Solution Lesson02: KMeans

Machine Learning using R

Exercise 1: Wine quality data

The wine quality data set summarizes various properties of different wines. The variables include amongst others the amount of alcohol, the ph-value, the density, the residual sugar and a quality-score (ranging from 0-10) .There are two data files in the folder of this exercise, one for red wines and one for white wines. Load the data sets "winequality-red.csv" and "winequality-white.csv" using the `read.csv('winequality-red.csv', stringsAsFactors=TRUE)` command (same for `winequality-white.csv`).

a) Since the red and white wine data have the same variables, we wish to combine them into one big data frame. First, add a new column (`colour`) to both data sets taking the value `"red"` for red wines and `"white"` for white wines. Afterwards, bind the two sets together into one big data set. (**Hint**: `rbind()`)

```r
dat.white$colour <- 'white'
dat.red$colour <- 'red'
dat.wine <- rbind(dat.white, dat.red)
```

b) Check the structure of the newly created data set. Turn the `colour` variable into a factor.

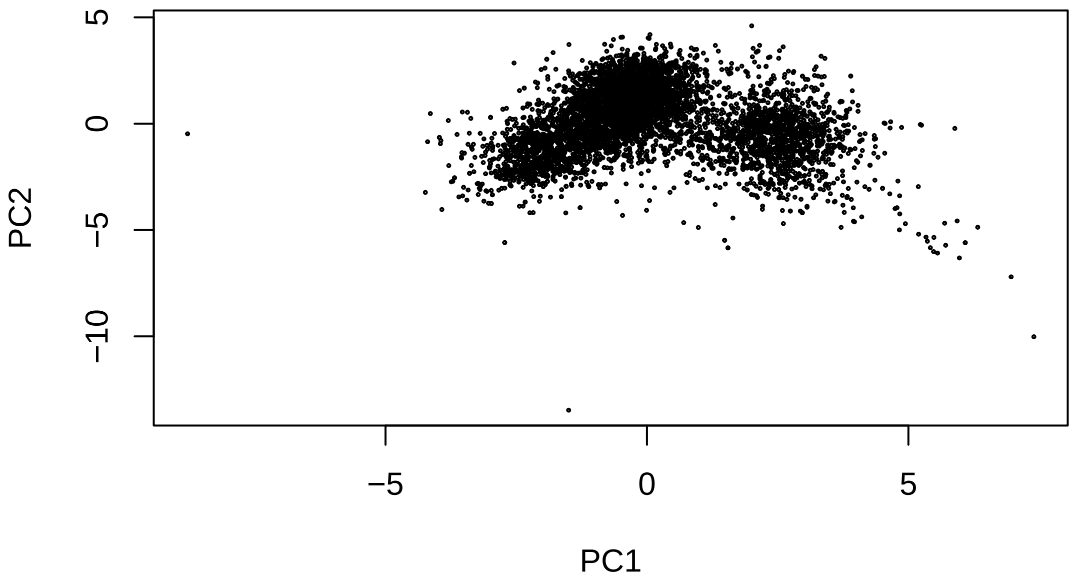
```r
str(dat.wine)
```

```
## 'data.frame':    6497 obs. of  13 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ colour              : chr  "white" "white" "white" "white" ...
```

```r
dat.wine$colour <- as.factor(dat.wine$colour)
```

c) We wish to represent the wine data in two dimensions. Run a PCA and plot the two first PCs. Exclude the colour factor from the PCA (only numerical variables allowed). Also exclude the `quality` variable from the PCA. (**Hint**: `prcomp(..., scale.=T)`)

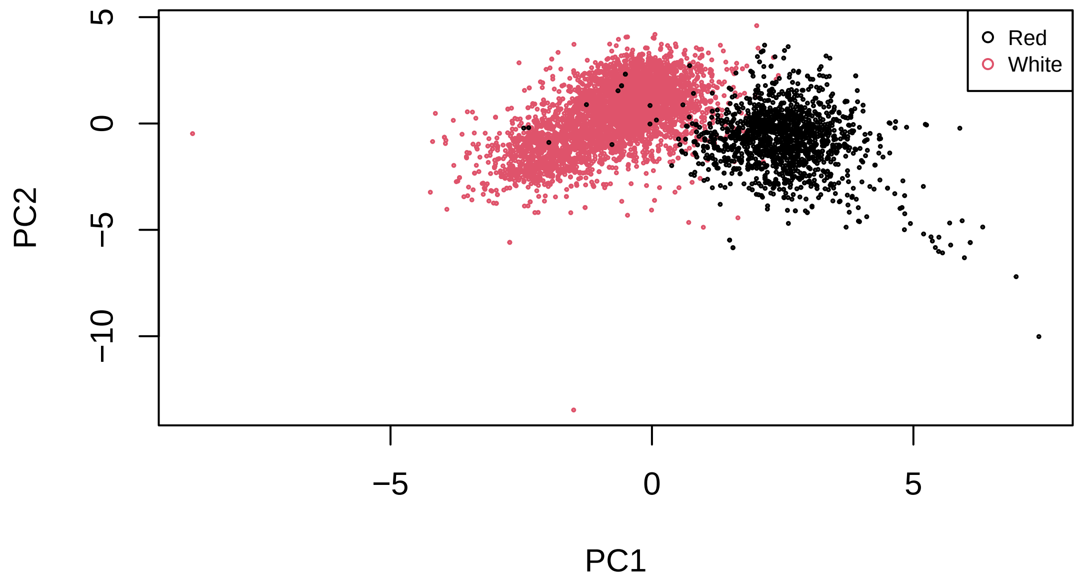
```
dat.pca <- dat.wine
dat.pca$colour <- NULL # remove column
dat.pca$quality <- NULL # remove column
pc.wine <- prcomp(dat.pca, scale. = TRUE)
plot(PC2~PC1, data=pc.wine$x, cex=0.2)
```



It seems there are two main clusters in the PCA plot.

d) Plot the PCs again with the colour of the points representing white and red wines. What can you observe?

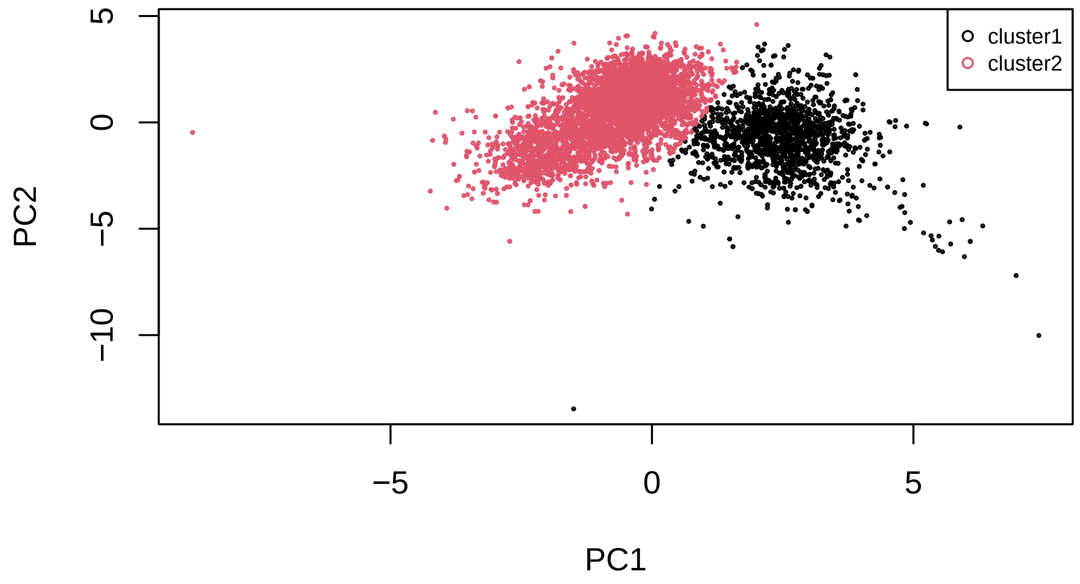
```
plot(PC2~PC1, data=pc.wine$x, col=dat.wine$colour, cex=0.2)
legend('topright', c('Red', 'White'), pch=c(1,1),col=c(1,2), cex=0.7)
```



The two clusters observed above seem to be the red and white wines. Thus, red and white wines are relatively well separated by the first two PCs.

e) We now want to see how a k-means algorithm would cluster the data represented in the 2D PCA-coordinates. Apply k-means clustering to the coordinates of the two first PCs with k=2 clusters. Set the random seed to 111 (`set.seed(111)`). Plot the created clusters. (**Hint**: `kmeans()`, `pca_obj$x`)

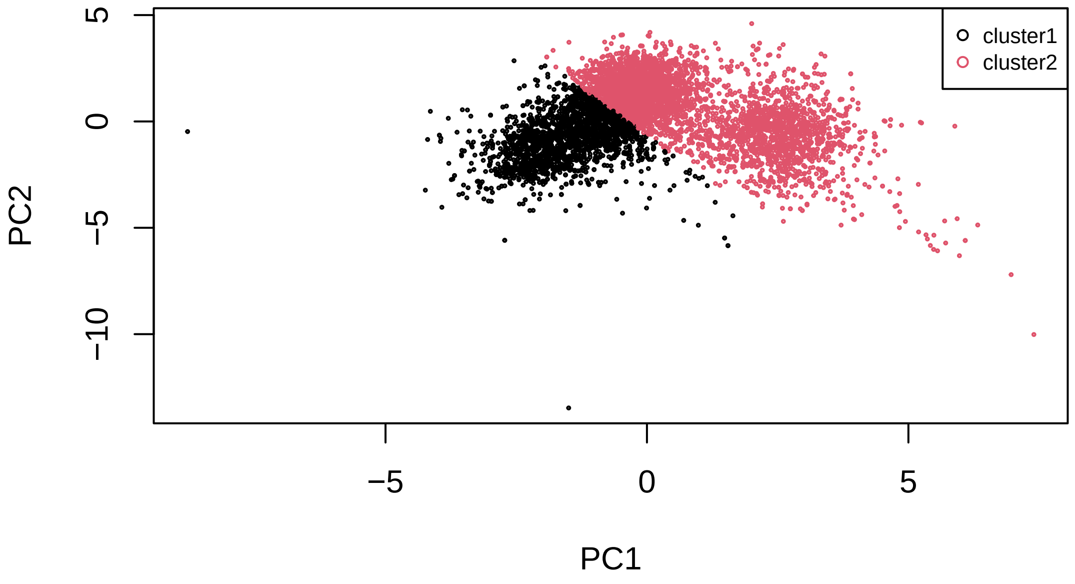
```
set.seed(111)
km.dat <- pc.wine$x[,1:2]
km.1 <- kmeans(km.dat, centers = 2)
plot(PC2~PC1, data=pc.wine$x, col=km.1$cluster, cex=0.2)
legend('topright', c('cluster1', 'cluster2'), pch=c(1,1),col=c(1,2), cex=0.7)
```



The k-means algorithm creates two clusters which mostly agree with the visual impression of the data.

f) Apply again k-means clustering with k=2 clusters to the data, but this time set the random seed to 12 (`set.seed(12)`). How do you explain the result?

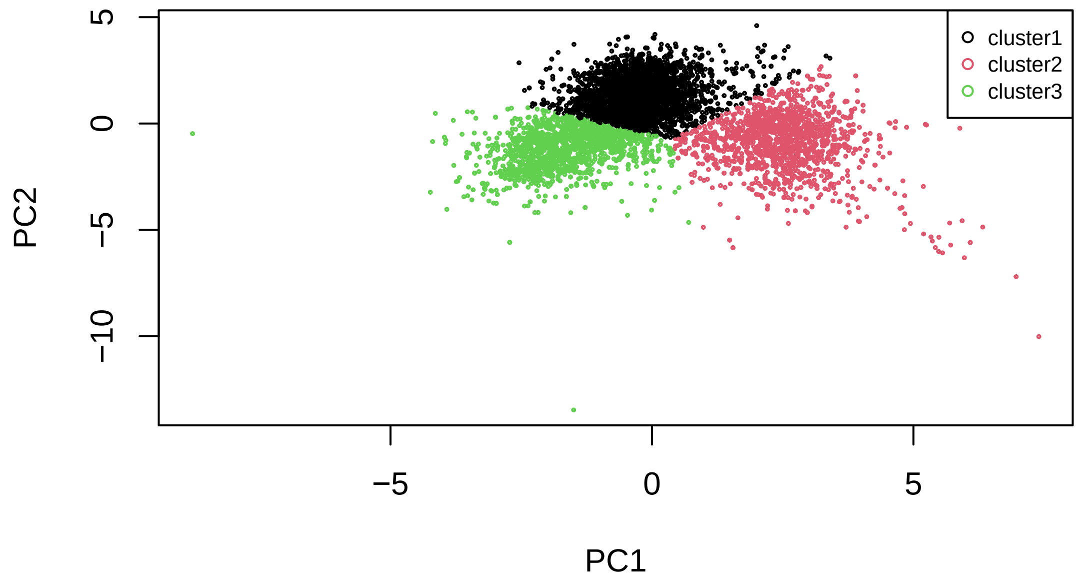
```
set.seed(12)
km.2 <- kmeans(km.dat, centers = 2)
plot(PC2~PC1, data=pc.wine$x, col=km.2$cluster, cex=0.2)
legend('topright', c('cluster1', 'cluster2'), pch=c(1,1),col=c(1,2), cex=0.7)
```



This time the clusters are chosen differently. This is due to k-mean's sensitivity to the starting configuration. Using a different random.seed has caused the algorithm to converge at a different final result. The created two clusters agree less with the clusters one would expect based on the visual inspection of the data.

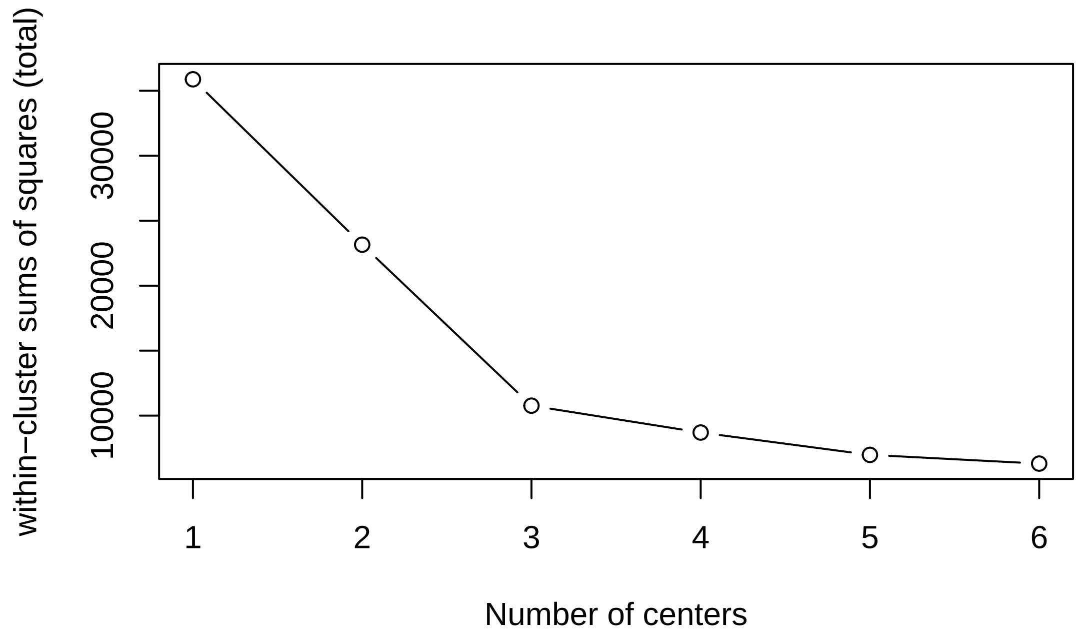
g) Use k=3 clusters and plot the results.

```
km.3 <- kmeans(km.dat, centers = 3)
plot(PC2~PC1, data=pc.wine$x, col=km.3$cluster, cex=0.2)
legend('topright', c('cluster1', 'cluster2', 'cluster3'), pch=c(1,1,1),col=c(1,2,3), cex=0.7)
```



h) What would you say is the best number of clusters for this data? Answer based on your visual impression of the results and also by looking at the within-cluster sums of squares.

```r
wss <- rep(NA, 6)  # initialize
set.seed(234)
for(i in 1:6){  # Check for up to 6 clusters
  wss[i] <- kmeans(km.dat, centers = i)$tot.withinss  # Sum of within-clust.SumSquares
}
plot(1:6, wss, type = 'b', xlab = 'Number of centers',
     ylab = 'within-cluster sums of squares (total)')
```

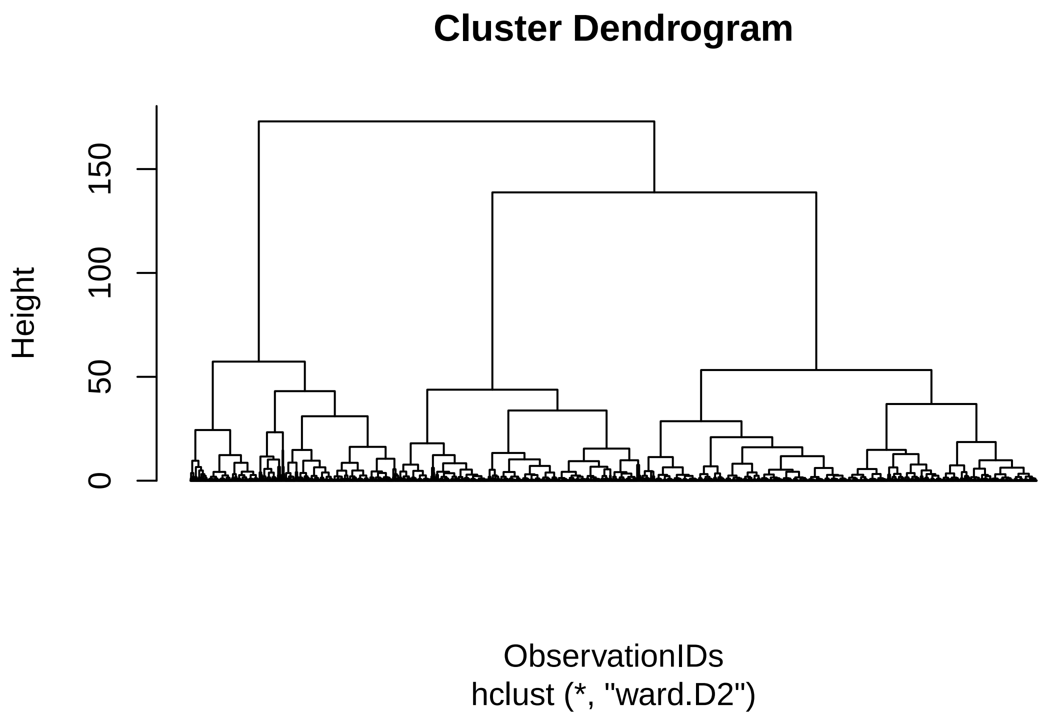


Based on the visual inspection of the data it seems that two clusters are sufficient to represent the data.

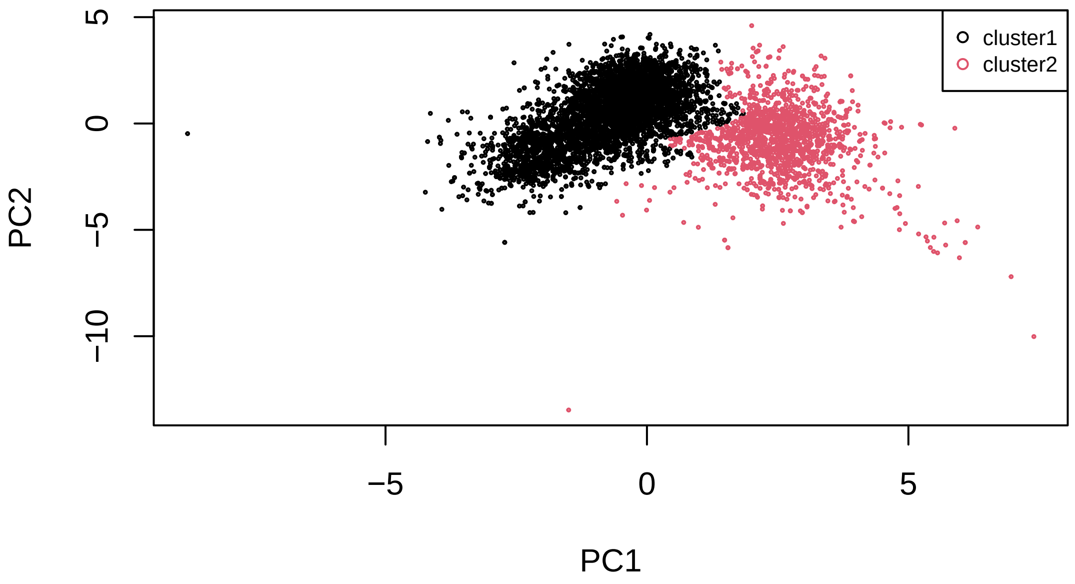
Looking at the plotted within-cluster sums of squares, however, there is a large drop in variance when going from two to three clusters. This would suggest that three clusters best represent the data. Which of the two options is finally chosen is up to the respective researcher and his/her assessment of the situation.

i) Apply hierarchical clustering to the coordinates of the first two PCs (choose the `ward.D2` method). Plot the created dendrogram. Plot the PCs again and indicate by colour the two biggest clusters of the dendrogram. (**Hint**: `hclust()`, `cutree()`)

```
dist.points <- dist(km.dat)
hc <- hclust(dist.points, method = 'ward.D2')
plot(hc, xlab = 'ObservationIDs', labels = FALSE, hang=-1)
```



```
plot(PC2~PC1, data=pc.wine$x, col=cutree(hc, k = 2), cex=0.2)
legend('topright', c('cluster1', 'cluster2'), pch=c(1,1),col=c(1,2), cex=0.7)
```



As we can see, the clusters are not exactly the same as the two clusters produced with the k-means algorithm.