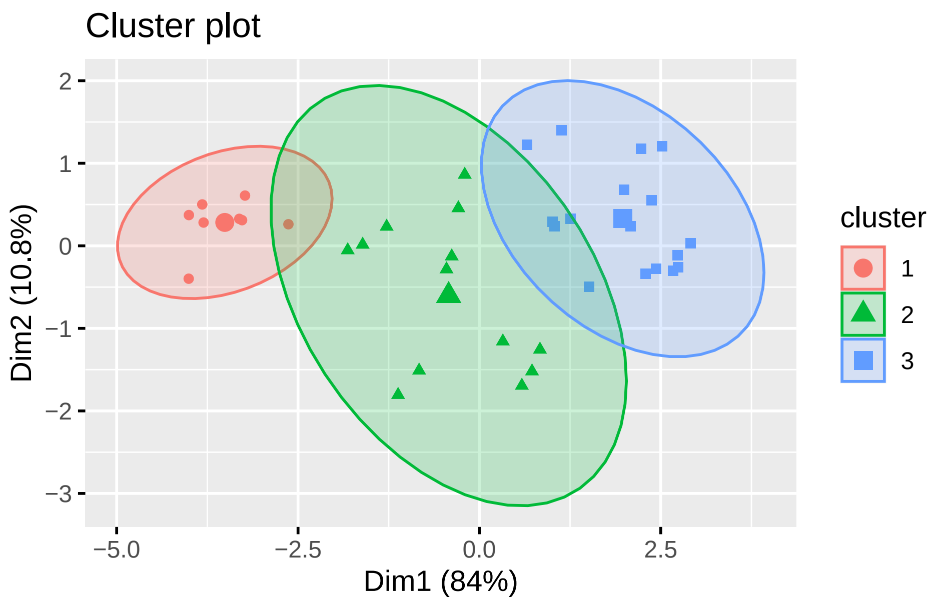
On the down side, we recall from the lecture that the final result from runing k-means largely depend on the initial cluster assignments. In other words, if we rearrange our data, it's very possible that we'll get a different solution every time we change the ordering of the data. Another criticism of the k-means is that we need to pre-specify the value of k. For example, a 3 cluster solution that seems to work pretty well, but when we look for the 4 cluster solution, all of the structure that the 3 cluster solution revealed is gone. This makes the procedure somewhat unattractive if we don't know exactly how many clusters we should have in the first place.

- **k-means clustering via `kmeans()`**

We can visualize k-means clusters by `fviz_cluster()` in package `factoextra`.

```
# install.packages("factoextra")
library(factoextra)
```

```r
fviz_cluster(km, data = scar, geom = "point",
             stand = FALSE, frame.type = "norm")
```



(b) Hierarchical Clustering

Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of *k*. Hierarchical clustering has an added advantage over k-means clustering in that it results in an attractive tree-based representation of the observations, called a dendrogram.

To perform hierarchical clustering, we can use function `hclust()` by passing the distance matrix into it. By default, `hclust()` uses complete linkage (defaulting `method="complete"`). We can use single linkage (by specifying `method="single"`) and average linkage (`method="average"`) too. The `cluster` library provides a similar function, called `agnes` to perform hierarchical cluster analysis. Naturally, the first step is calculating a distance matrix. Again, there are two functions that can be used to calculate distance matrices, `dist()` and `daisy()`. The Hierarchical clustering is shown below.

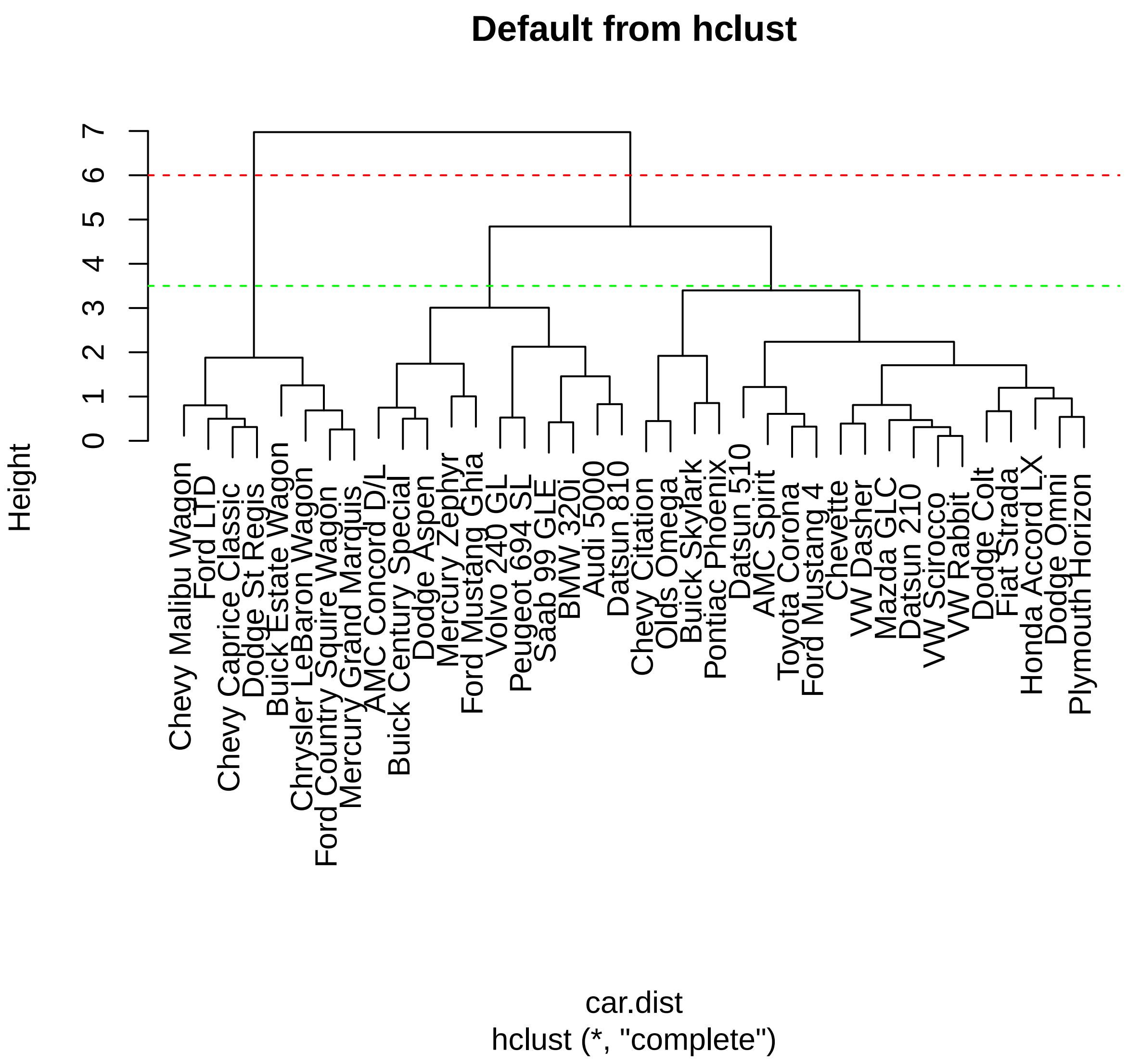
```r
# We use the dist function in this example.
car.dist = dist(scar)
# Can also do: car.dist = daisy(scar)

# Agglomerative Hierarchical clutering
set.seed(1)
car.hclust = hclust(car.dist) # complete linkage
# Can also do: car.hclust = agnes(car.dist)
```

Note: We're using the default method of `hclust()`, which is to update the distance matrix using what the "complete" linkage. Using this method, when a cluster is formed, its distance to other objects is computed as the maximum distance between any object in the cluster and the other object. Other linkage methods will provide different solutions, and should not be ignored. Check them!

Now that we've got a cluster solution (actually a sequence of cluster assignments), the main graphical tool for looking at a hierarchical cluster solution is known as a dendrogram. This is a tree-like display that lists the objects which are clustered along the x-axis, and the distance at which the cluster was formed along the y-axis. (Distances along the x-axis are meaningless in a dendrogram; the observations are equally spaced to make the dendrogram easier to read.) To create a dendrogram from a cluster solution, simply pass it to the plot() function. The dendrogram can be generated as below.

```r
# Plot dendogram
plot(car.hclust, labels=car$Car, main='Default from hclust')
# Add a horizontal line at a certain height
abline(h=6, col="red", lty=2)
abline(h=3.5, col="green", lty=2)
```



If we choose any height along the y-axis of the dendrogram, and move across the dendrogram counting the number of lines that we cross, each line represents a group that was identified when objects were joined together into clusters. The observations in that group are represented by the branches of the dendrogram that spread out below the line. For example, the red line gives us a 2-cluster solution, and the green line gives a 3-cluster result.

Looking at the dendrogram for the car data, there are clearly two very distinct groups; the right hand group seems to consist of two more distinct cluster, while most of the observations in the left hand group are clustering together at about the same height. It looks like either two or three groups might be an interesting place to start investigating.

One of the first things we can look at is how many cars are in each of the groups. We'd like to do this for both the 2-cluster and 3-cluster solutions. We can create a vector showing the cluster membership of each observation by using the `cutree()` function. Since the object returned by a hierarchical cluster analysis contains information about solutions with different numbers of clusters, we pass the `cutree()` function the cluster object and the number of clusters we're interested in.

```
clus = cutree(car.hclust, 3)
```

The output `clus` simply gives the cluster assignments to each observations:

```
clus
```

```
##  [1] 1 1 1 1 2 2 2 2 3 3 3 3 3 3 3 3 1 1 1 1 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 2
```

A good check is to use the table function to see how many observations are in each cluster. We'd like a solution where there aren't too many clusters with just a few observations, because it may make it difficult to interpret our results. For the three cluster solution, the distribution among the clusters looks good:

```
table(clus)
```

```
## clus
##  1  2  3
##  8 19 11
```

3. Dendrogram by `dendextend`

As we can see from the above results of cluster dendrogram, when the number of observations is large, the dendrogram looks messy, making it less interpretable. Here we introduce another R package called `dendextend`. First of all, the function `as.dendrogram()` is used to produce tree-like structures. Next, functions `color_branches()` and `color_labels()` can color all branches and labels by clusters. For example,

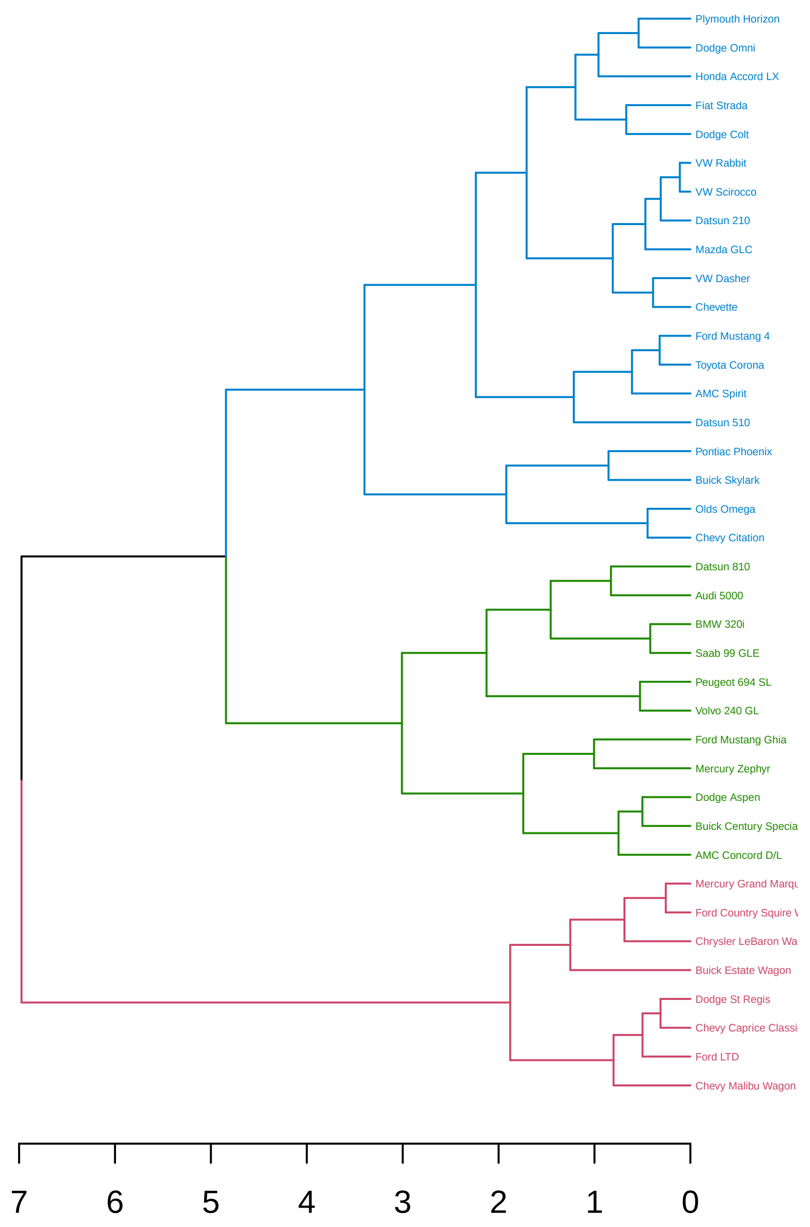
```
# install.packages("dendextend")
library(dendextend)

## dendrogram: branches colored by 3 groups
dend1 = as.dendrogram(car.hclust)
# color branches and labels by 3 clusters
dend1 = color_branches(dend1, k=3)
dend1 = color_labels(dend1, k=3)
# change label size
dend1 = set(dend1, "labels_cex", 0.3)
```

In order to make the dendrogram more readable, we would like to rotate it counter-clockwise, making the tips shown on the right.

```
# add true labels to observations
dend1 = set_labels(dend1, labels=car$Car[order.dendrogram(dend1)])
# plot the dendrogram
plot(dend1, horiz=T, main = "Dendrogram colored by three clusters")
```

Dendrogram colored by three clusters



Your turn

```r
# Perform a hierachical clustering (based on dis) using single linkage,
# and save this hclust object as "s.hclust"

# s.hclust = ?


# Compare the results of above hierachical clustering with car.hclust by a simple table
```

Credit: Adopted from http://www.stat.berkeley.edu/classes/s133/Cluster2a.html