Mini Project K Means Clustering

Exercise 0

Install these packages if you don't have them already:

install.packages(c("cluster", "rattle", "NbClust"))

load the data and look at the first few rows

```
data(wine, package="rattle")
head(wine)
```

```
##   Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids
## 1    1   14.23  1.71 2.43       15.6       127    2.80       3.06
## 2    1   13.20  1.78 2.14       11.2       100    2.65       2.76
## 3    1   13.16  2.36 2.67       18.6       101    2.80       3.24
## 4    1   14.37  1.95 2.50       16.8       113    3.85       3.49
## 5    1   13.24  2.59 2.87       21.0       118    2.80       2.69
## 6    1   14.20  1.76 2.45       15.2       112    3.27       3.39
##   Nonflavanoids Proanthocyanins Color  Hue Dilution Proline
## 1          0.28            2.29  5.64 1.04     3.92    1065
## 2          0.26            1.28  4.38 1.05     3.40    1050
## 3          0.30            2.81  5.68 1.03     3.17    1185
## 4          0.24            2.18  7.80 0.86     3.45    1480
## 5          0.39            1.82  4.32 1.04     2.93     735
## 6          0.34            1.97  6.75 1.05     2.85    1450
```

Exercise 1:

Remove the first column from the data and scale it using the scale() function

```
df <- scale(wine[-1])
head(df)
```

```
##        Alcohol       Malic        Ash Alcalinity  Magnesium   Phenols
## [1,] 1.5143408 -0.56066822  0.2313998 -1.1663032 1.90852151 0.8067217
## [2,] 0.2455968 -0.49800856 -0.8256672 -2.4838405 0.01809398 0.5670481
## [3,] 0.1963252  0.02117152  1.1062139 -0.2679823 0.08810981 0.8067217
## [4,] 1.6867914 -0.34583508  0.4865539 -0.8069748 0.92829983 2.4844372
## [5,] 0.2948684  0.22705328  1.8352256  0.4506745 1.27837900 0.8067217
## [6,] 1.4773871 -0.51591132  0.3043010 -1.2860793 0.85828399 1.5576991
##      Flavanoids Nonflavanoids Proanthocyanins      Color        Hue
## [1,]  1.0319081    -0.6577078       1.2214385  0.2510088  0.3611585
## [2,]  0.7315653    -0.8184106      -0.5431887 -0.2924962  0.4049085
## [3,]  1.2121137    -0.4970050       2.1299594  0.2682629  0.3174085
## [4,]  1.4623994    -0.9791134       1.0292513  1.1827317 -0.4263410
## [5,]  0.6614853     0.2261576       0.4002753 -0.3183774  0.3611585
## [6,]  1.3622851    -0.1755994       0.6623487  0.7298108  0.4049085
##       Dilution     Proline
## [1,] 1.8427215  1.01015939
```

```
## [2,] 1.1103172  0.96252635
## [3,] 0.7863692  1.39122370
## [4,] 1.1807407  2.32800680
## [5,] 0.4483365 -0.03776747
## [6,] 0.3356589  2.23274072
```

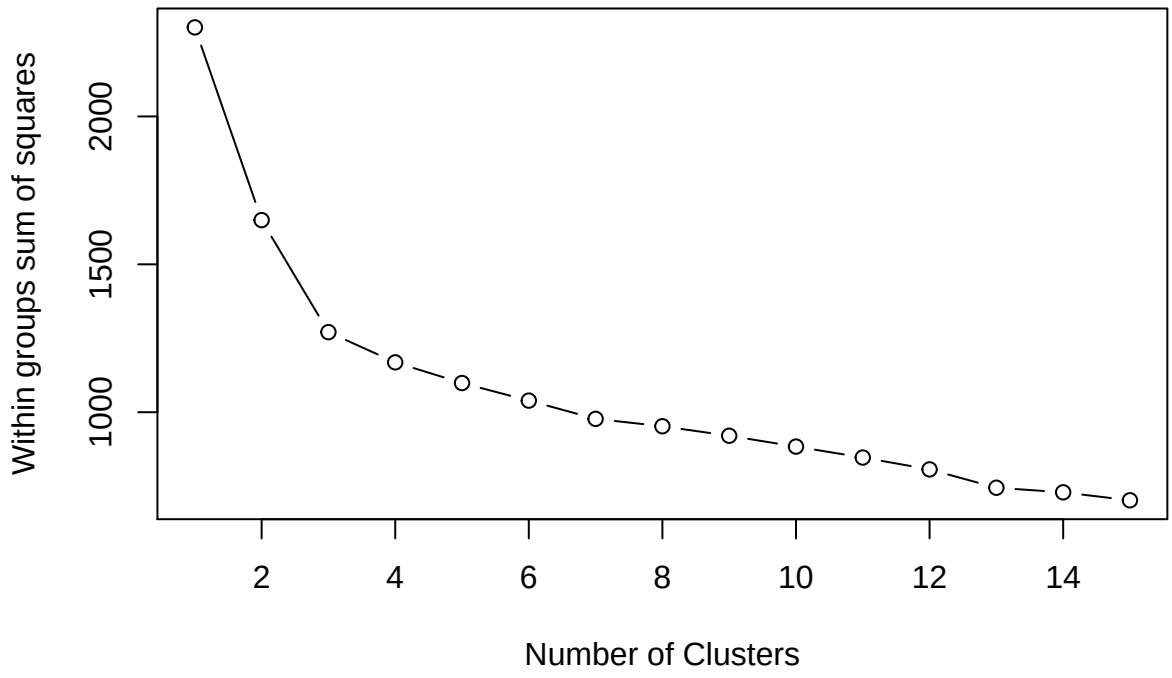
Now we'd like to cluster the data using K-Means. How do we decide how many clusters to use if you don't know that already? We'll try two methods.

Method 1: A plot of the total within-groups sums of squares against the number of clusters in a K-means solution can be helpful. A bend in the graph can suggest the appropriate number of clusters.

```
wssplot <- function(data, nc=15, seed=1234){
                  wss <- (nrow(data)-1)*sum(apply(data,2,var))
                      for (i in 2:nc){
                set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}

              plot(1:nc, wss, type="b", xlab="Number of Clusters",
                           ylab="Within groups sum of squares")
       }

wssplot(df)
```



Exercise 2:

- How many clusters does this method suggest?

Based on where the graph curves, this method suggests 3 clusters

- Why does this method work? What's the intuition behind it?

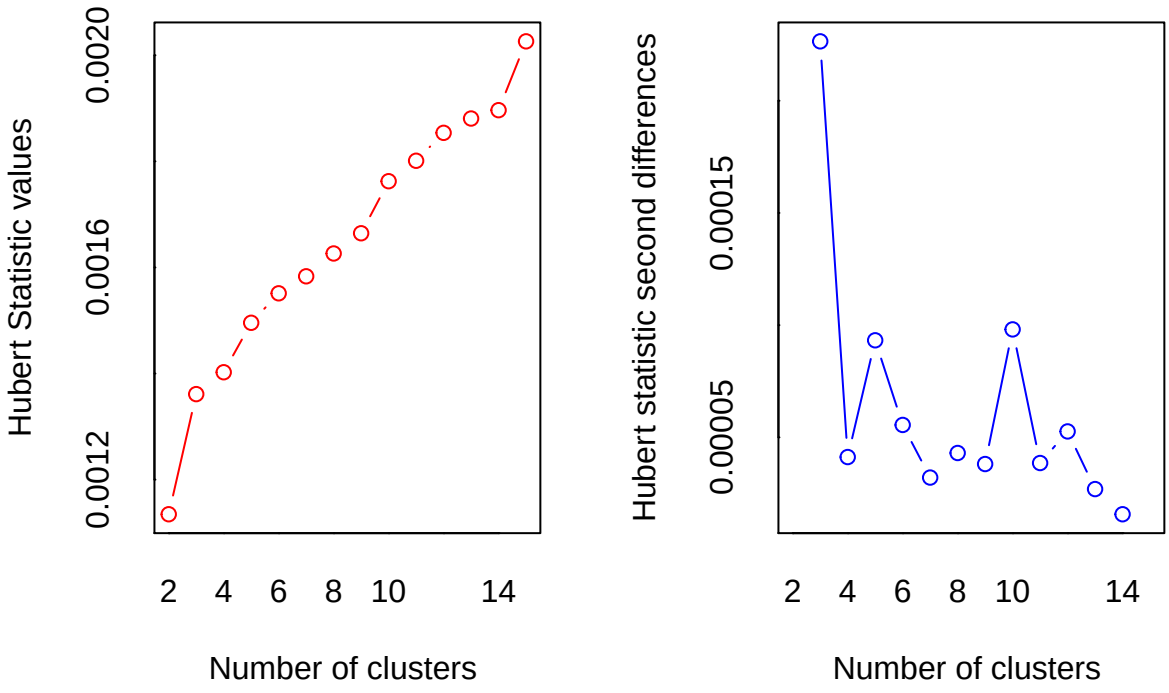
This method explains the degree of variance due to the number of clusters. There is a distinct drop in within groups sum of squares when moving from 1 to 3 clusters. After three clusters this decrease drops off meaning there are decreasing gains by adding further clusters, suggesting that a 3-cluster solution may be a good fit to the data

- Look at the code for wssplot() and figure out how it works

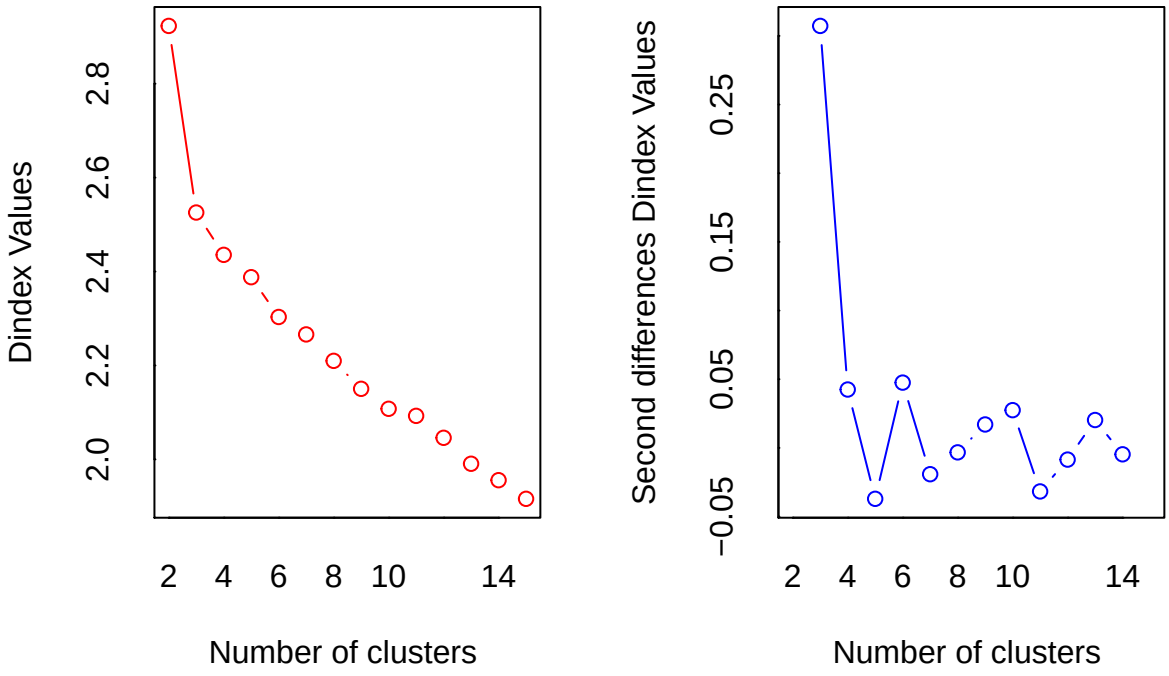
wssplot is a function that loops through fifteen k values. The first value for wss is assigned the sum of squares for k=1 by canceling out the n-1 term in the sum of the variances. Then the function loops from k=2 to k=15, assigning the within sum of squares from the kmeans$withinss component for each alue of k. The function then creates a plot of the within groups sum of squares.

Method 2: Use the NbClust library, which runs many experiments and gives a distribution of potential number of clusters.

```
library(NbClust)
set.seed(1234)
nc <- NbClust(df, min.nc=2, max.nc=15, method="kmeans")
```

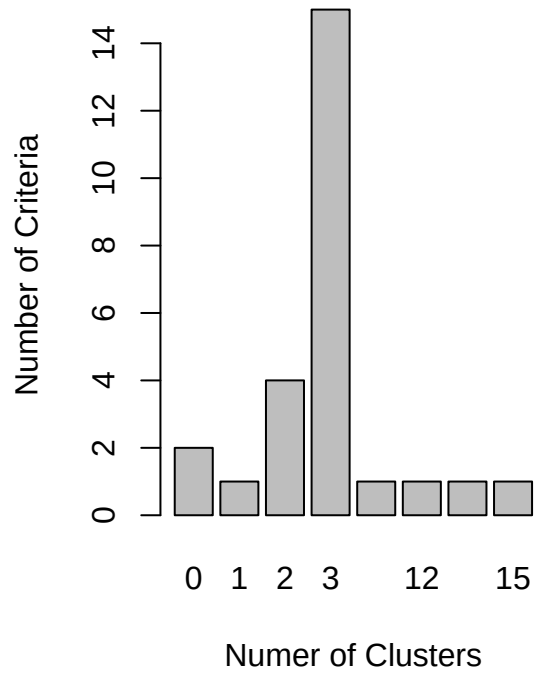


```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *****************************************************************
## * Among all indices:
## * 4 proposed 2 as the best number of clusters
## * 15 proposed 3 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
## * 1 proposed 12 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
## * 1 proposed 15 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *****************************************************************
```

```
barplot(table(nc$Best.n[1,]),
              xlab="Numer of Clusters", ylab="Number of Criteria",
                    main="Number of Clusters Chosen by 26 Criteria")
```



Exercise 3:

How many clusters does this method suggest?

This method also suggests three clusters

Exercise 4:

Once you've picked the number of clusters, run k-means using this number of clusters. Output the result of calling kmeans() into a variable fit.km

```
set.seed(1234)
fit.km <- kmeans(df, centers = 3)
```

Now we want to evaluate how well this clustering does.

Exercise 5:

using the table() function, show how the clusters in fit.km$clusters compares to the actual wine types in wine$Type.

```
table(wine$Type, fit.km$cluster)
```

```
##
##      1  2  3
##   1 59  0  0
##   2  3 65  3
##   3  0  0 48
```

Would you consider this a good clustering?

Yes: this indicates there are only 6 errors in our model

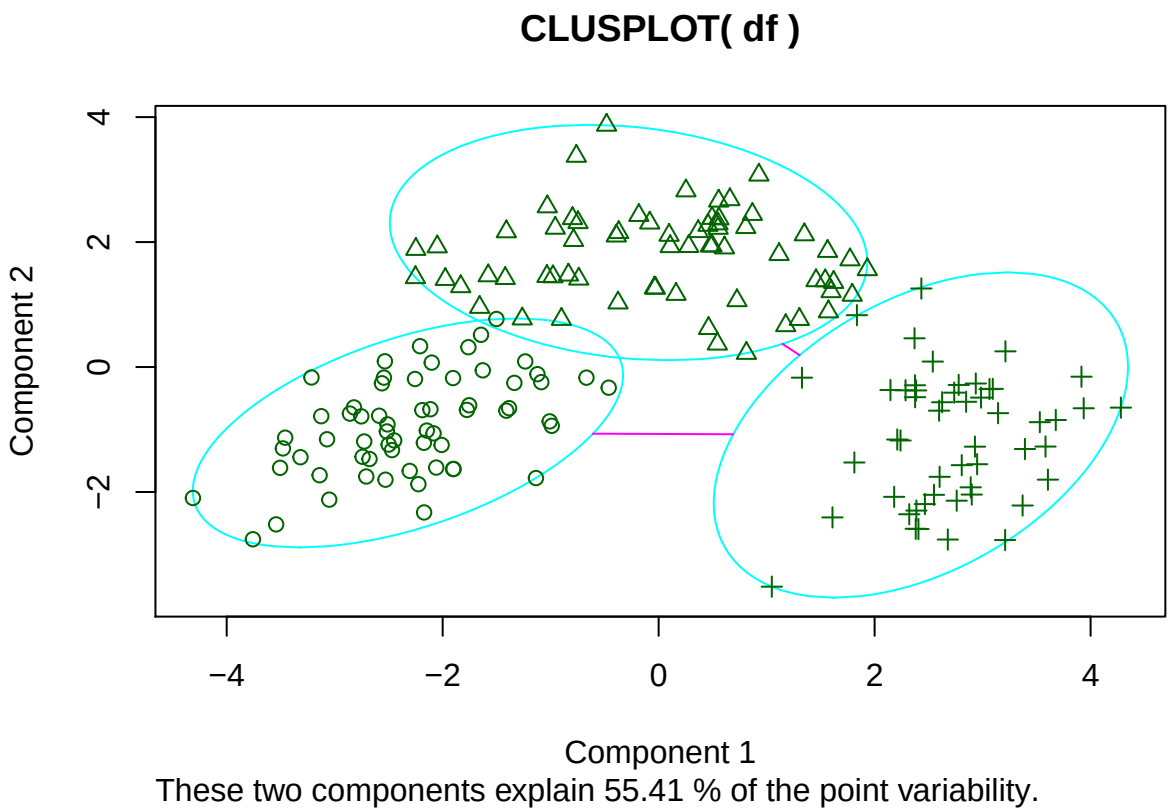
Exercise 6:

- Visualize these clusters using function clusplot() from the cluster library

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 3.2.5
```

```
clusplot(df, fit.km$cluster)
```



- Would you consider this a good clustering?

Yes, however it is important to note that only 55% of the variance is shown in this plot, as many variables have been reduced to two components to create this graph