Exercise:
K-Means Clustering

Introduction to Machine Learning with R

SOLUTION

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 sets, one for red wines and one for white wines.

a) Load the two data sets contained in the `wine_data.rda` file using the `load` command. Inspect the two data sets to get a first impression of the data. You can also draw pairs plots for the two data sets to get a visual impression.

```
load('wine_data.rda')
head(wine_red, 3)   # Only showing 3 rows to save space
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
##   quality
## 1       5
## 2       5
## 3       5
```

```
head(wine_white, 3)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
##   quality
## 1       6
## 2       6
## 3       6
```

```
str(wine_red)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

```r
str(wine_white)
```

```
## 'data.frame':    4898 obs. of  12 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 ...
```

The two data sets for white and red wines have the same variables. All variables are numeric variables. There are fewer observations for red (1599) than white wines (4898). The pairs plots of the two data sets (not shown) reveals that there are some extreme points in some of the variables. For further analyses one could think about removing these potential outliers, but here we will leave them as they are.

b) We want to add a `colour` column to both data sets. For the white wine data, this column should only have the value `"white"` and for the red wine data `"red"`. (**Hint**: `wine_red$colour <- ...`)

```r
wine_red$colour <- 'red'
wine_white$colour <- 'white'
head(wine_white, 3)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
##   quality colour
## 1       6  white
## 2       6  white
## 3       6  white
```

c) Combine the two data sets into one big data set called `wines` using the `rbind` command. The `rbind` command can take multiple data frames as arguments and combines them rowwise. Check what the dimensions of the new data set are to make sure the combination worked.

```
wines <- rbind(wine_red, wine_white)
dim(wines)
```

```
## [1] 6497   13
```

We now have all the data combined in one data frame, which now has 1599 + 4898 = 6497 rows.

d) Turn the `colour` variable in the `wines` data frame into a factor.

```
wines$colour <- factor(wines$colour)
```

e) Although the `colour` variable will be of use later, for k-means clustering we can only use numerical variables. Create a copy of the data frame with the name `wines_nocol` which does not include the `colour` column. (**Hint** `wines_nocol$... <- NULL`)

```
wines_nocol <- wines
wines_nocol$colour <- NULL
```

f) The `wines_nocol` data frame only includes numeric variables. We want to scale this data before we apply any cluster algorithm to it. The reason is that the variables are on rather different scales, and we don't want variables with larger variance to dominate any measures of distance between points. To scale all columns of a data frame, we can use the `scale` function, which takes a data frame as an argument. Because the `scale` function automatically changes the data into a matrix, we will apply the `data.frame` function to the result again, so that we retain the data as a data frame. (**Hint**: `data.frame( scale(...) )`)

```
wines_nocol <- data.frame(scale(wines_nocol))
```

Now all variables have a variance of 1 and a mean of 0.

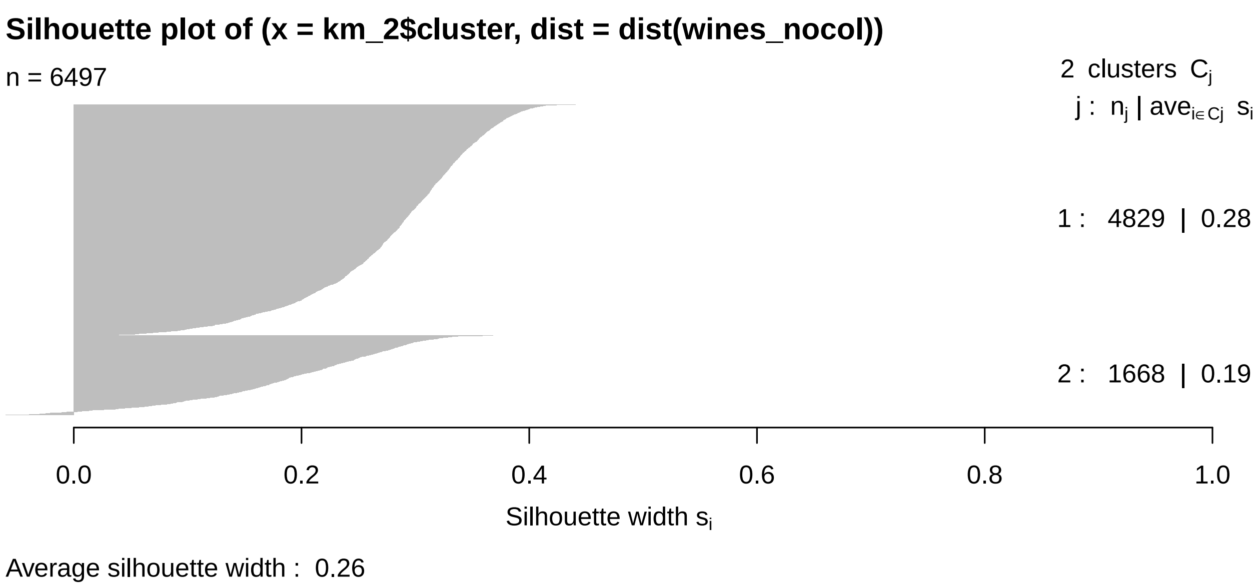
g) Apply k-means clustering to the scaled data looking for two clusters. Do not set the random seed before applying the `kmeans` function. To get an idea of how good the found cluster pattern is, create a silhouette plot of the found solution. Interpret the result. Now run the used code a couple of times again. Do the results always turn out the same way? Why not?

```
km_2 <- kmeans(wines_nocol, centers = 2)
library(cluster)
plot(silhouette(x = km_2$cluster, dist = dist(wines_nocol)), border=NA)
```

We get different solutions because k-means clustering can depend on the randomly selected starting values (only one solution shown here). The solutions we get do not achieve very good silhouette scores.

h) We can force the `kmeans` function to try out multiple random starting positions and choose among the achieved solutions the best one. This can be done with the `nstart` option. Apply again k-means clustering to the scaled data looking for two clusters, but trying out 10 random starting positions (**Hint**: `kmeans(..., nstart=10)`). Draw again a silhouette plot to inspect the solution.

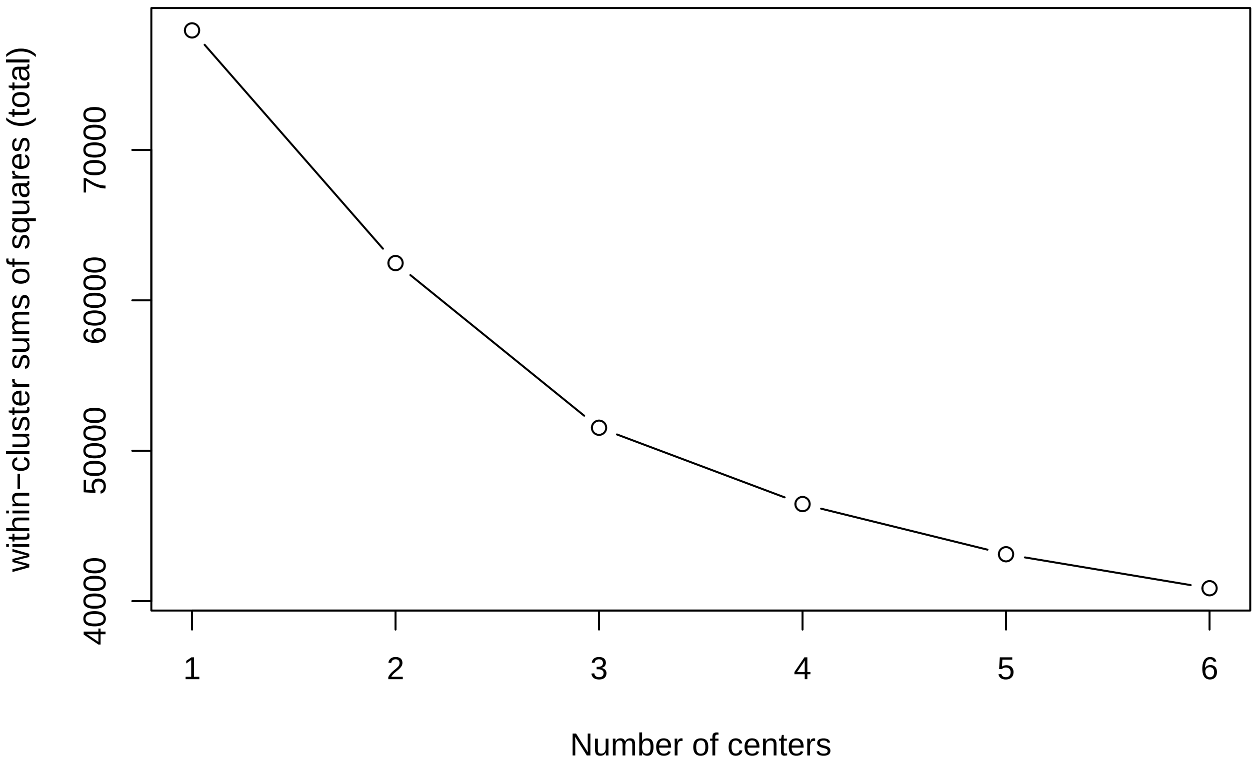
```
set.seed(4890)
km_2 <- kmeans(wines_nocol, centers = 2, nstart = 10)
plot(silhouette(x = km_2$cluster, dist = dist(wines_nocol)), border=NA)
```



Although the solution is better than the previously shown one, according to the usual rules of thumb the silhouette plot still shows rather bad average cluster scores. However, as we will see later, for our data this is actually not such a bad solution as one might think based on the silhouette plot alone.

i) To investigate whether a different number of clusters fits our data better, we want to calculate the within-cluster SSQ for different values of k. Visualize the resulting SSQ when using the values 1 to 6 for k. Use again multiple random starting positions when applying `kmeans`. Interpret the resulting figure.

```
wss <- rep(NA, 6)    # Initialize
set.seed(2145)
for(i in 1:6){ # Check for up to 6 clusters
  wss[i] <- kmeans(wines_nocol, centers = i, nstart = 10)$tot.withinss
}
plot(1:6, wss, type = 'b', xlab = 'Number of centers',
ylab = 'within-cluster sums of squares (total)')
```



The figure doesn't show a very clear bend in the line. There might to be a bit of an 'elbow' at the value k=3.

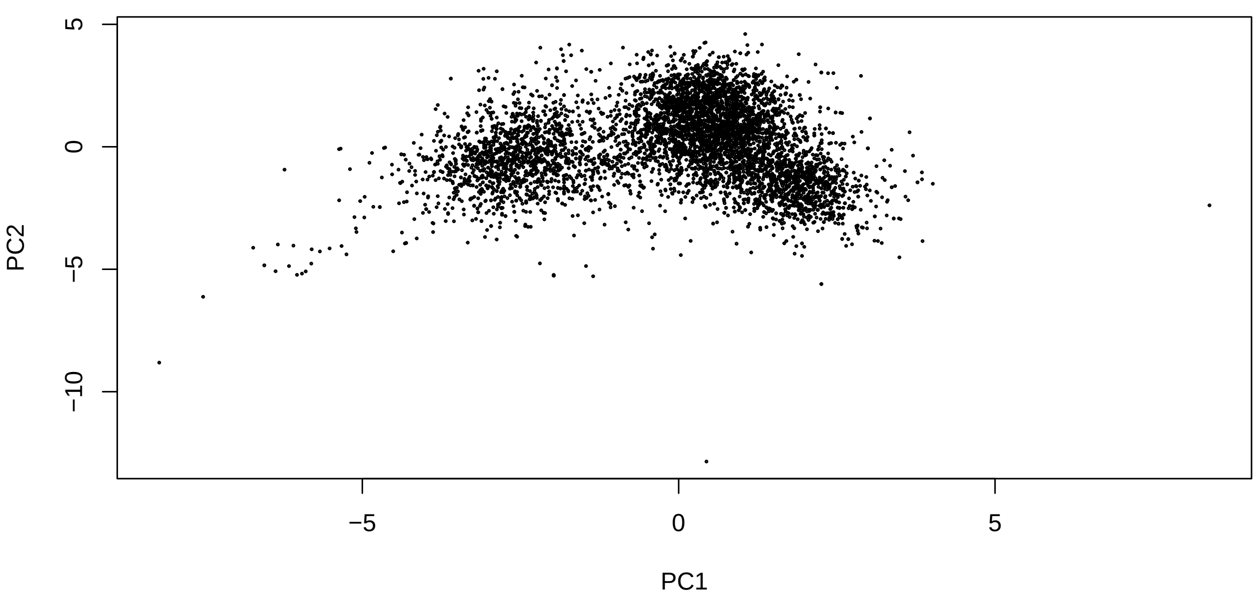
j) Based on the previous exercise, apply k-means clustering to the data looking for three clusters and create the associated silhouette plot.

```
set.seed(8893)
km_3 <- kmeans(wines_nocol, centers = 3, nstart = 10)
plot(silhouette(x = km_3$cluster, dist = dist(wines_nocol)), border=NA)
```

Again, the silhouette scores are not that good.

k) We would like to get a visual impression of the data to see what is going on. Perform a dimensionality reduction by applying a PCA to the data and plot the first two PCs (since we have scaled the data before, it doesn't make a difference if we apply the PCA with or without scaling).

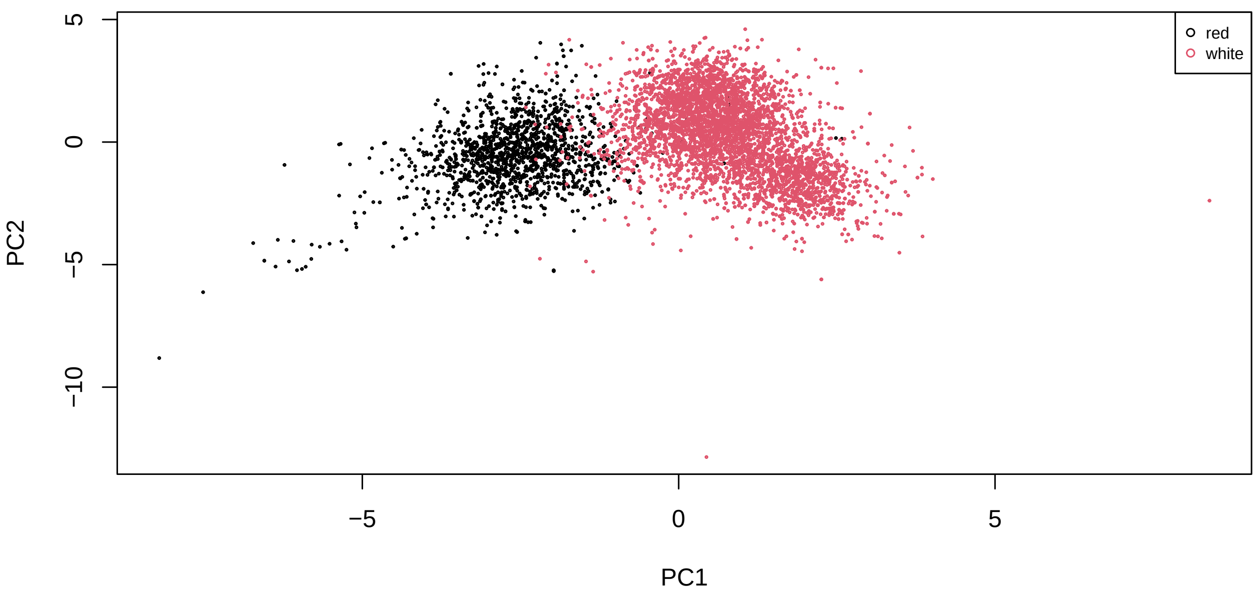
```
pc_wine <- prcomp(wines_nocol)
plot(PC2~PC1, data=pc_wine$x, cex=0.2)
```



Using the two dimensional representation of the data, we can see that the observations lie pretty close toghether and, therefore, it is indeed difficult to find a clear cluster pattern. From the humen eye it looks like two or three clusters might describe the data best.

l) Create again the same plot but this time colour the points according to whether they represent a white or red wine. (**Hint**: `wines$colour`).

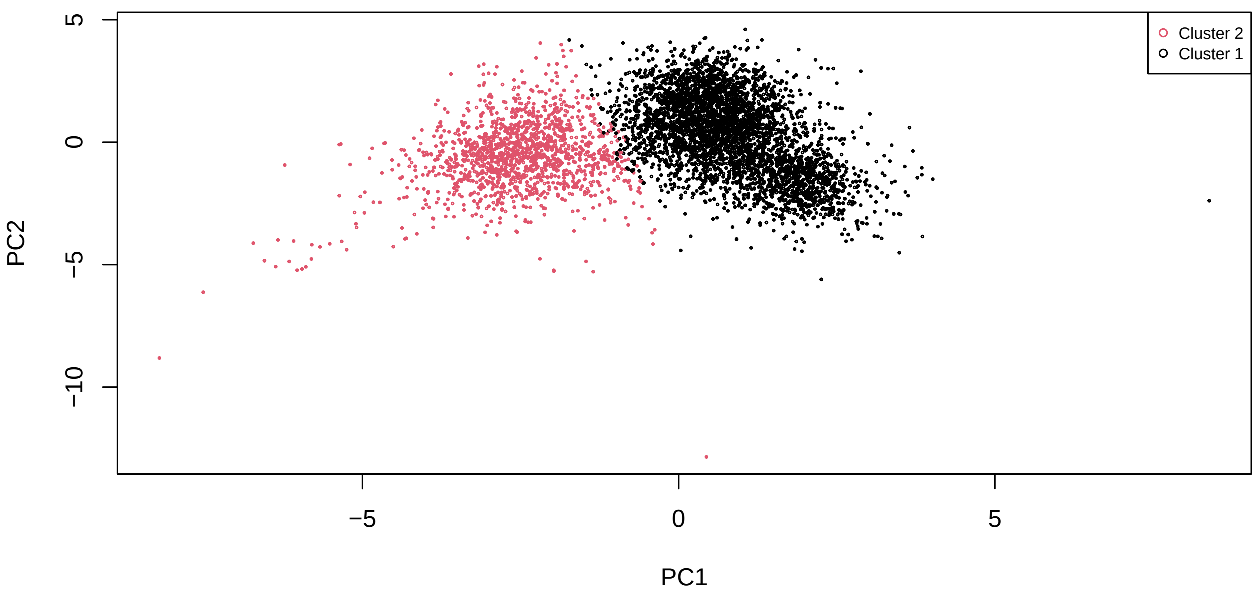
```r
plot(PC2~PC1, data=pc_wine$x, cex=0.2, col=wines$colour)
legend('topright', levels(wines$colour), pch=c(1,1),
       col=1:nlevels(wines$colour), cex=0.7)
```



Interestingly, the two clusters which we thought to identify by eye, more or less represent the colour of the wines.

m) Create the plot again but now colour the points according to the assigned cluster numbers of the two-cluster solution found with k-means in exercise g).

```r
plot(PC2~PC1, data=pc_wine$x, cex=0.2, col=km_2$cluster)
legend('topright', paste0('Cluster ', unique(km_2$cluster)),
       pch=1, col= unique(km_2$cluster), cex=0.7)
```



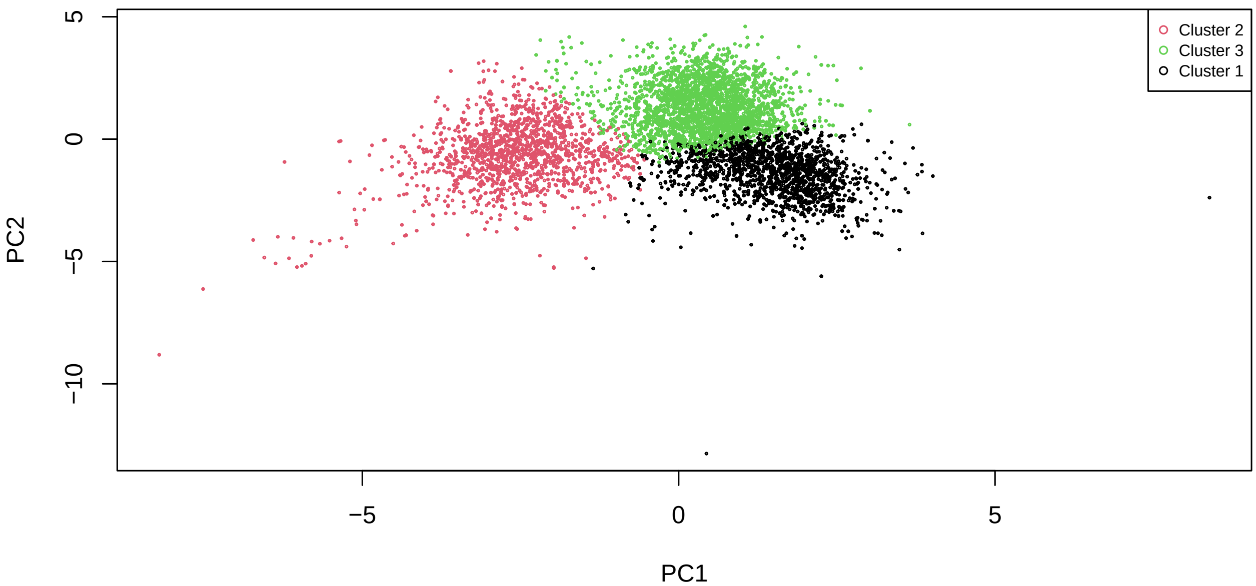
The cluster solution of kmeans also heavily overlaps with the wine colour. We can further see this when creating a confusion table of the wine colour and the assigned cluster numbers:

```
table(km_2$cluster, wines$colour)
```

```
##
##       red white
##    1   25  4804
##    2 1574    94
```

n) Visualize in the same way the three-cluster solution from exercise j).

```
plot(PC2~PC1, data=pc_wine$x, cex=0.2, col=km_3$cluster)
legend('topright', paste0('Cluster ', unique(km_3$cluster)),
       pch=1, col= unique(km_3$cluster), cex=0.7)
```



In the two-dimensional representation of the data, this solution seems perhaps a bit less appropriate compared to the solution with only two clusters.