Clustering project

Chris Leisner

July 29, 2017

This mini-project is based on the K-Means exercise from 'R in Action' Go here (http://www.r-bloggers.com/k-means-clustering-from-r-in-action/) for the original blog post and solutions

Exercise 0: Install these packages if you don't have them already

```
#install.packages(c("cluster", "rattle","NbClust"))
#I had to comment this out because it would not run in r markdown.
```

```
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 the remaining data using the scale() function

```
wineType <- wine$Type
levels(wineType)
```

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

We will change the names of the winetypes from 1, 2, 3 to "type1", "type2", "type3" so that later, we can more easily distinguish the type labels from the cluster labels

```
wineType2 <- ifelse(wineType == 1, "type1", ifelse(wineType == 2, "type2", "type3"))
```

Next we'll make sure that we re-labeled the wine types correctly

```
wineTypes <- as.data.frame(cbind(wineType, wineType2))
table(wineTypes$wineType, wineTypes$wineType2)
```

```
##
##     type1 type2 type3
##   1    59     0     0
##   2     0    71     0
##   3     0     0    48
```

Yes, the re-labeling of wine types was done correctly.

Remove "type" from wine data, then scale the wine data

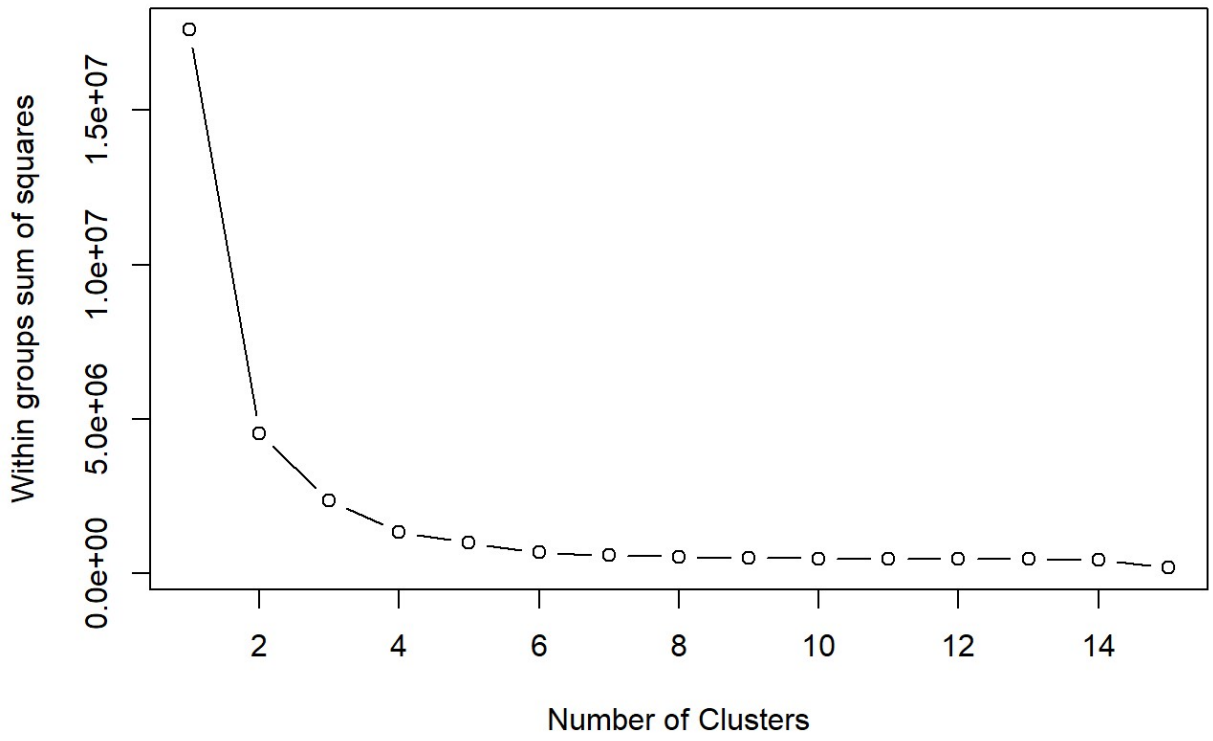
```
wine <- subset(wine, select = -Type )
scale(wine, center = TRUE, scale = TRUE)
```

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")
       }
plot.new()
wssplot(wine)
```



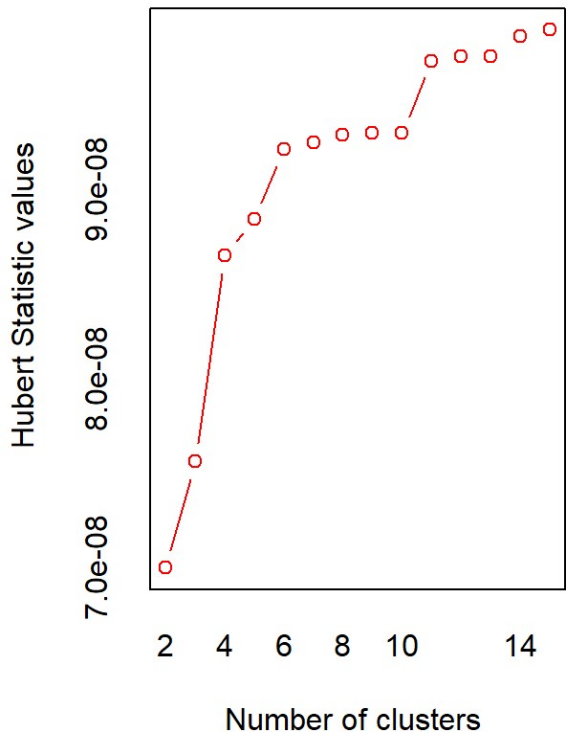
Exercise 2:

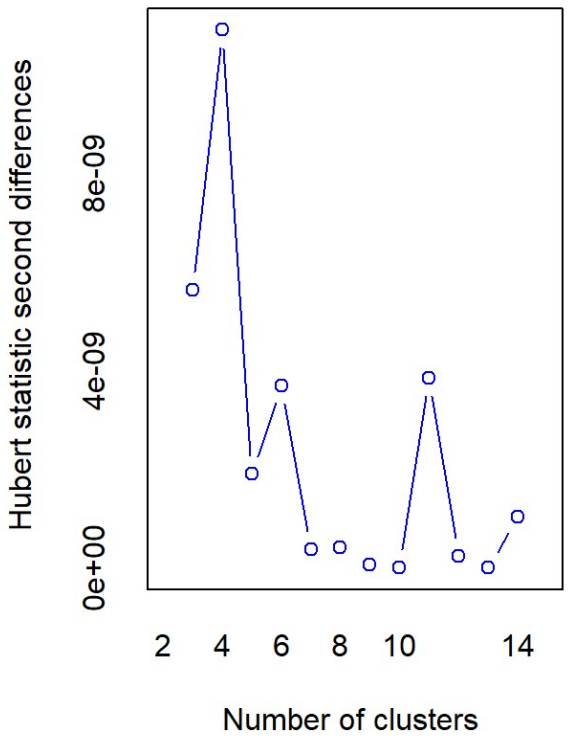
- How many clusters does this method suggest?
- Why does this method work? What's the intuition behind it?
- Look at the code for wssplot() and figure out how it works

This method suggests about four clusters, because according to the graph, there is a leveling off of the total within-groups sum of squares at four clusters. Basically, the total inter-cluster distances from the cluster points to the cluster centers is almost minimized when we use four clusters. We could minimize the total inter-cluster distance even more by making more clusters, but the resulting increased "tightness" of the clusters would be tiny and not worth the extra cluster creation.

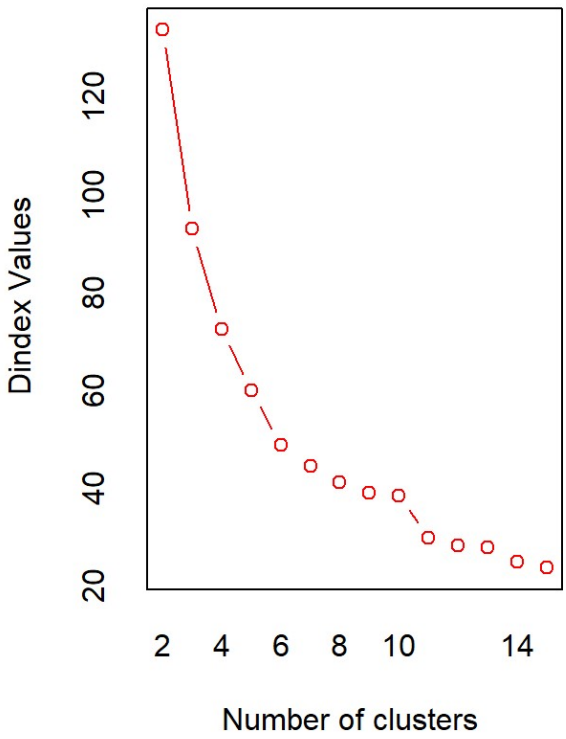
Method 2: Use the NbClust library, which runs many experiments and gives a distribution of potential number of clusters.

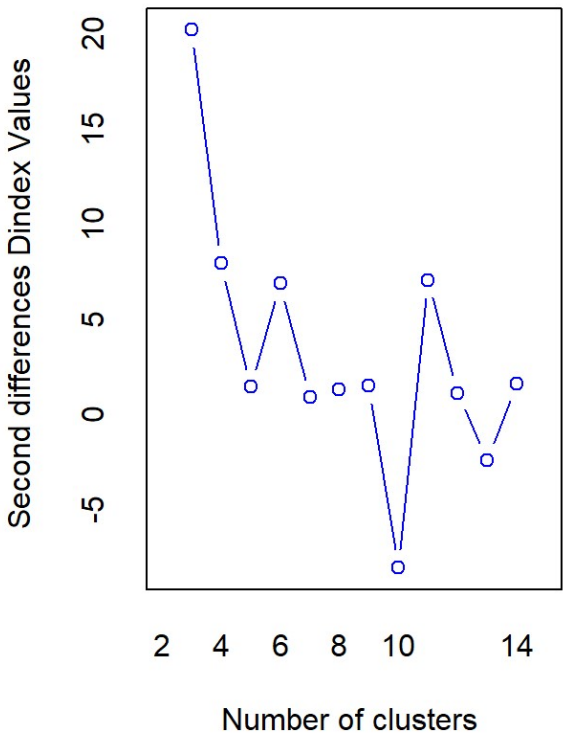
```
library(NbClust)
set.seed(1234)
nc <- NbClust(wine, min.nc=2, max.nc=15, method="kmeans")
```





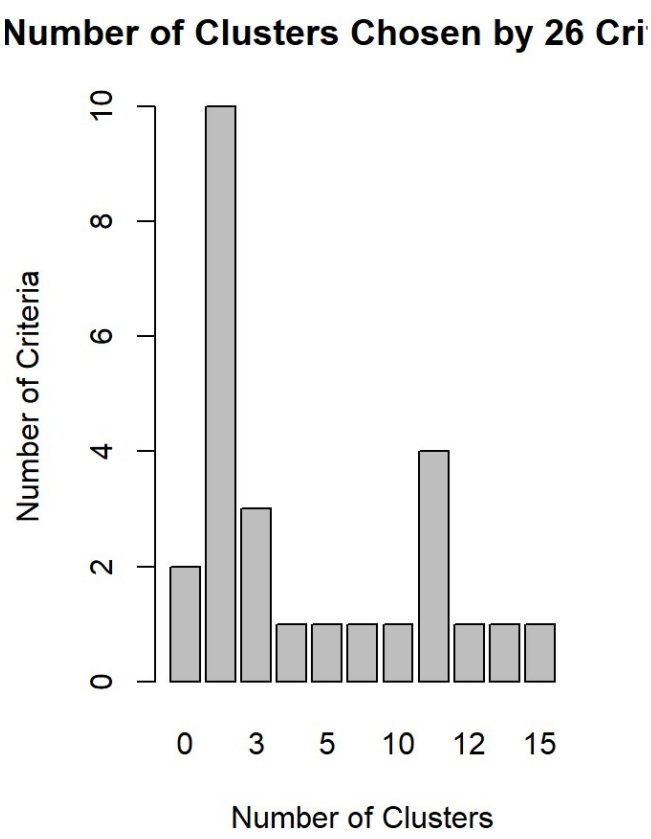

```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corres
ponds to a
##                 significant increase of the value of the measure i.e the significan
t peak in Hubert
##     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


Dindex Values
20
40
60
80
100
120
2
4
6
8
10
14
Number of clusters


Second differences Dindex Values
-5
0
5
10
15
20
2
4
6
8
10
14
Number of clusters

```
## *** : 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 significan
t peak in Dindex
##                 second differences plot) that corresponds to a significant increas
e of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 10 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 9 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
## * 4 proposed 11 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  2
##
##
## *******************************************************************
```

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



Exercise 3: How many clusters does this method suggest?

"According to the majority rule, the best number of clusters is 2"

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

Since the first method suggested four clusters and the second method suggested two clusters, I will choose the average of these two numbers as my number of clusters. An additional justification for choosing three clusters is that this is the number of wine types.

```
fit.km <- kmeans(wine, centers = 3, iter.max = 1000 )
wineClusters <- fit.km$cluster
```

Here we will rename clusters 1, 2 and 3 "cluster1", "cluster2", and "cluster3" to distinguish the clusters from the wine types more easily:

```
wineClusters2 <- ifelse(wineClusters == 1, "cluster1", ifelse(wineClusters == 2, "clus
ter2", "cluster3"))
wineClusters3 <- as.data.frame(cbind(wineClusters, wineClusters2))
```

Now we make sure that the renaming of the clusters was done correctly:

```
table(wineClusters3$wineClusters, wineClusters3$wineClusters2)
```

```
##
##     cluster1 cluster2 cluster3
##   1       62        0        0
##   2        0       69        0
##   3        0        0       47
```

Yes, the renaming was done correctly

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. Would you consider this a good clustering?

```
wine <- cbind(wine, wineClusters2)
wine <- cbind(wine, wineType2)
table(wine$wineClusters2, wine$wineType2)
```

```
##
##            type1 type2 type3
##   cluster1    13    20    29
##   cluster2     0    50    19
##   cluster3    46     1     0
```

This table is rather confusing, because the cluster numbers don't equal the numbers of their corresponding wine types. The comparison of types and clusters will be more clear if we rename the clusters to match the numbers of the corresponding wine types. That is, relabel cluster 1 as cluster 3 and cluster 3 as cluster 1. We do this below:

```
wine$wineClusters3 <- ifelse(wine$wineClusters2 == "cluster1", "cluster3", ifelse(wine
$wineClusters2 == "cluster3", "cluster1", "cluster2"))
```

Now we make sure that the renaming of the the clusters was done correctly:

```
table(wine$wineClusters2, wine$wineClusters3)
```

```
##
##            cluster1 cluster2 cluster3
##   cluster1        0        0       62
##   cluster2        0       69        0
##   cluster3       47        0        0
```

Yes, the renaming of the clusters was done correctly.

Now let's do the table again:

```
table(wine$wineClusters3, wine$wineType2)
```

```
##
##            type1 type2 type3
##   cluster1    46     1     0
##   cluster2     0    50    19
##   cluster3    13    20    29
```

I would not consider this a good clustering, because there are too many misclassified wines. For example, 13 of the 59 wines of type 1, or 22 percent of the wines of type 1, were placed into the wrong cluster. In addition, 29.6 percent of the wines of type 2 were placed into the wrong cluster, and 0.4 percent of the wines of type 3 were placed into the wrong cluster.

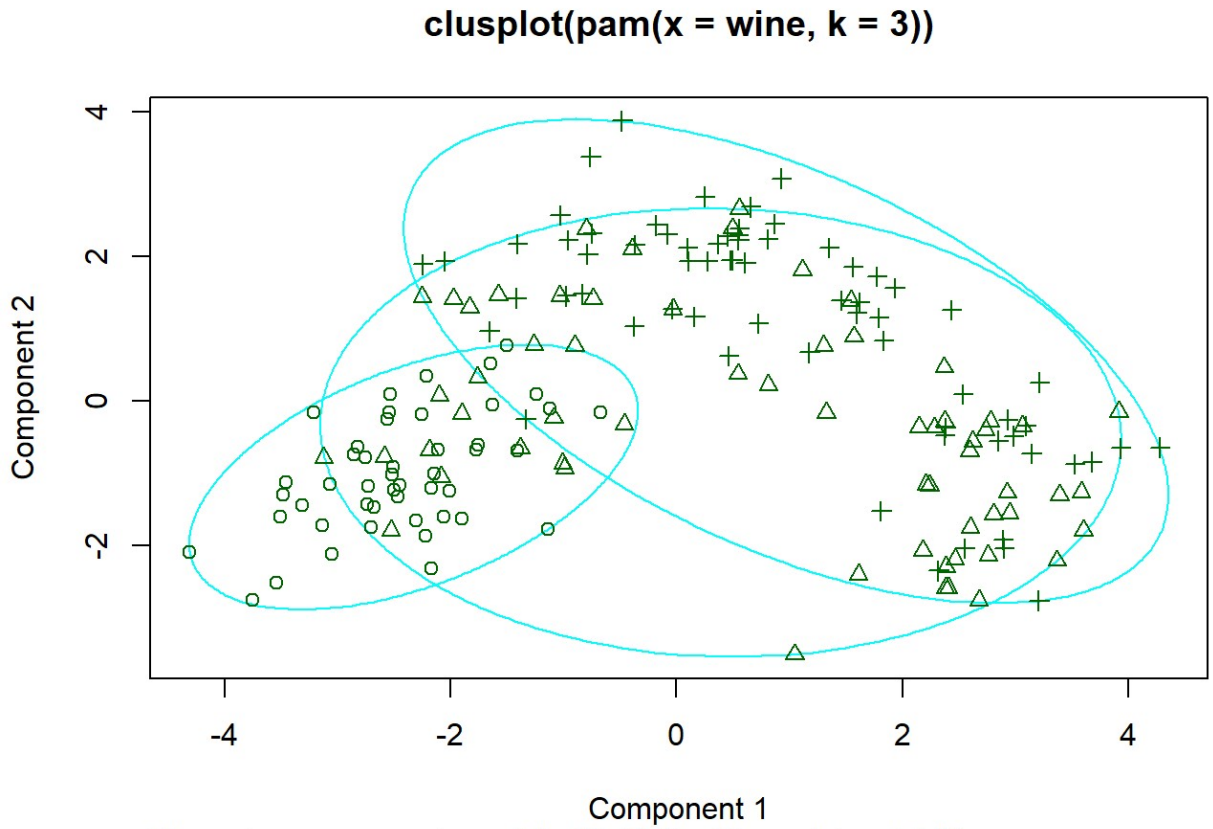
Exercise 6:

- Visualize these clusters using function clusplot() from the cluster library
- Would you consider this a good clustering?

```
wine <- subset(wine, select = -c(wineClusters2, wineClusters3, wineType2))
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 3.3.3
```

```
clusplot(pam(wine,3))
```



These two components explain 55.41 % of the point variability.

I would not consider this a good clustering. There are a great number of observations in the intersections of the three clusters. Moreover, “These two components explain [only] 55.41% of the point variability”. In a good clustering the components would explain close to 100% of the point variability.