Clustering

Dr. Marko Mitic

Problem 1: **DOCUMENT CLUSTERING WITH DAILY KOS**

Document clustering, or text clustering, is a very popular application of clustering algorithms. A web search engine, like Google, often returns thousands of results for a simple query. For example, if you type the search term "jaguar" into Google, around 200 million results are returned. This makes it very difficult to browse or find relevant information, especially if the search term has multiple meanings. If we search for "jaguar", we might be looking for information about the animal, the car, or the Jacksonville Jaguars football team.

Clustering methods can be used to automatically group search results into categories, making it easier to find relavent results. This method is used in the search engines PolyMeta and Helioid, as well as on FirstGov.gov, the official Web portal for the U.S. government. The two most common algorithms used for document clustering are Hierarchical and k-means.

In this problem, we'll be clustering articles published on Daily Kos https://www.dailykos.com/, an American political blog that publishes news and opinion articles written from a progressive point of view. Daily Kos was founded by Markos Moulitsas in 2002, and as of September 2014, the site had an average weekday traffic of hundreds of thousands of visits.

The file dailykos.csv contains data on 3,430 news articles or blogs that have been posted on Daily Kos. These articles were posted in 2004, leading up to the United States Presidential Election. The leading candidates were incumbent President George W. Bush (republican) and John Kerry (democratic). Foreign policy was a dominant topic of the election, specifically, the 2003 invasion of Iraq.

Each of the variables in the dataset is a word that has appeared in at least 50 different articles (1,545 words in total). The set of words has been trimmed according to some of the techniques covered in the previous week on text analytics (punctuation has been removed, and stop words have been removed). For each document, the variable values are the number of times that word appeared in the document.

Let's start by building a hierarchical clustering model.

```r
dailykos = read.csv("dailykos.csv")
str(dailykos)
```

```
## 'data.frame':    3430 obs. of  1545 variables:
##  $ abandon                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ abc                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ability                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ abortion                            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ absolute                            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ abstain                             : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ abu                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ abuse                               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ accept                              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ access                              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ accomplish                          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ account                             : int  0 0 2 0 0 0 0 0 0 0 ...
##  $ accurate                            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ accusations                         : int  0 0 0 2 0 0 0 0 0 0 ...
##  $ achieve                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ acknowledge                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ act                                 : int  0 0 0 0 0 0 0 0 0 0 ...
```

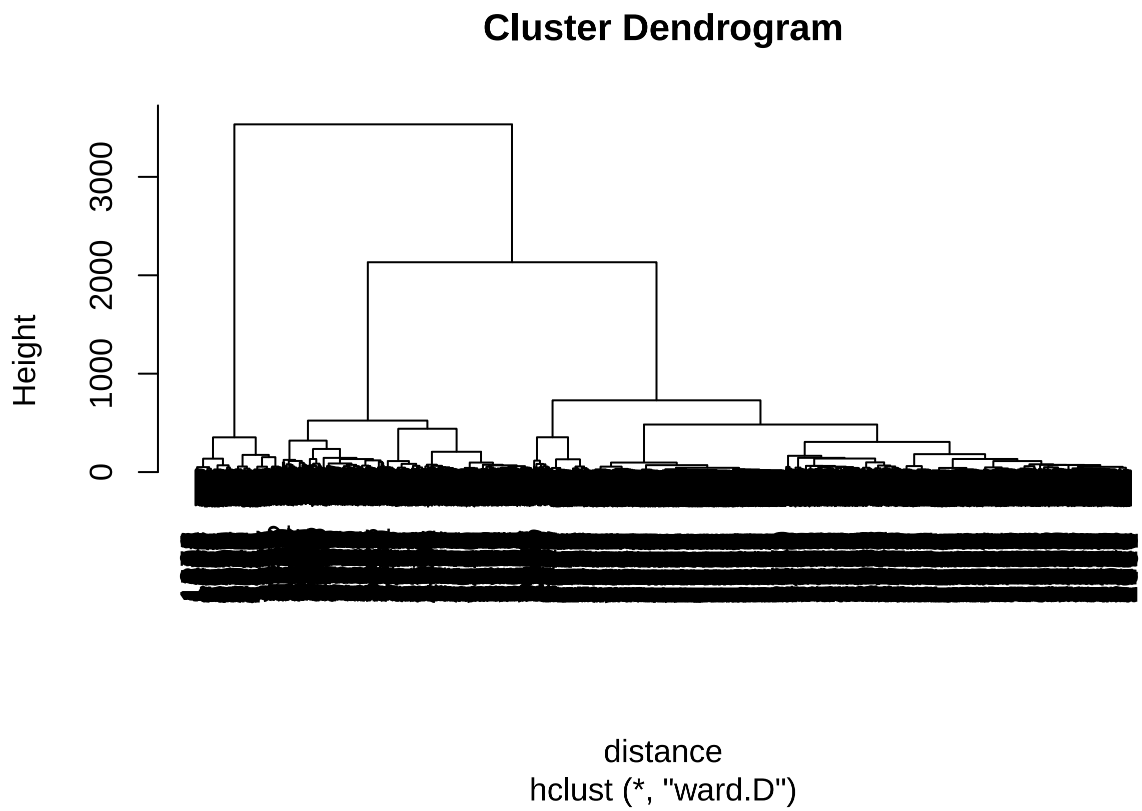
```
##  $ action                                  : int  2 0 0 0 0 0 0 0 0 0 ...
##  $ active                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ activist                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ actual                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ add                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ added                                   : int  1 0 0 0 1 0 0 0 1 0 ...
##  $ addition                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ address                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ admin                                   : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ administration                          : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ admit                                   : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ advance                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ advantage                               : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ advertise                               : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ advised                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ affair                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ affect                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ affiliate                               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ afghanistan                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ afraid                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ afternoon                               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ agencies                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ agenda                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ agree                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ahead                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ aid                                     : int  0 0 0 1 1 0 0 0 0 0 ...
##  $ aim                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ air                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ alaska                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ allegation                              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ allegory                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ allied                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ allowed                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ alternative                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ altsite                                 : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ amazing                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ amendment                               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ america                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ american                                : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ amount                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ amp                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ analysis                                : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ analyst                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ anecdotal                               : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ anger                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ angry                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ announce                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ annual                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ answer                                  : int  0 0 0 1 0 0 1 0 0 0 ...
##  $ apologies                               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ apparent                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ appeal                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ appearance                              : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ applied                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ appointed                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ approach                                 : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ approval                                 : int  1 0 0 0 1 0 0 0 1 0 ...
##  $ apr                                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ april                                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ arab                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ area                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ arent                                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ arg                                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ argue                                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ argument                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ arizona                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ arm                                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ armstrong                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ army                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ arrest                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ arrive                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ article                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ asap                                     : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ asked                                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ass                                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ assess                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ assist                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ associate                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ assume                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ atlanta                                  : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ atrios                                   : int  0 0 0 0 0 0 1 0 0 0 ...
##   [list output truncated]
```

```r
distance = dist(dailykos, method = "euclidean")
cluster = hclust(distance, method = "ward.D")
```

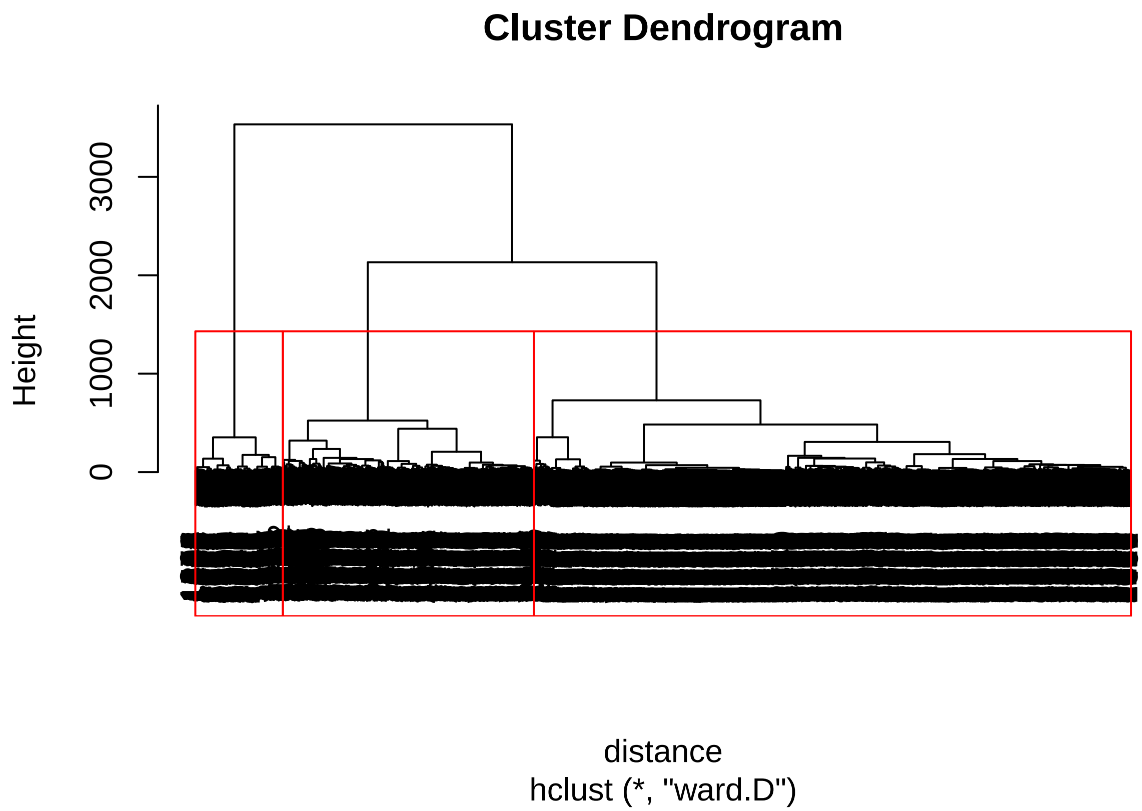
The computation may take some time, since we have lots of observations and variables in the dataset. Let us next plot the dendogram:

```r
plot(cluster)
```



The choices 2 and 3 are good cluster choices according to the dendrogram, because there is a lot of space between the horizontal lines in the dendrogram in those cut off spots (draw a horizontal line across the dendrogram where it crosses 2 or 3 vertical lines). This can be shown by using `rect.hist` function for drawing cluster:

```
plot(cluster)
rect.hclust(cluster, k=3, border="red")
```



However, just thinking about the application, it is probably better to show the reader more categories than 2 or 3. These categories would probably be too broad to be useful. Seven or eight categories seems more reasonable. Let us next subset each of the seven clusters:

```
clusterGroups =cutree(cluster, k = 7)
cluster1 = subset(dailykos, clusterGroups == 1)
cluster2 = subset(dailykos, clusterGroups == 2)
cluster3 = subset(dailykos, clusterGroups == 3)
cluster4 = subset(dailykos, clusterGroups == 4)
cluster5 = subset(dailykos, clusterGroups == 5)
cluster6 = subset(dailykos, clusterGroups == 6)
cluster7 = subset(dailykos, clusterGroups == 7)
```

By using `str` function we observe that cluster1 contains most observations, while cluster4 has lowest number of them. We can also see the frequency for each varaible in each cluster. Combination of `tail`, `sort` and `colMeans` computes the mean frequency values of each of the words in cluster, and then outputs the 6 words that occur the most frequently. The colMeans function computes the column (word) means, the sort function orders the words in increasing order of the mean values, and the tail function outputs the last 6 words listed, which are the ones with the largest column means.

```
tail(sort(colMeans(cluster1)))
```

```
##      state republican       poll   democrat      kerry       bush
##  0.7575039  0.7590837  0.9036335  0.9194313  1.0624013  1.7053712
```

We observe that the word “bush” is most frequent word in this cluster. For cluster 2 this are words “november” and “poll”.

```r
tail(sort(colMeans(cluster2)))
```

```
##      bush  democrat challenge      vote      poll  november
##  2.847352  2.850467  4.096573  4.398754  4.847352 10.339564
```

Next, we can run k-means algorithm, to find new patterns.

```r
k = 7 #seven clusters
set.seed(1000)
KMC = kmeans(dailykos, centers = k)
```

We now subset the `KMC`, as in hierarchical clustering:

```r
dailykosClusters = KMC$cluster
```

The number observations in each cluster can be determined using sum function:

```r
sum(dailykosClusters==1)
```

```
## [1] 146
```

```r
sum(dailykosClusters==2)
```

```
## [1] 144
```

```r
sum(dailykosClusters==3)
```

```
## [1] 277
```

```r
sum(dailykosClusters==4)
```

```
## [1] 2063
```

```r
sum(dailykosClusters==5)
```

```
## [1] 163
```

```r
sum(dailykosClusters==6)
```

```
## [1] 329
```

```r
sum(dailykosClusters==7)
```

```
## [1] 308
```

```r
# or using:
KmeansCluster = split(dailykos, dailykosClusters)
#str(KmeansCluster)
```

It can be observed that cluster 4 and cluster 2 have largest and smallest number of observations. This is, of course, different comparing hierarchical clustering case. Most frequent terms can also be obtained using cobination of `tail`, `sort` and `colMeans`:

```r
KmeansCluster1 = subset(dailykos, KMC$cluster == 1)
KmeansCluster2 = subset(dailykos, KMC$cluster == 2)
KmeansCluster3 = subset(dailykos, KMC$cluster == 3)
KmeansCluster4 = subset(dailykos, KMC$cluster == 4)
KmeansCluster5 = subset(dailykos, KMC$cluster == 5)
KmeansCluster6 = subset(dailykos, KMC$cluster == 6)
KmeansCluster7 = subset(dailykos, KMC$cluster == 7)
tail(sort(colMeans(KmeansCluster1)))
```

```
##          state           iraq          kerry administration       presided
##       1.609589       1.616438       1.636986       2.664384       2.767123
##           bush
##      11.431507
```

Comparing these results with hierarchical clustering, we can determine the similarity of each cluster. For example, using the `table` function, we observe that the hierarchical cluster 7 is most similar to K-means cluster 2:

```r
table(clusterGroups, KMC$cluster)
```

```
##              
## clusterGroups    1    2    3    4    5    6    7
##             1    3   11   64 1045   32    0  111
##             2    0    0    0    0    0  320    1
##             3   85   10   42   79  126    8   24
##             4   10    5    0    0    1    0  123
##             5   48    0  171  145    3    1   39
##             6    0    2    0  712    0    0    0
##             7    0  116    0   82    1    0   10
```

Similarly, it is interesting to note that K-means cluster 6 is almost identical to hierarchical cluster 2. We can also conclude that no more than 123 (39.9%) of the observations in K-Means Cluster 7 fall in any hierarchical cluster.

Problem 2: **MARKET SEGMENTATION FOR AIRLINES**

Market segmentation is a strategy that divides a broad target market of customers into smaller, more similar groups, and then designs a marketing strategy specifically for each group. Clustering is a common technique for market segmentation since it automatically finds similar groups given a data set.

In this problem, we'll see how clustering can be used to find similar groups of customers who belong to an airline's frequent flyer program. The airline is trying to learn more about its customers so that it can target different customer segments with different types of mileage offers.

The file AirlinesCluster.csv contains information on 3,999 members of the frequent flyer program. This data comes from the textbook "Data Mining for Business Intelligence," by Galit Shmueli, Nitin R. Patel, and Peter C. Bruce. For more information, see the website for the book http://www.dataminingbook.com/.

There are seven different variables in the dataset, described below:

- **Balance** = number of miles eligible for award travel
- **QualMiles** = number of miles qualifying for TopFlight status
- **BonusMiles** = number of miles earned from non-flight bonus transactions in the past 12 months
- **BonusTrans** = number of non-flight bonus transactions in the past 12 months
- **FlightMiles** = number of flight miles in the past 12 months
- **FlightTrans** = number of flight transactions in the past 12 months
- **DaysSinceEnroll** = number of days since enrolled in the frequent flyer program

First,let's load the dataset and look at statistical summary:

```
airlines = read.csv("AirlinesCluster.csv")
str(airlines)
```

```
## 'data.frame':    3999 obs. of  7 variables:
##  $ Balance        : int  28143 19244 41354 14776 97752 16420 84914 20856 443003 104860 ...
##  $ QualMiles      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BonusMiles     : int  174 215 4123 500 43300 0 27482 5250 1753 28426 ...
##  $ BonusTrans     : int  1 2 4 1 26 0 25 4 43 28 ...
##  $ FlightMiles    : int  0 0 0 0 2077 0 0 250 3850 1150 ...
##  $ FlightTrans    : int  0 0 0 0 4 0 0 1 12 3 ...
##  $ DaysSinceEnroll: int  7000 6968 7034 6952 6935 6942 6994 6938 6948 6931 ...
```

```
summary(airlines)
```

```
##     Balance          QualMiles         BonusMiles       BonusTrans
##  Min.   :      0   Min.   :    0.0   Min.   :     0   Min.   : 0.0
##  1st Qu.:  18528   1st Qu.:    0.0   1st Qu.:  1250   1st Qu.: 3.0
##  Median :  43097   Median :    0.0   Median :  7171   Median :12.0
##  Mean   :  73601   Mean   :  144.1   Mean   : 17145   Mean   :11.6
##  3rd Qu.:  92404   3rd Qu.:    0.0   3rd Qu.: 23801   3rd Qu.:17.0
##  Max.   :1704838   Max.   :11148.0   Max.   :263685   Max.   :86.0
##   FlightMiles       FlightTrans     DaysSinceEnroll
##  Min.   :    0.0   Min.   : 0.000   Min.   :   2
##  1st Qu.:    0.0   1st Qu.: 0.000   1st Qu.:2330
##  Median :    0.0   Median : 0.000   Median :4096
##  Mean   :  460.1   Mean   : 1.374   Mean   :4119
##  3rd Qu.:  311.0   3rd Qu.: 1.000   3rd Qu.:5790
##  Max.   :30817.0   Max.   :53.000   Max.   :8296
```

It is obious that firstly we need to normalize the data. If we don't normalize the data, the clustering will be dominated by the variables that are on a larger scale.This is done next in our analysis:

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

```
## Warning: package 'caret' was built under R version 3.2.1
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

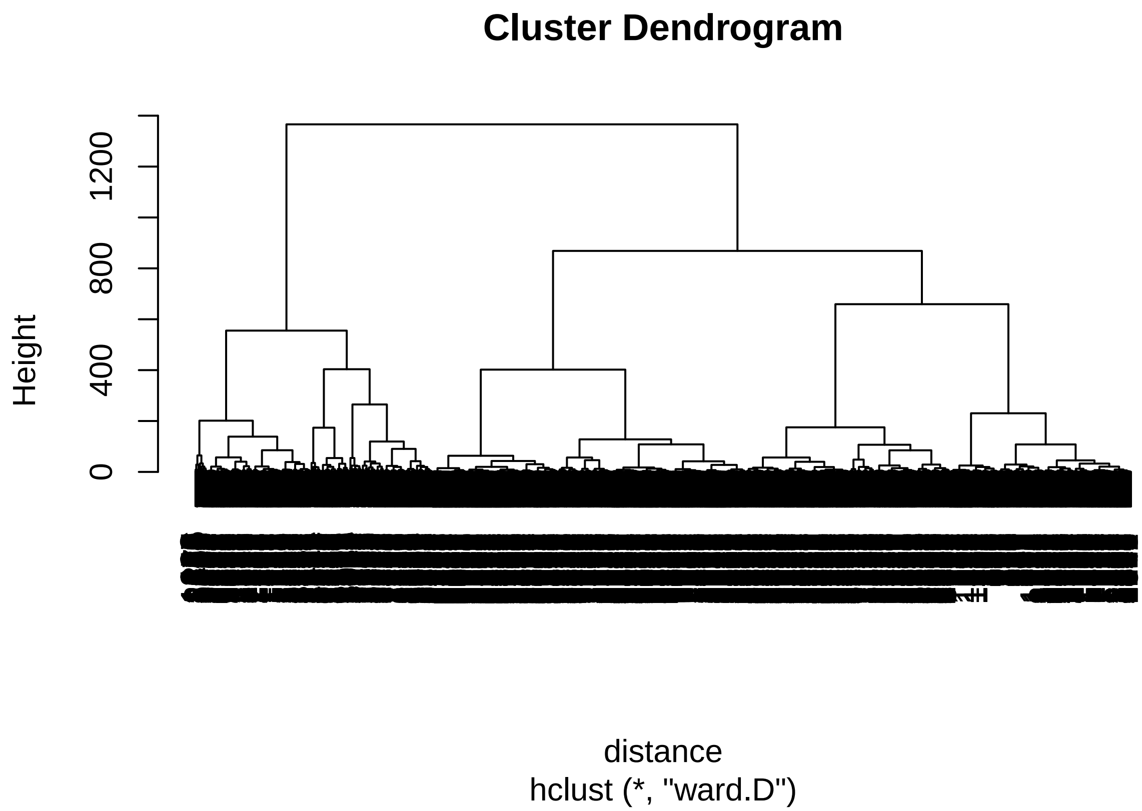
```
## Warning: package 'ggplot2' was built under R version 3.2.1
```

```r
preproc = preProcess(airlines)
airlinesNorm = predict(preproc, airlines)
summary(airlinesNorm)
```

```
##     Balance          QualMiles         BonusMiles        BonusTrans
##  Min.   :-0.7303   Min.   :-0.1863   Min.   :-0.7099   Min.   :-1.20805
##  1st Qu.:-0.5465   1st Qu.:-0.1863   1st Qu.:-0.6581   1st Qu.:-0.89568
##  Median :-0.3027   Median :-0.1863   Median :-0.4130   Median : 0.04145
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.1866   3rd Qu.:-0.1863   3rd Qu.: 0.2756   3rd Qu.: 0.56208
##  Max.   :16.1868   Max.   :14.2231   Max.   :10.2083   Max.   : 7.74673
##   FlightMiles       FlightTrans       DaysSinceEnroll
##  Min.   :-0.3286   Min.   :-0.36212   Min.   :-1.99336
##  1st Qu.:-0.3286   1st Qu.:-0.36212   1st Qu.:-0.86607
##  Median :-0.3286   Median :-0.36212   Median :-0.01092
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.:-0.1065   3rd Qu.:-0.09849   3rd Qu.: 0.80960
##  Max.   :21.6803   Max.   :13.61035   Max.   : 2.02284
```

One can see from the output that FlightMiles now has the largest maximum value, and DaysSinceEnroll now has the smallest minimum value. Note that these were not the variables with the largest and smallest values in the original dataset airlines. Next, we're going to develop hierarchical clustering model:

```r
distance = dist(airlinesNorm, method = "euclidean")
HierCluster = hclust(distance, method = "ward.D")
plot(HierCluster)
```



Looking at the denddogram, we can decide that total number of clusters is from 2 to 7. In the next analysis we'll use k=5 clusters. We can subset the data for each of the cluster as follows:

```
clusterGroups =cutree(HierCluster, k = 5)
HierCluster1 = subset(airlinesNorm, clusterGroups == 1)
HierCluster2 = subset(airlinesNorm, clusterGroups == 2)
HierCluster3 = subset(airlinesNorm, clusterGroups == 3)
HierCluster4 = subset(airlinesNorm, clusterGroups == 4)
HierCluster5 = subset(airlinesNorm, clusterGroups == 5)
```

We can use `lapply` to compare the average values in each of the variables for the 5 clusters (the centroids of the clusters):

```
colMeans(subset(airlines, clusterGroups == 1))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    5.786690e+04    6.443299e-01    1.036012e+04    1.082345e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    8.318428e+01    3.028351e-01    6.235365e+03
```

```
colMeans(subset(airlines, clusterGroups == 2))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    1.106693e+05    1.065983e+03    2.288176e+04    1.822929e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    2.613418e+03    7.402697e+00    4.402414e+03
```

```r
colMeans(subset(airlines, clusterGroups == 3))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    1.981916e+05    3.034615e+01    5.579586e+04    1.966397e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    3.276761e+02    1.068826e+00    5.615709e+03
```

```r
colMeans(subset(airlines, clusterGroups == 4))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    52335.913594        4.847926    20788.766129       17.087558
##     FlightMiles     FlightTrans DaysSinceEnroll
##      111.573733        0.344470     2840.822581
```

```r
colMeans(subset(airlines, clusterGroups == 5))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    3.625591e+04    2.511177e+00    2.264788e+03    2.973174e+00
##     FlightMiles     FlightTrans DaysSinceEnroll
##    1.193219e+02    4.388972e-01    3.060081e+03
```

```r
lapply(split(airlines, clusterGroups), colMeans)
```

```
## $`1`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    5.786690e+04    6.443299e-01    1.036012e+04    1.082345e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    8.318428e+01    3.028351e-01    6.235365e+03
##
## $`2`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    1.106693e+05    1.065983e+03    2.288176e+04    1.822929e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    2.613418e+03    7.402697e+00    4.402414e+03
##
## $`3`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    1.981916e+05    3.034615e+01    5.579586e+04    1.966397e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    3.276761e+02    1.068826e+00    5.615709e+03
##
## $`4`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    52335.913594        4.847926    20788.766129       17.087558
##     FlightMiles     FlightTrans DaysSinceEnroll
##      111.573733        0.344470     2840.822581
##
## $`5`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    3.625591e+04    2.511177e+00    2.264788e+03    2.973174e+00
##     FlightMiles     FlightTrans DaysSinceEnroll
##    1.193219e+02    4.388972e-01    3.060081e+03
```

We also want to analyze the data using K-means algorithm as follows:

```
k = 5 #five clusters
set.seed(88)
KMC = kmeans(airlinesNorm, centers = k, iter.max = 1000)
```

The number of observations in each cluster is easily determined:

```
sum(KMC$cluster==1)
```

```
## [1] 408
```

```
sum(KMC$cluster==2)
```

```
## [1] 141
```

```
sum(KMC$cluster==3)
```

```
## [1] 993
```

```
sum(KMC$cluster==4)
```

```
## [1] 1182
```

```
sum(KMC$cluster==5)
```

```
## [1] 1275
```

```
# or table(KMC$cluster)
```

We can compare the cluster centroids to each other either by dividing the data points into groups and then using tapply, or by looking at the output of kmeansClust$centers, where “kmeansClust” is the name of the output of the kmeans function.

```
colMeans(subset(airlines, KMC$cluster == 1))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    2.191614e+05    5.395784e+02    6.247448e+04    2.152451e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    6.238725e+02    1.921569e+00    5.605051e+03
```

```
colMeans(subset(airlines, KMC$cluster == 2))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    174431.51064       673.16312     31985.08511        28.13475
##     FlightMiles     FlightTrans DaysSinceEnroll
##      5859.23404        17.00000      4684.90071
```

```r
colMeans(subset(airlines, KMC$cluster == 3))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    6.797744e+04    3.499396e+01    2.449002e+04    1.842900e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    2.894713e+02    8.851964e-01    3.416783e+03
```

```r
colMeans(subset(airlines, KMC$cluster == 4))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    6.016618e+04    5.520812e+01    8.709712e+03    8.362098e+00
##     FlightMiles     FlightTrans DaysSinceEnroll
##    2.032589e+02    6.294416e-01    6.109540e+03
```

```r
colMeans(subset(airlines, KMC$cluster == 5))
```

```
##         Balance       QualMiles      BonusMiles      BonusTrans
##    3.270667e+04    1.264667e+02    3.097478e+03    4.284706e+00
##     FlightMiles     FlightTrans DaysSinceEnroll
##    1.814698e+02    5.403922e-01    2.281055e+03
```

```r
lapply(split(airlines, KMC$cluster), colMeans)
```

```
## $`1`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    2.191614e+05    5.395784e+02    6.247448e+04    2.152451e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    6.238725e+02    1.921569e+00    5.605051e+03
##
## $`2`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    174431.51064       673.16312     31985.08511        28.13475
##     FlightMiles     FlightTrans DaysSinceEnroll
##      5859.23404        17.00000      4684.90071
##
## $`3`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    6.797744e+04    3.499396e+01    2.449002e+04    1.842900e+01
##     FlightMiles     FlightTrans DaysSinceEnroll
##    2.894713e+02    8.851964e-01    3.416783e+03
##
## $`4`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    6.016618e+04    5.520812e+01    8.709712e+03    8.362098e+00
##     FlightMiles     FlightTrans DaysSinceEnroll
##    2.032589e+02    6.294416e-01    6.109540e+03
##
## $`5`
##         Balance       QualMiles      BonusMiles      BonusTrans
##    3.270667e+04    1.264667e+02    3.097478e+03    4.284706e+00
##     FlightMiles     FlightTrans DaysSinceEnroll
##    1.814698e+02    5.403922e-01    2.281055e+03
```

The clusters are not displayed in a meaningful order, so while there may be a cluster produced by the k-means algorithm that is similar to Cluster 1 produced by the Hierarchical method, it will not necessarily be shown first.