Exercise 2

Tobias Raidl, 11717659

2023-10-22

Contents

```r
library("pgmm")
```

```
## Warning: package 'pgmm' was built under R version 4.1.3
```

```r
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```r
library("e1071")
```

```
## Warning: package 'e1071' was built under R version 4.1.3
```

```r
library("mclust")
```

```
## Warning: package 'mclust' was built under R version 4.1.3

## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
library("factoextra")
```

```
## Warning: package 'factoextra' was built under R version 4.1.3
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library("robCompositions")
```

```
## Warning: package 'robCompositions' was built under R version 4.1.3
```

```
## Loading required package: pls
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
## Loading required package: data.table
```

```
## Warning: package 'data.table' was built under R version 4.1.3
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## Registered S3 method overwritten by 'perry':
##   method         from
##   print.cvFolds cvTools
```

```r
library("cluster")
```

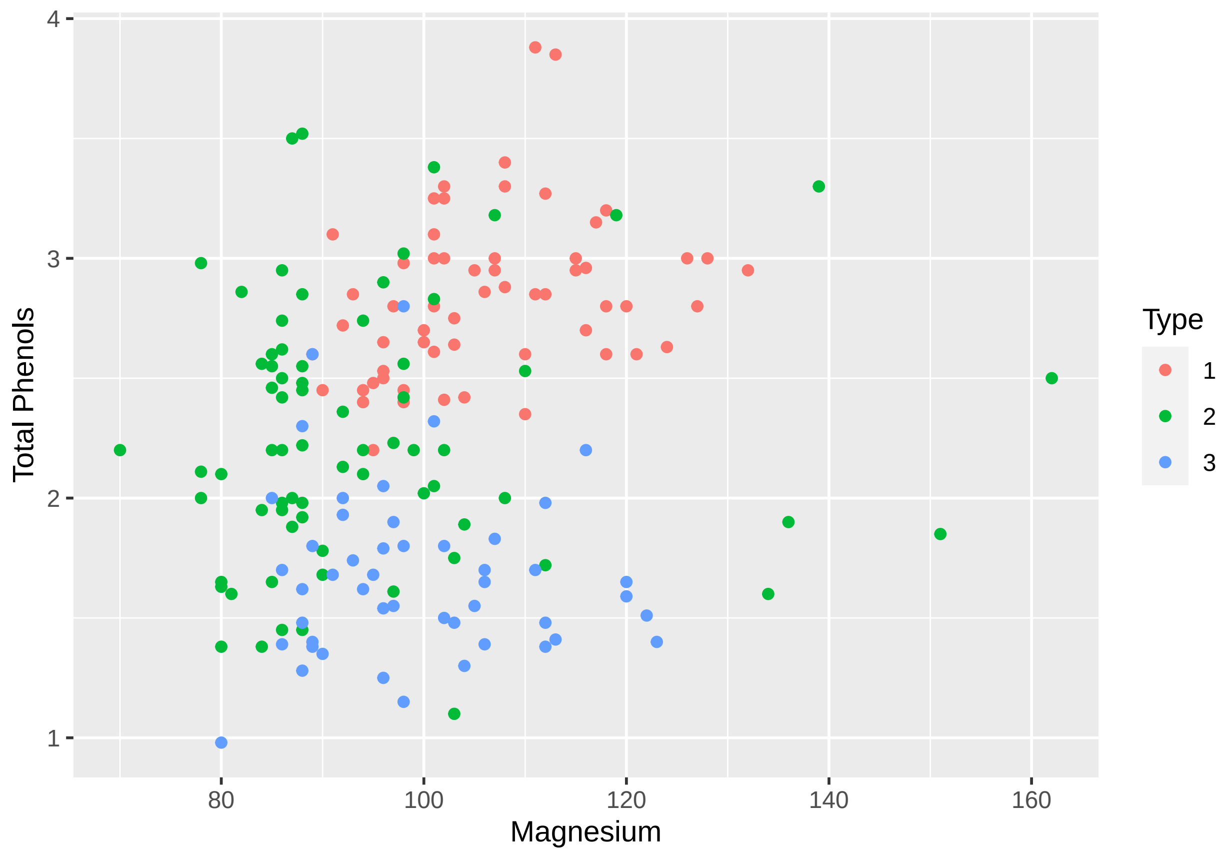
```
## Warning: package 'cluster' was built under R version 4.1.3
```

```r
data(wine)
wine$Type = as.factor(wine$Type)
```

Task 1

Applying k-means on scaled (standardized) variables makes sure to avoid distorted results. This can be the case wen te means or variances differ a lot between each variables. Non scaled variables cannot meaningfully be compared by distance. It is like stating 10km is less than 11m because 10<11.

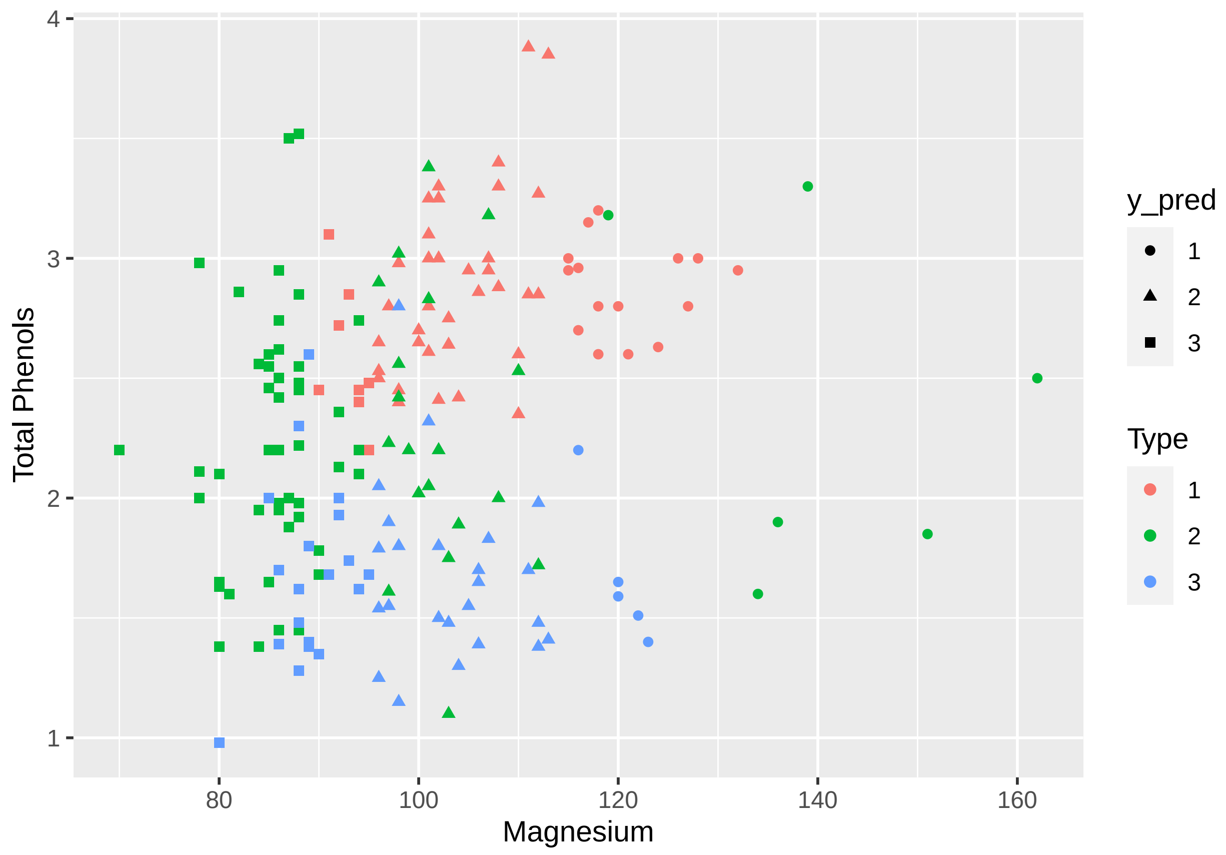
```r
df1 = select(wine, c("Type", "Magnesium", "Total Phenols"))
ggplot(wine, aes(x=Magnesium, y=`Total Phenols`, colour=Type)) +
  geom_point()
```



```
k = 3

res.km1 = kmeans(select(df1, -Type), k, nstart=10)
df1["y_pred"] = as.factor(res.km1$cluster)

ggplot(df1, aes(x=Magnesium, y=`Total Phenols`, colour=Type)) +
  geom_point(aes(shape=y_pred))
```



```r
table(df1$Type, df1$y_pred)
```

```
##
##       1  2  3
##   1 15 35  9
##   2  6 19 46
##   3  5 24 19
```

```r
matchClasses(table(df1$Type, df1$y_pred))
```

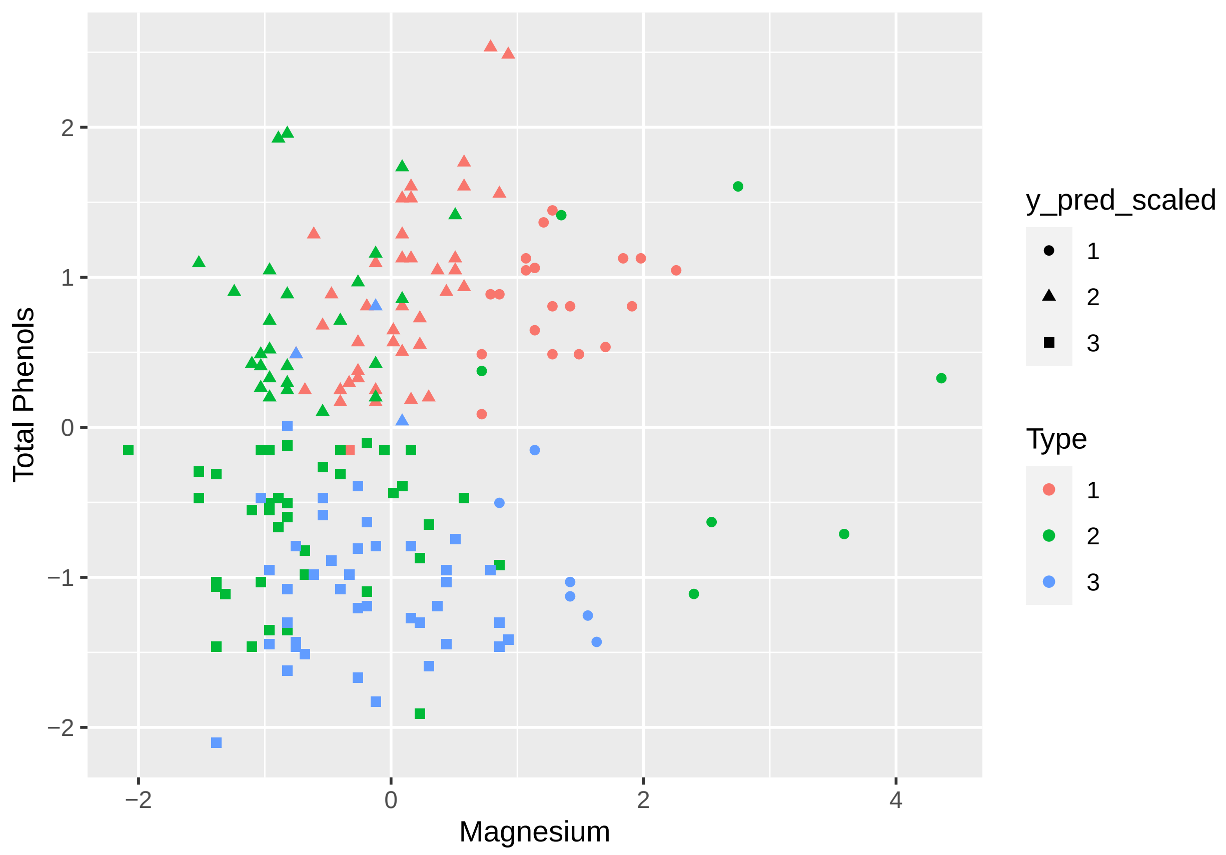
```
## Cases in matched pairs: 58.99 %
```

```
## 1 2 3
## 2 3 2
```

```r
df1 = select(wine, c("Type", "Magnesium", "Total Phenols"))
df1[,-c(1)] = scale(df1[,-c(1)])

res.km1.scaled = kmeans(select(df1, -Type), k, nstart=10)
df1["y_pred_scaled"] = as.factor(res.km1.scaled$cluster)

ggplot(df1, aes(x=Magnesium, y=`Total Phenols`, colour=Type)) +
  geom_point(aes(shape=y_pred_scaled))
```



```r
table(df1$Type, df1$y_pred_scaled)
```

```
##
##       1  2  3
##   1  19 39  1
##   2   7 26 38
##   3   6  3 39
```

```r
matchClasses(table(df1$Type, df1$y_pred_scaled))
```

```
## Cases in matched pairs: 65.17 %

## 1 2 3
## 2 3 3
```

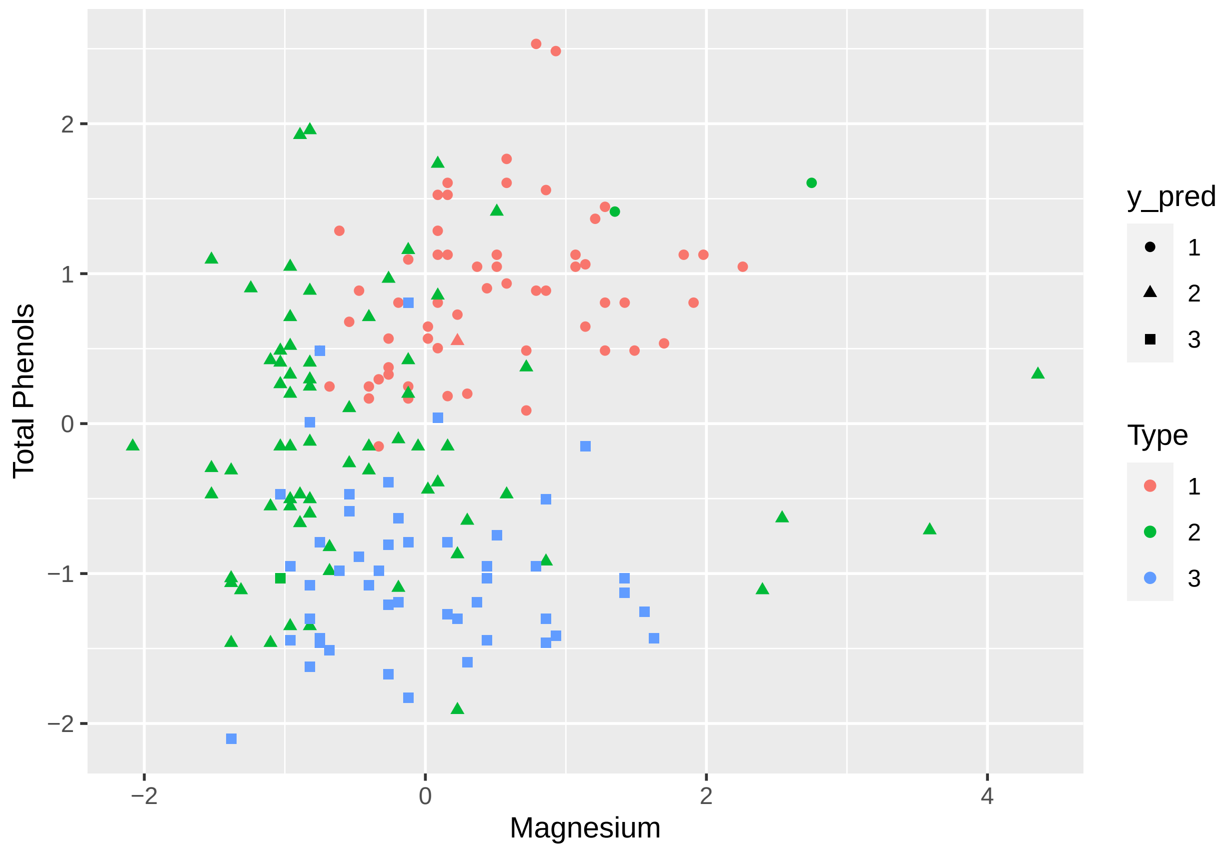
Task 2

```r
wine[,-c(1)] = scale(wine[,-c(1)])
df2 = wine

k = 3
df2.km2 = kmeans(select(df2, -Type), center=k)

df2["y_pred"] = as.factor(df2.km2$cluster)

ggplot(df2, aes(x=Magnesium, y=`Total Phenols`, colour=Type)) +
  geom_point(aes(shape=y_pred))
```



```
table(df2$Type, df2$y_pred)
```

```
##
##       1  2  3
##   1  58  1  0
##   2   2 68  1
##   3   0  0 48
```

```
matchClasses(table(df2$Type, df2$y_pred))
```

```
## Cases in matched pairs: 97.75 %
```

```
## 1 2 3
## 1 2 3
```

The variables with the highest sum of pairwise distances between each types cluster center are the ones that contain most information for clustering. In our case it is Flavanoids.

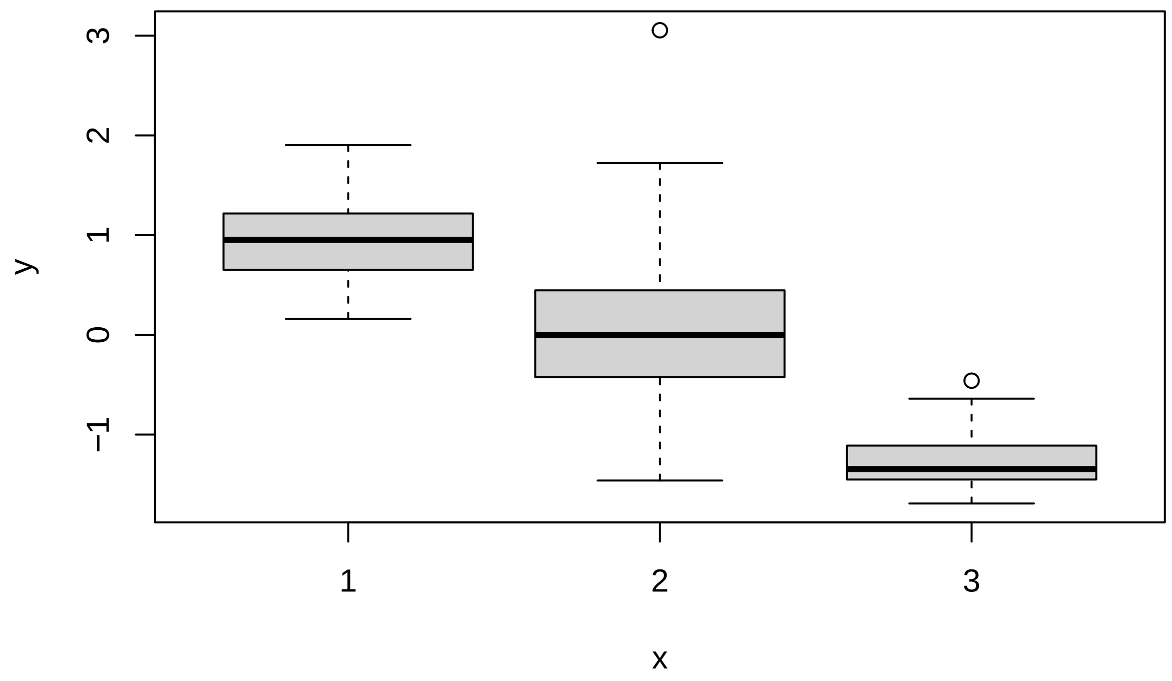
```
sum_of_pairwise_distances = function(vec) {
  sum(c(abs(vec[1]-vec[2]),abs(vec[1]-vec[3]),abs(vec[2]-vec[3])))
}

df2.km.centers = data.frame(df2.km2$centers)
sum_pairwise_distance = data.frame("sum_pairwise_dist" = t(apply(df2.km.centers, 2,
↪  FUN=sum_of_pairwise_distances)))
```

```
most_information = colnames(sum_pairwise_distance)[apply(sum_pairwise_distance, 1,
↪  which.max)]
cat("Most information provided by variable: ", most_information)
```

```
## Most information provided by variable:  sum_pairwise_dist.Flavanoids
```

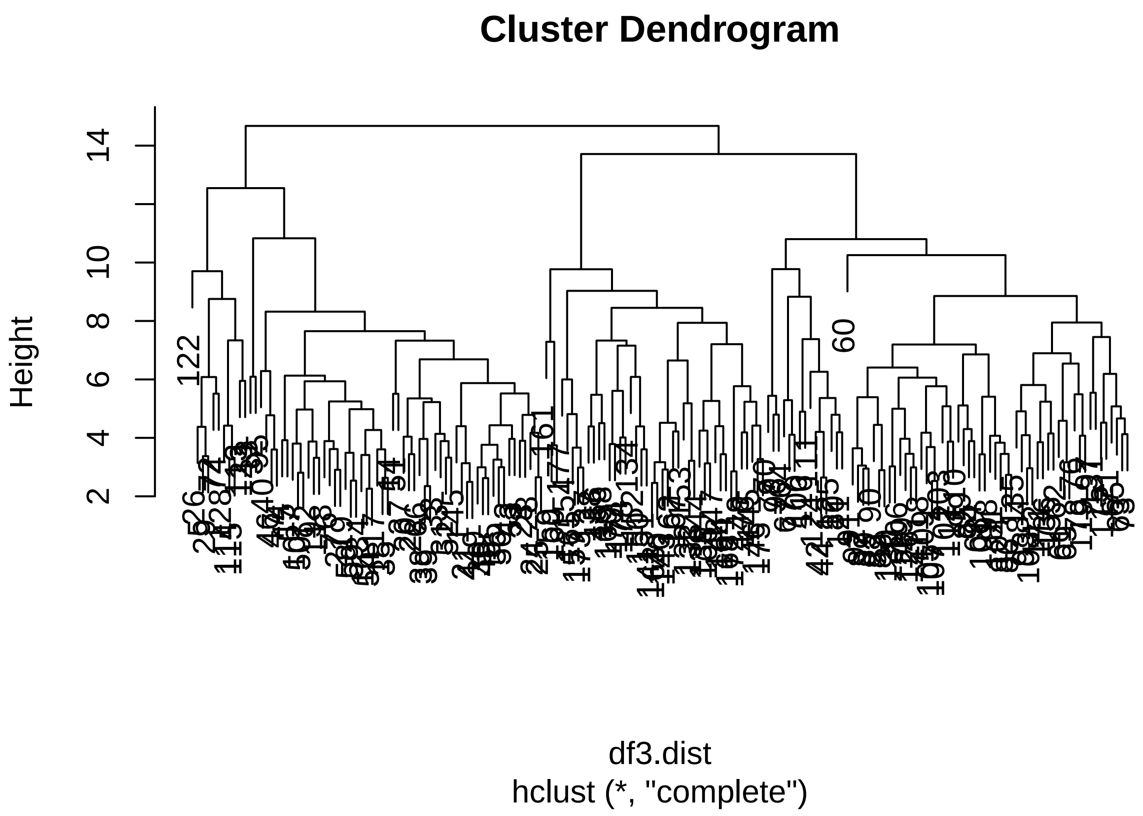
```
plot(x=wine$Type, y=wine$Flavanoids)
```



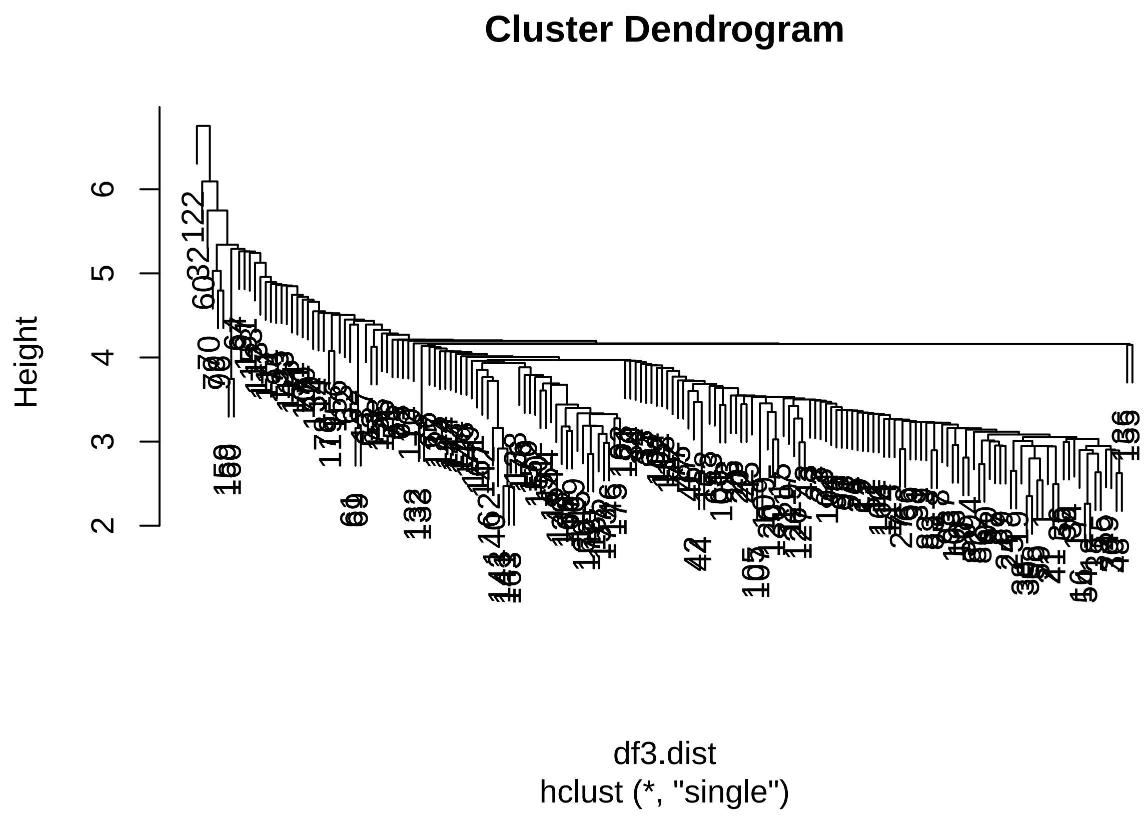
Task 3

Hierarchical clustering with metods single and centroid result in the same pair matching rate. Method complete is worse than these two methods.

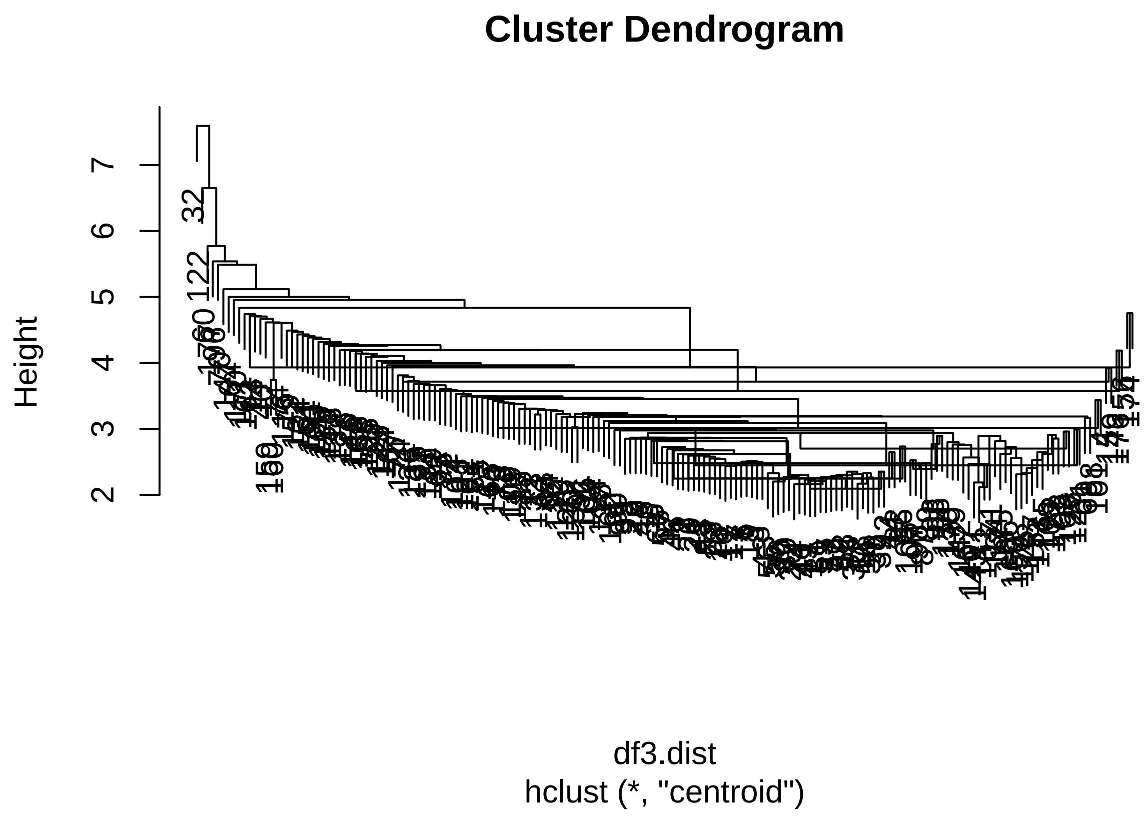
```
df3 = wine
df3.dist = dist(select(df3, -Type))
df3.hc.complete = hclust(df3.dist, method="complete")
df3.hc.single = hclust(df3.dist, method="single")
df3.hc.centroid = hclust(df3.dist, method="centroid")
plot(df3.hc.complete)
```



```
plot(df3.hc.single)
```



```
plot(df3.hc.centroid)
```



```r
df3["y_pred_complete"] = as.factor(cutree(df3.hc.complete, k=3))
df3["y_pred_single"] = as.factor(cutree(df3.hc.single, k=3))
df3["y_pred_centroid"] = as.factor(cutree(df3.hc.centroid, k=3))

cat("\nComplete:\n")
```

```
##
## Complete:
```

```r
matchClasses(table(df3$Type, df3$y_pred_complete))
```

```
## Cases in matched pairs: 89.89 %
```

```
## 1 2 3
## 1 2 3
```

```r
cat("\nSingle:\n")
```

```
##
## Single:
```

```r
matchClasses(table(df3$Type, df3$y_pred_single))
```

```
## Cases in matched pairs: 98.88 %
```

```
## 1 2 3
## 1 1 1
```

```
cat("\nCentroid:\n")
```

```
##
## Centroid:
```

```
matchClasses(table(df3$Type, df3$y_pred_centroid))
```

```
## Cases in matched pairs: 98.88 %
```

```
## 1 2 3
## 1 1 1
```

Task 4

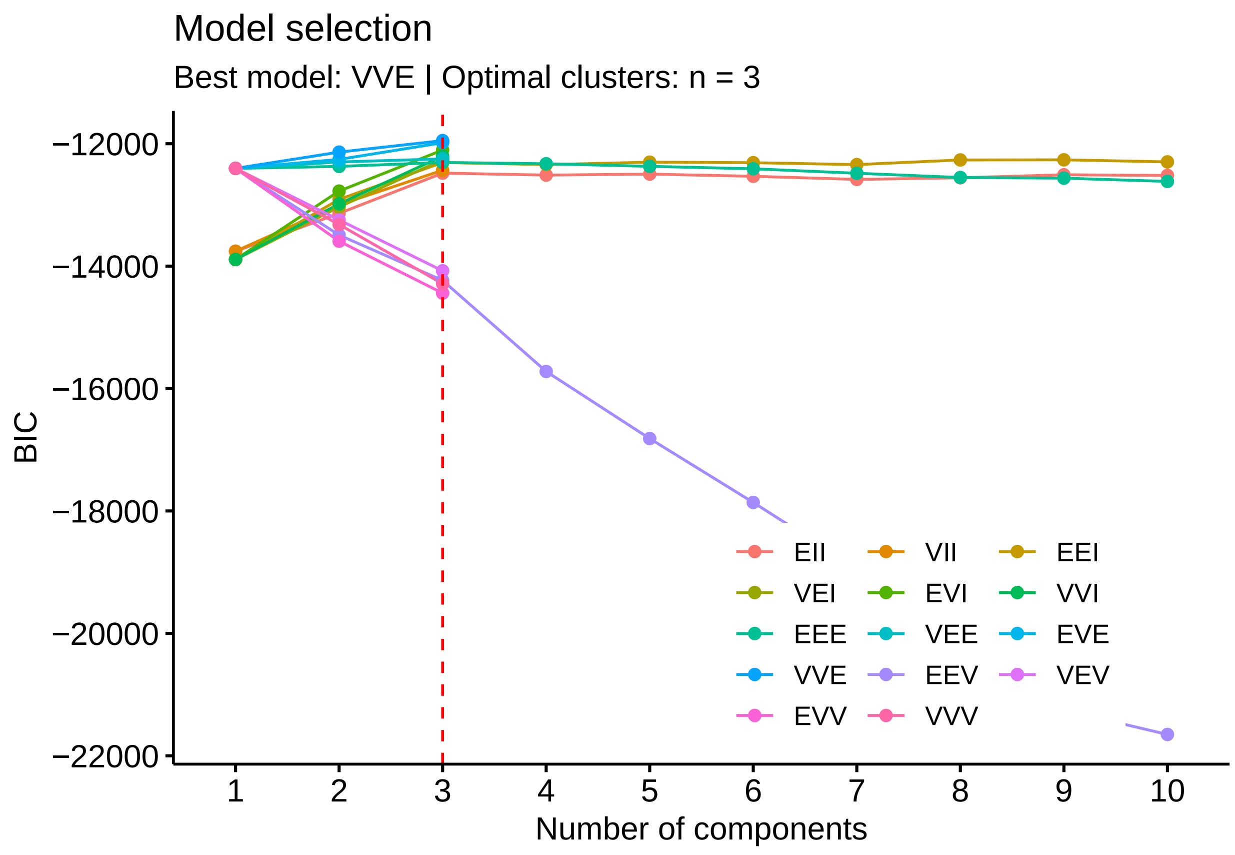
Optimal model is VVE with num. of clusters 3.

```
df4 = wine
df4.mc = Mclust(select(df4, -Type), 1:10)
summary(df4.mc)
```

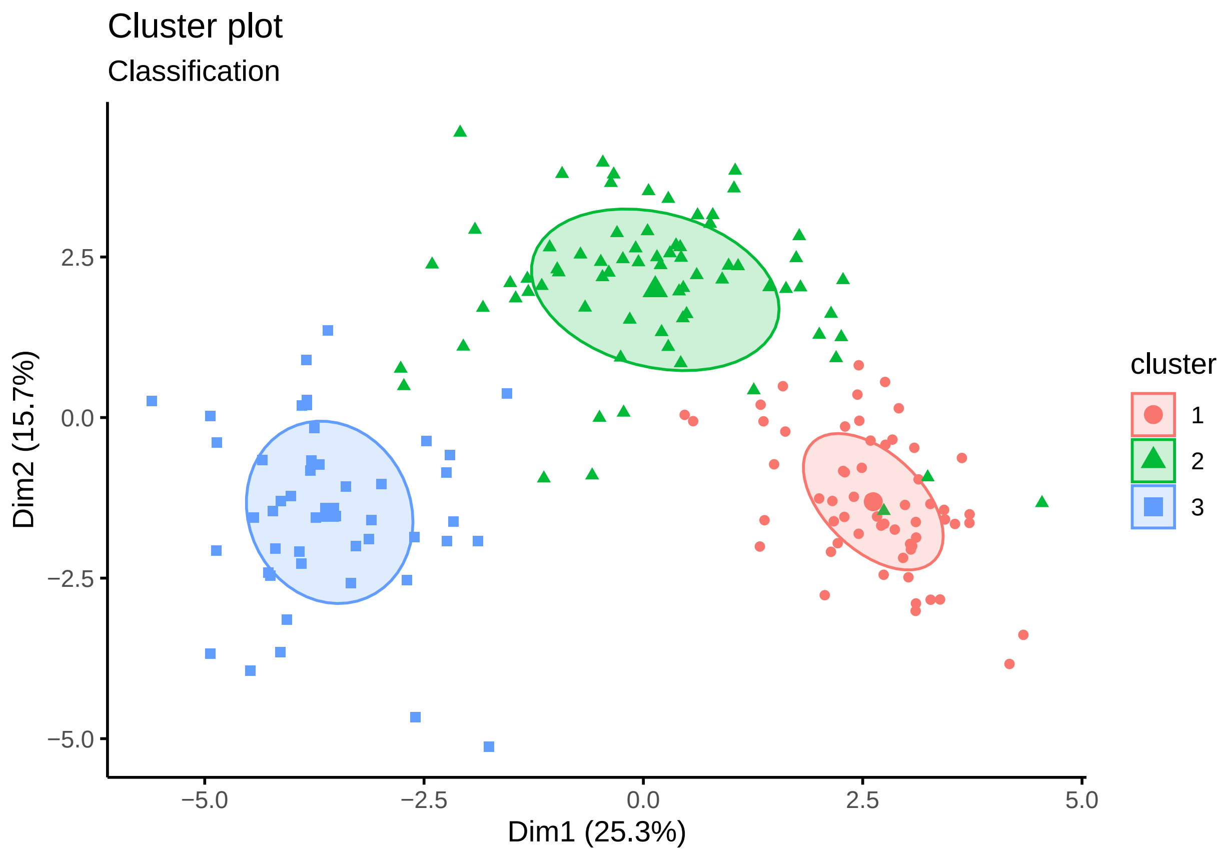
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVE (ellipsoidal, equal orientation) model with 3 components:
##
##  log-likelihood   n  df      BIC       ICL
##        -4640.44 178 515 -11949.5 -11949.52
##
## Clustering table:
##  1  2  3
## 59 73 46
```

```
fviz_mclust_bic(df4.mc)
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## i Please use `gather()` instead.
## i The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```r
fviz_mclust(df4.mc, "classification", geom = "point")
```



```
df4["y_pred"] = as.factor(df4.mc$classification)

table(df4$Type, df4$y_pred)
```

```
##
##       1  2  3
##    1 58  1  0
##    2  1 70  0
##    3  0  2 46
```

```
matchClasses(table(df4$Type, df4$y_pred))
```

```
## Cases in matched pairs: 97.75 %
```

```
## 1 2 3
## 1 2 3
```

Task 5

The ternary dendogram provides an overview of the cluster memberships for each observation. The color corresponds to the observations actual type. Observations of type 1 and 3 are predicted with a higher confidence than those of type 2, because their coefficients have a higher variance in the dendogram.

```
df5 = wine
df5.cm = cmeans(df5, centers=3)
df5$y_pred = as.factor(df5.cm$cluster)
```

```
table(df5$Type, df5$y_pred)
```

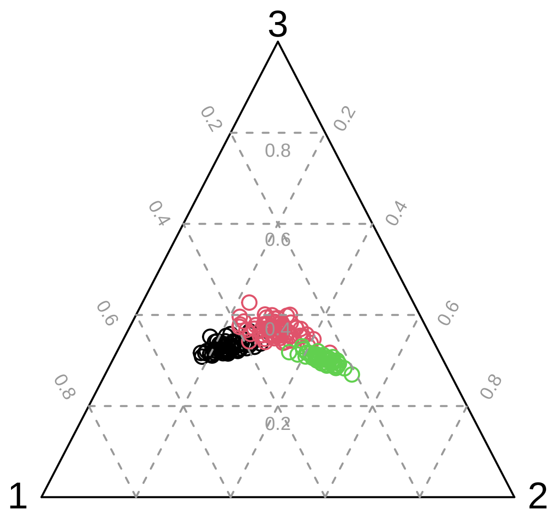
```
##
##       1  2  3
##   1 59  0  0
##   2  8 11 52
##   3  0 48  0
```

```
matchClasses(table(df5$Type, df5$y_pred))
```

```
## Cases in matched pairs: 89.33 %
```

```
## 1 2 3
## 1 3 2
```

```
ternaryDiag(df5.cm$membership, col=df5$Type)
```

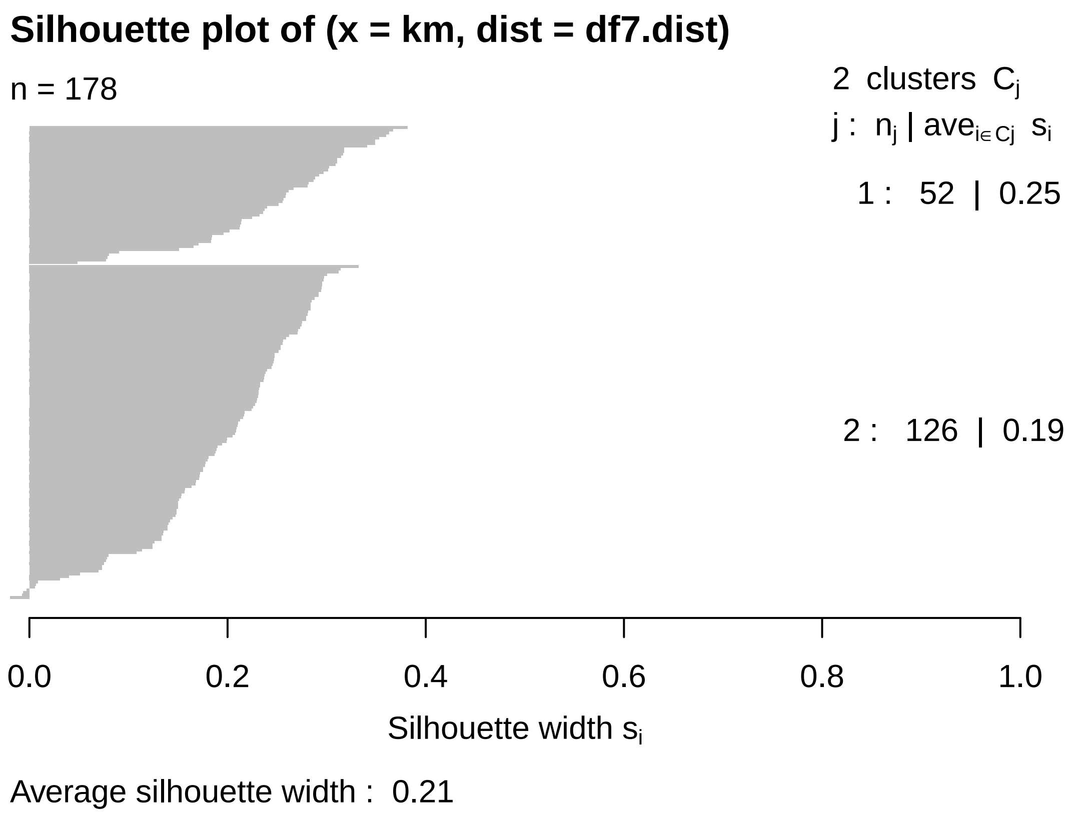


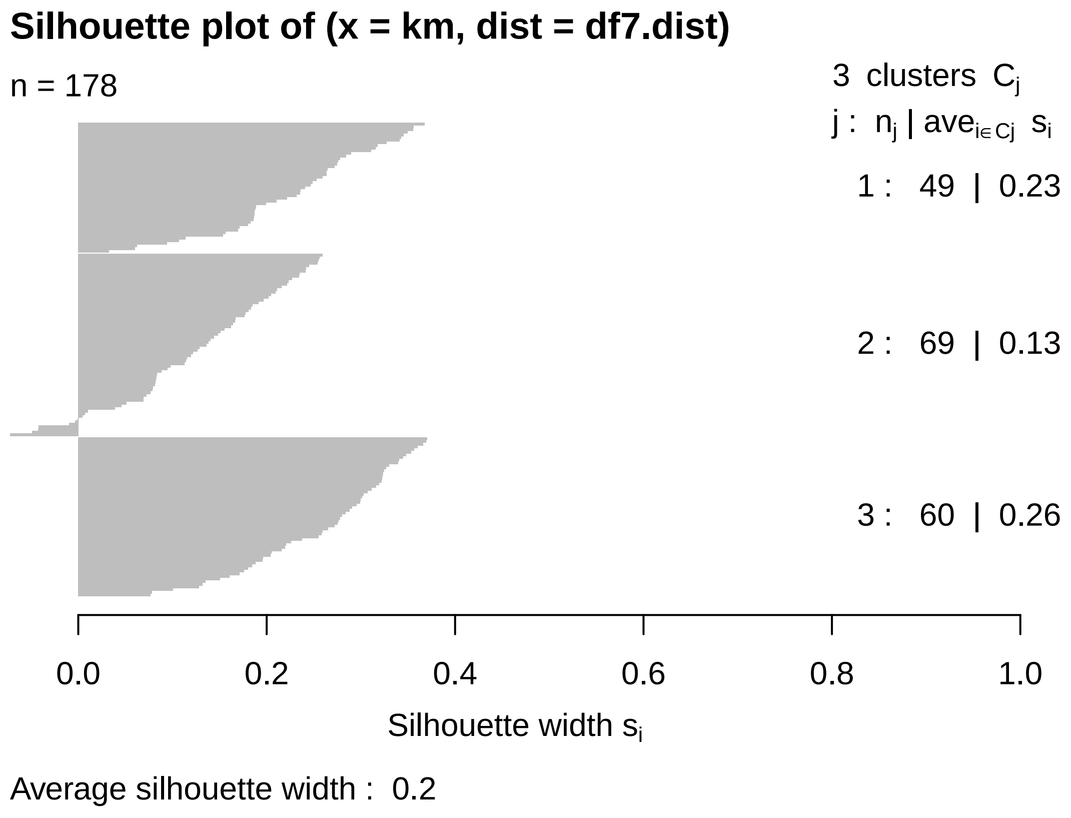
Task 7

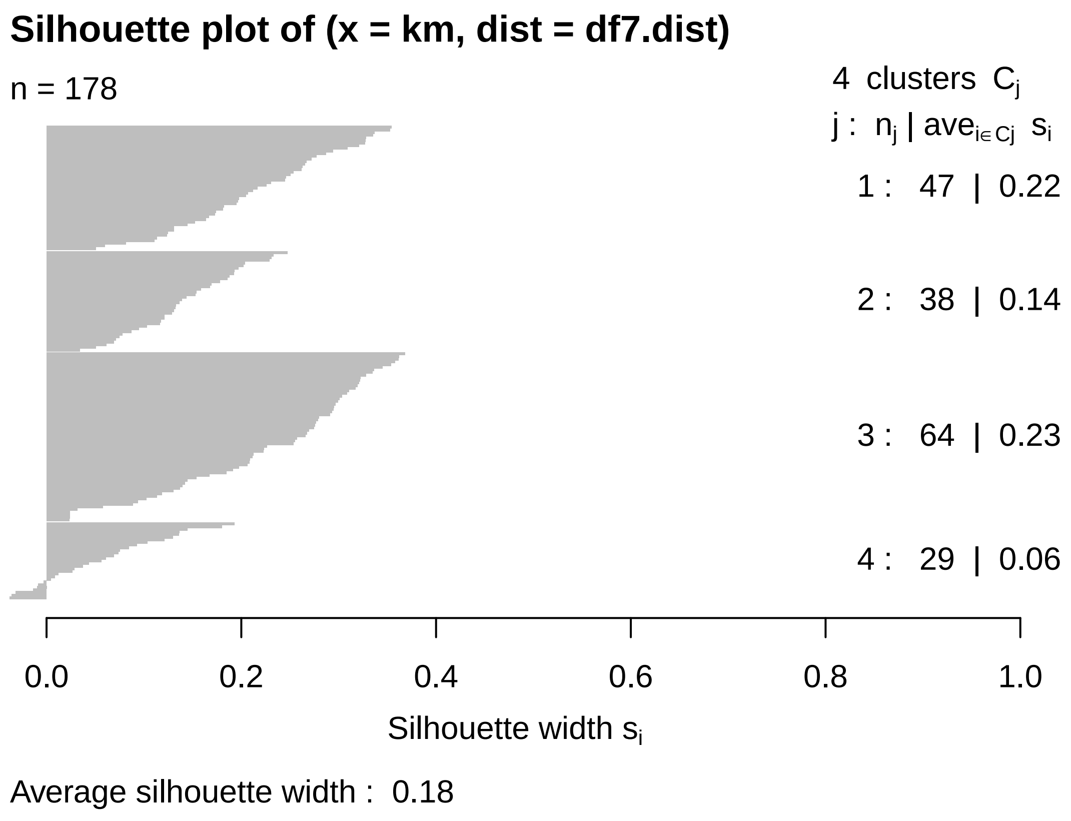
2 or 3 clusters have the lowest average silhouette width and therefore fit the requirements for optimal num. of cluster.

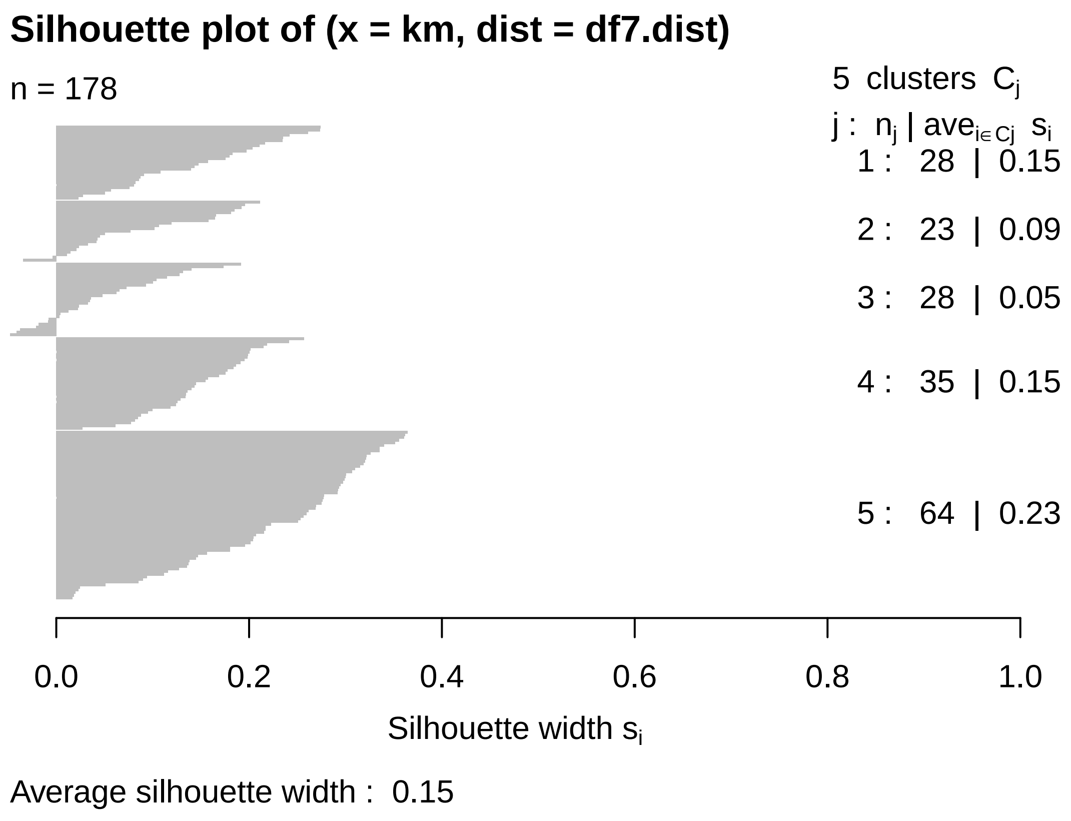
```
df7 = wine
df7.dist = dist(select(df7, -Type))

for(k in 2:6) {
  km = unlist(kmeans(select(df7, -Type), center=k, nstart=10)$cluster)
  plot(silhouette(km, df7.dist))
}
```

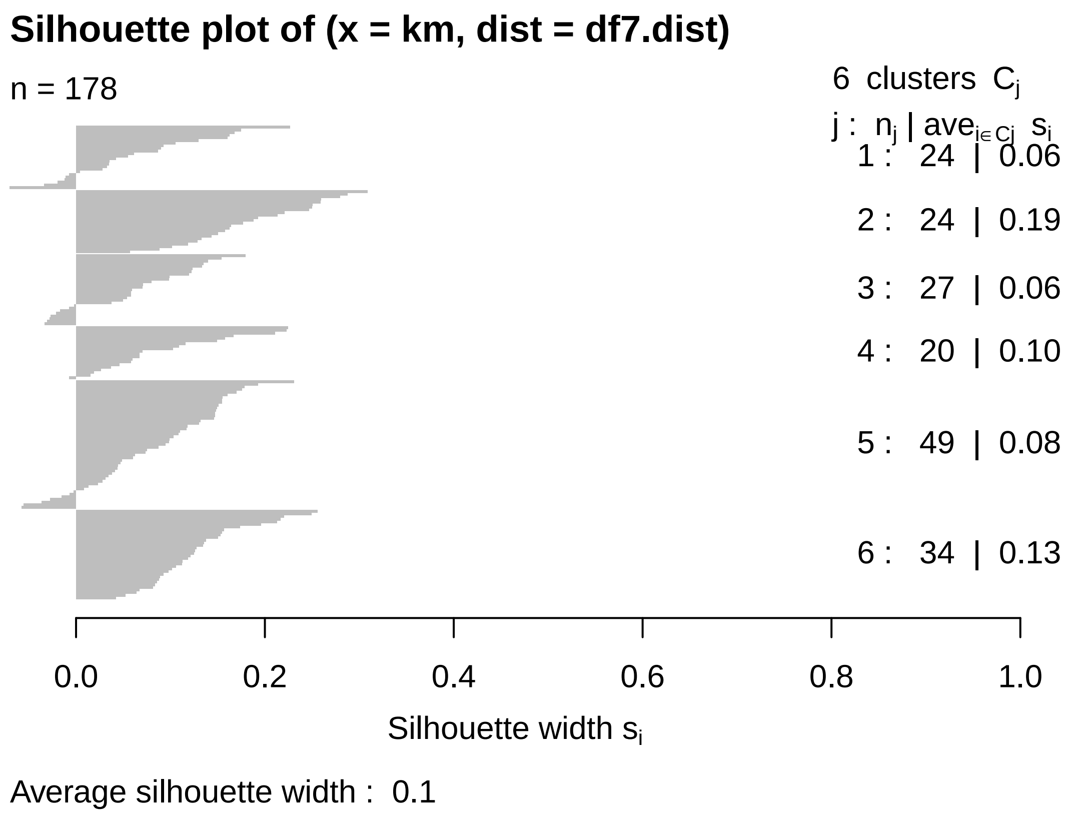



Silhouette plot of (x = km, dist = df7.dist)
n = 178
3 clusters C_j
j : n_j | $ave_{i \in Cj}$ s_i
1 : 49 | 0.23
2 : 69 | 0.13
3 : 60 | 0.26
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.2


Silhouette plot of (x = km, dist = df7.dist)
n = 178
4 clusters C_j
j : n_j | $ave_{i \in Cj}$ s_i
1 : 47 | 0.22
2 : 38 | 0.14
3 : 64 | 0.23
4 : 29 | 0.06
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.18



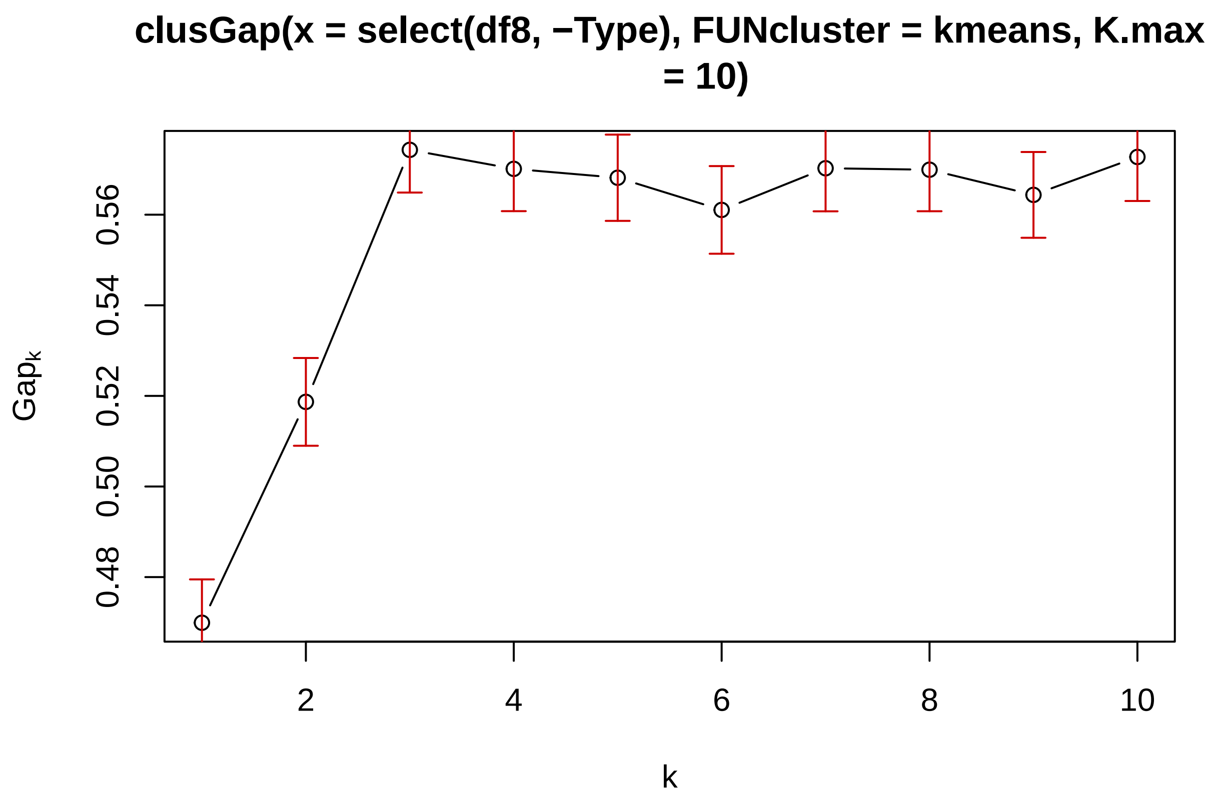
Silhouette plot of (x = km, dist = df7.dist)
n = 178
5 clusters C_j
j : n_j | $\text{ave}_{i\in C_j}$ s_i
1 : 28 | 0.15
2 : 23 | 0.09
3 : 28 | 0.05
4 : 35 | 0.15
5 : 64 | 0.23
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.15



Task 8

The optimal num. of clusters is dependent on the correct classification - model complexity tradeoff. Lower k corresponds to lower complexity. Therefore 3 clusters seem to be the optimal fit.

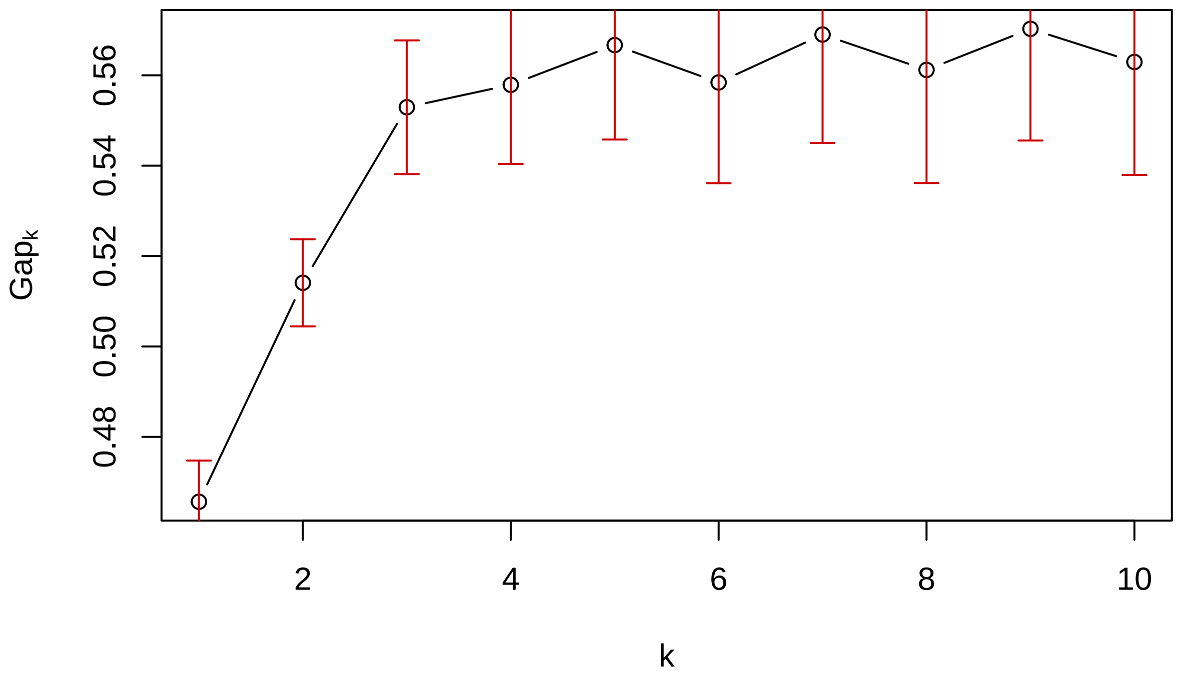
```
df8 = wine
df8.cg = clusGap(select(df8, -Type), FUN=kmeans,K.max=10)
plot(df8.cg)
```



# Task 9

```
df9 = wine
df9.cg <- clusGap(select(df9, -Type), FUNcluster = cmeans, K.max = 10)
plot(df9.cg)
```

clusGap(x = select(df9, −Type), FUNcluster = cmeans, K.max = 10)


Gap_k
k