Midterm 2: Clustering & SVM's

Advanced Topics in Data Science II
Harvard University, Spring 2017

Tim Hagmann

April 10, 2017

Contents

Introduction

In this exam we're asking you to work with measurements of genetic expression for patients with two related forms of cancer: Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML). We ask you to perform two general tasks: (1) Cluster the patients based only on their provided genetic expression measurements and (2) classify samples as either ALL or AML using Support Vector Machines.

In the file `MT2_data.csv`, you are provided a data set containing information about a set of 72 different tissue samples. The data have already been split into training and testing when considering the SVM analyses, as the first column indicates. The first 34 samples will be saved for testing while the remaining 38 will be used for training. Columns 2-4 contain the following general information about the sample:

- ALL.AML: Whether the patient had AML or ALL.
- BM.PB: Whether the sample was taken from bone marrow or from peripheral blood.
- Gender: The gender of the patient the sample was obtained from.

Note that some of the samples have missing information in these columns. Keep this in mind when conducting some of the analyses below. The remaining columns contain expression measurements for 107 genes. You should treat these as the features. The genes have been pre-selected from a set of about 7000 that are known to be relevant to, and hopefully predictive of, these types of cancers.

Problem 1: Clustering [60 points]

For the following, **you should use all 72 samples** – you will only use the genetic information found in columns 5-111 of the dataset. The following questions are about performing cluster analysis of the samples using only genetic data (not columns 2-4).

Preparation

In the following code chunk all the necessary setup for the modelling environment is done.

```
## Options
options(scipen = 10)                          # Disable scientific notation
update_package <- FALSE                       # Use old status of packages

## Init files (always execute, eta: 10s)
source("scripts/01_init.R")                   # Helper functions to load packages
source("scripts/02_packages.R")               # Load all necessary packages
source("scripts/03_functions.R")              # Load project specific functions
```

Load the data

The first step is to load the data and construct a small subset of all the genetic data.

```
## Read data
df_genes <- data.frame(read_csv("data/MT2_data.csv"))

# Reformat
names(df_genes) <- tolower(names(df_genes))

# Select
df_clust <- df_genes[, 5:111]
```

There are 72 observations (72 tissue samples) and 111 variables present. 107 out of the 111 contain expression measurements for 107 genes. The data is split in the 'train.test' variable into a training and testing set. The variable 2:4 contain information about the leukemia type (AML or ALL), Whether the sample was taken from bone marrow or from peripheral blood and the gender of the patient.

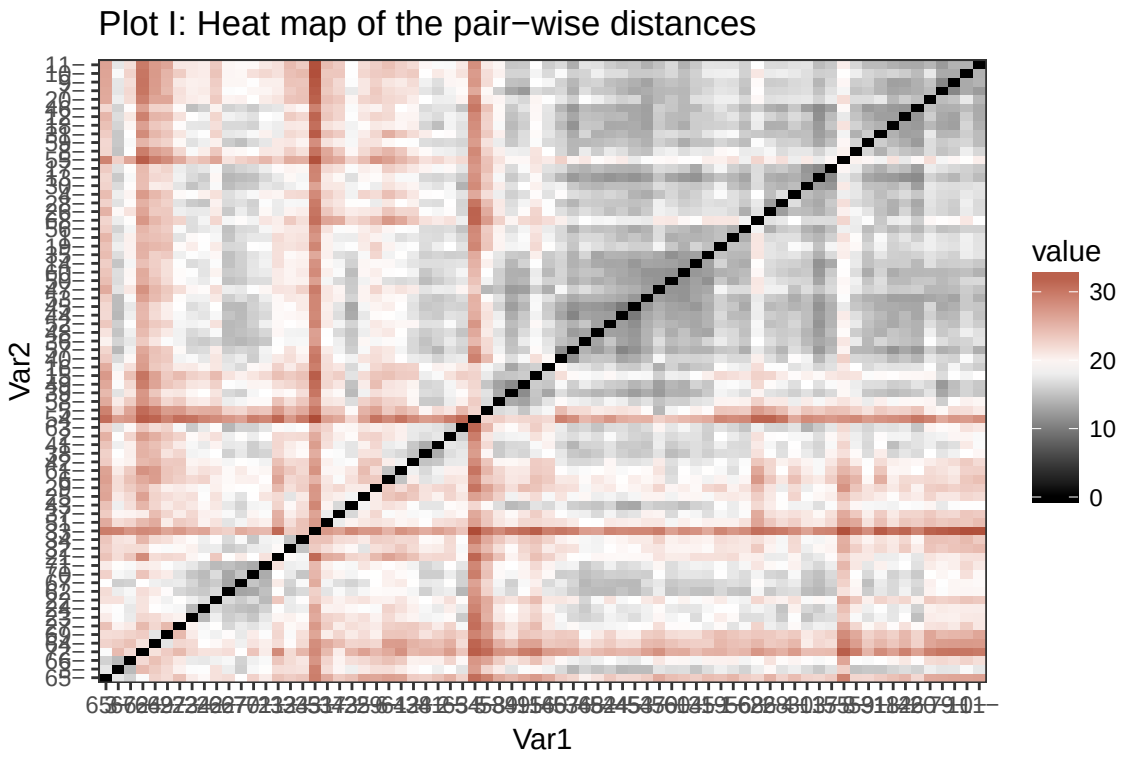
Question A.) (10 points)

Standardize the gene expression values, and compute the Euclidean distance between each pair of genes. Apply multi-dimensional scaling to the pair-wise distances, and generate a scatter plot of genes in two dimension. By visual inspection, into how many groups do the genes cluster? If you were to apply principal components analysis to the standardized data and then plot the first two principal components, how do you think the graph would differ? Briefly justify. (you do not need to perform this latter plot)

Rescaling

In the following part the data is rescaled and the Euclidean distance is calculated. Furthermore a heat map of the pair-wise distances is produced (not necessary but helpful).

```
dist_euclidean <- daisy(df_clust, metric="euclidean", stand=TRUE)
fviz_dist(dist_euclidean,
          gradient=list(low="black", mid="white", high="darkred"),
          lab_size=7) +
  ggtitle("Plot I: Heat map of the pair-wise distances") +
  theme_bw()
```



The above plot indicates, that there are different Clusters present. Those are marked trough the red color.

Apply MDS

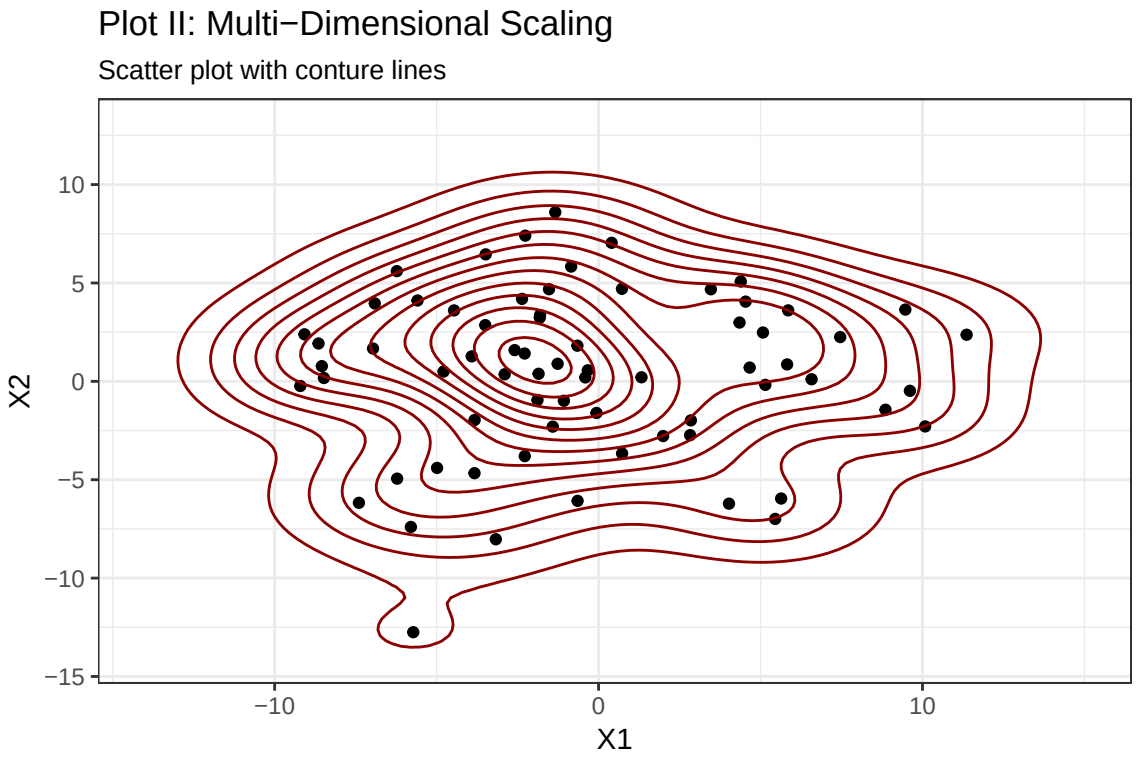
Apply multi-dimensional scaling to the pair-wise distances, and generate a scatter plot of the genes in two dimension.

Calculate multi-dimensional scaling (MDS)

```
vec_mds <- cmdscale(dist_euclidean)
colnames(vec_mds) <- c("X1", "X2")
```

Visualize

```
ggplot(data=vec_mds, aes(x=X1, y=X2)) +
  geom_point() +
  geom_point(size=0.9) +
  geom_density2d(color="darkred") +
  labs(title="Plot II: Multi-Dimensional Scaling",
  subtitle="Scatter plot with conture lines") +
  xlim(-14, 15) +
  ylim(-14, 13) +
  theme_bw()
```



The above scatter plot with added conture lines identifies only one cluster at the point (0, 0), however to the naked eye, there is a second cluster visible around the point (5, 2). Furthermore, a outlier is visible in the bottom part of the plot. This indicates that there might be more clusters than the one visualized above.

Difference to a PCA plot

It can be shown mathematicly that PCA scores and classical MDS have equivalent scores. It can therefore be assumed that the 2D dimension plot and the PCA plot look very similar, but they would probably not be at the same level. This would be reflected across 0 on the PC1/Dim1 axis. This might be seen by plot being on the opposite side of each other.

Question B.) (10 points)

Apply **Partitioning around medoids** (PAM) to the data, selecting the optimal number of clusters based on the Gap, Elbow and Silhouette statistics – if the they disagree, select the largest number of clusters indicated by the statistics. Summarize the results of clustering using a principal components plot, and comment on the quality of clustering using a Silhouette diagnostic plot.

Partitioning around medoids (PAM)

```
gapstat_pam <- clusGap(scale(df_clust),FUN=pam, K.max=10, B=500)
```

Print output

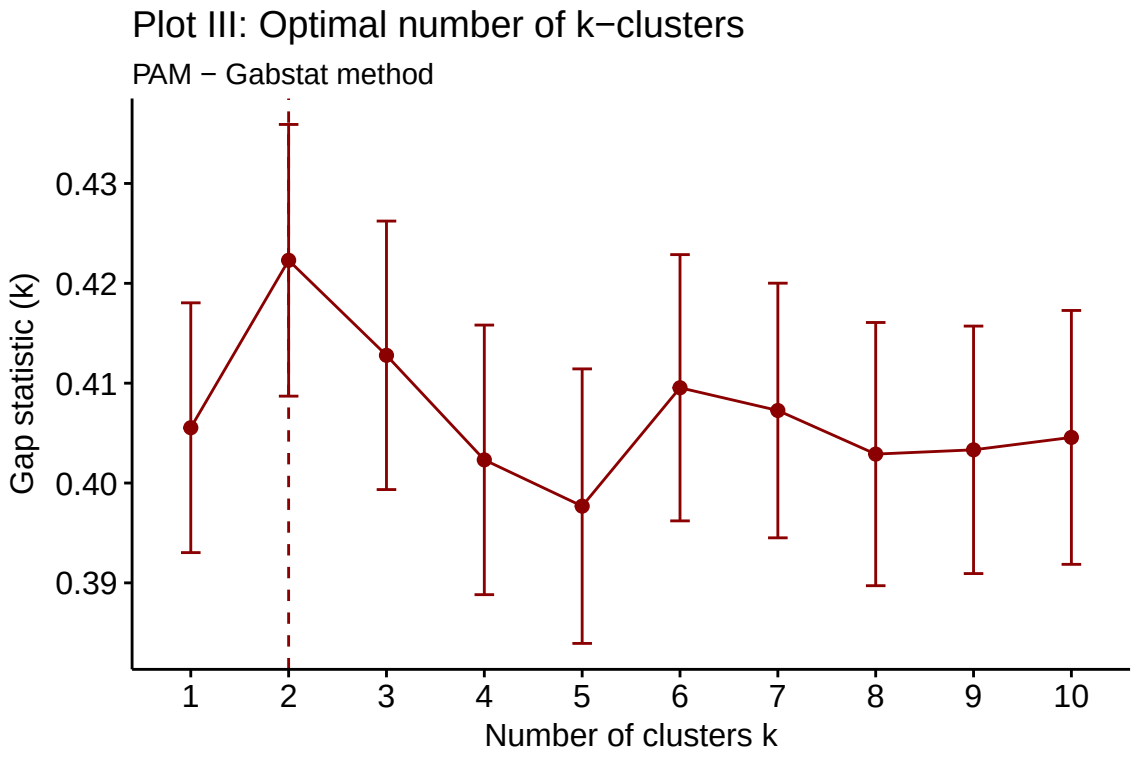
```
print(gapstat_pam, method = "Tibs2001SEmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_clust), FUNcluster = pam, K.max = 10, B = 500)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 2
##           logW   E.logW       gap     SE.sim
##  [1,] 5.539804 5.945169 0.4053649 0.01343121
##  [2,] 5.469942 5.891741 0.4217985 0.01518380
##  [3,] 5.435627 5.848800 0.4131730 0.01529404
##  [4,] 5.410186 5.813175 0.4029889 0.01487890
##  [5,] 5.383720 5.781244 0.3975244 0.01398347
##  [6,] 5.343561 5.752287 0.4087252 0.01330127
##  [7,] 5.317549 5.724415 0.4068656 0.01346751
##  [8,] 5.295769 5.697813 0.4020434 0.01295911
##  [9,] 5.269276 5.672146 0.4028701 0.01298306
## [10,] 5.243024 5.647237 0.4042131 0.01270216
```

The Tibshirani criterion has the idea that for a particular choice of K clusters, the total within cluster variation is compared to the expected within-cluster variation. According to this criterion two clusters are optimal.

PAM gap statistic

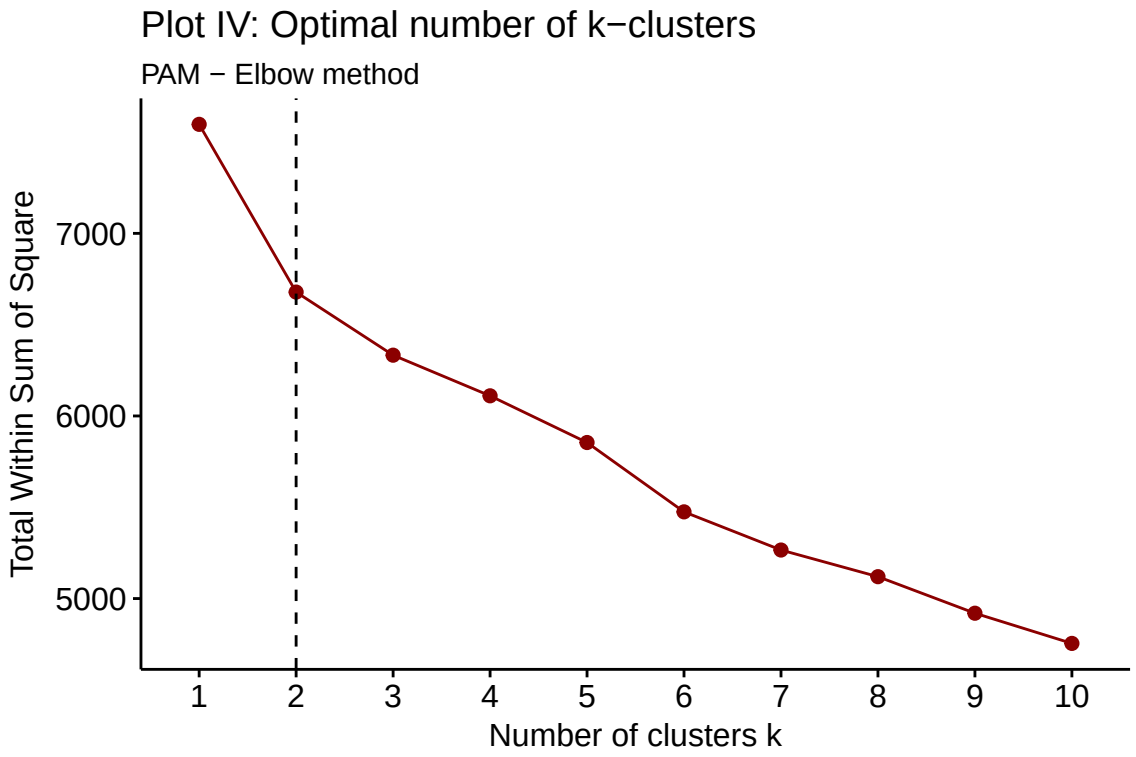
```
# Calculation
nbc_clust_gap <- fviz_nbclust(scale(df_clust), pam,
                 method="gap_stat", linecolor="darkred",
                 k.max=10)

# Visualization
nbc_clust_gap +
  labs(title="Plot III: Optimal number of k-clusters",
  subtitle="PAM - Gabstat method")
```



The above plot shows, that the optimal number of clusters according to the gap statistic is 2 clusters.

Elbow plot

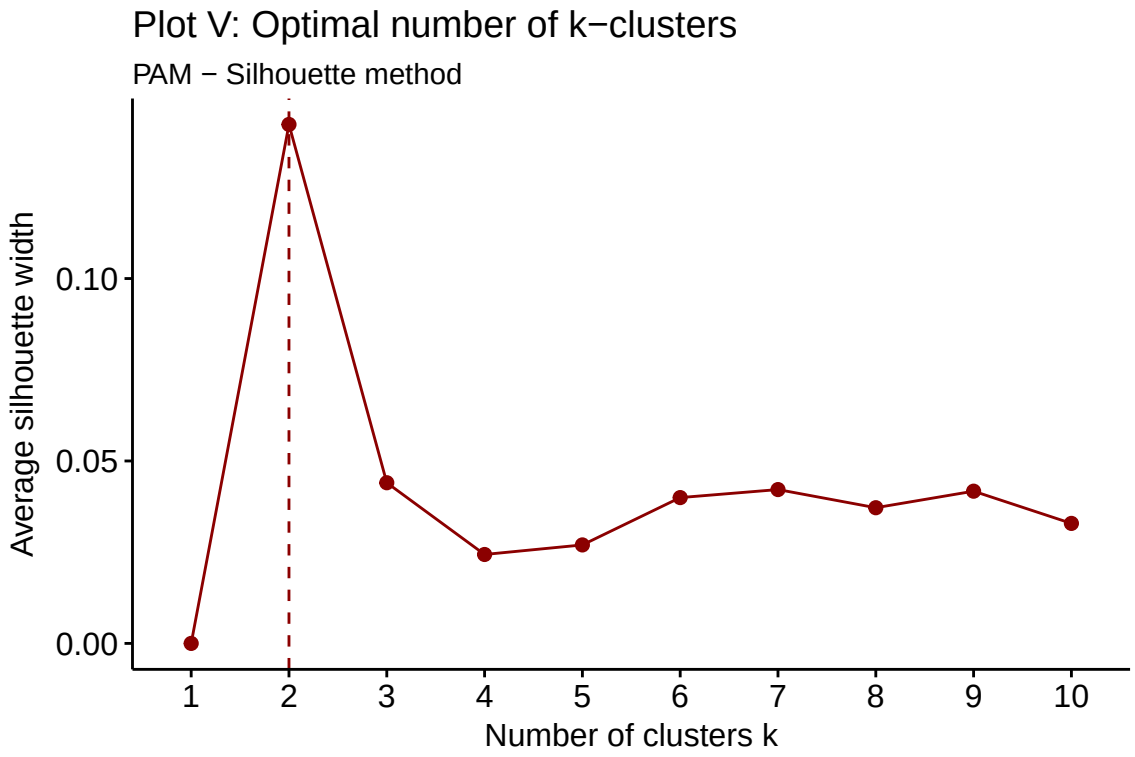
```
# Calculation
nbc_clust_elbow <- fviz_nbclust(scale(df_clust), pam,
                                method="wss", linecolor="darkred")

# Visualization
nbc_clust_elbow +
  geom_vline(xintercept=2,linetype=2) +
  labs(title="Plot IV: Optimal number of k-clusters",
  subtitle="PAM - Elbow method")
```



The above plot shows, that the optimal number of clusters is again at 2. This is indicated by the elbow that arises at the 2 cluster mark.

PAM silouhette

```
# Calculation
nbc_clust_sil <- fviz_nbclust(scale(df_clust), pam,
                 method="silhouette", linecolor="darkred",
                 k.max=10)

# Visualization
nbc_clust_sil +
  labs(title="Plot V: Optimal number of k-clusters",
  subtitle="PAM - Silhouette method")
```



According to the silhouette plot above the optimal number of clusters is again at 2.

Summary

The different methods for the pam algorithm do agree on the optimal amount of clusters. The gapstat, elbow and silhouette methods all indicate 2 clusters.

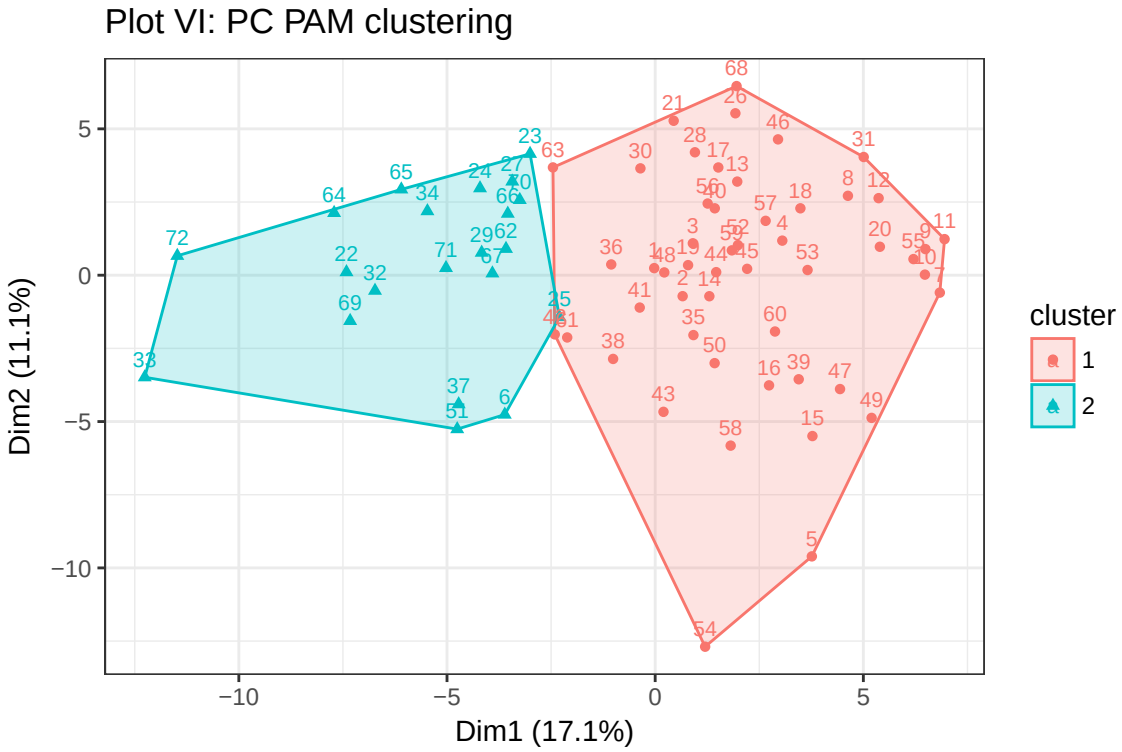
Calculate

The next thing is to summarize the results of clustering using a principal components plot.

```
# Options
opt_clust <- 2
sel_pam <- pam(scale(df_clust), opt_clust)

# Calculation
fviz_clust_pam <- fviz_cluster(sel_pam,
                               data=scale(df_clust),
                               linecolor="darkred",
                               labelsize=8)
```

Visualize PAM

```
fviz_clust_pam +
  ggtitle("Plot VI: PC PAM clustering") +
  theme_bw()
```



The above principal component plot visualizing the two cluster across the first and second dimension. It gives a nice picture about the cluster lines. There appears to be a small overlap at the 25/42 point. Otherwise the clusters appear to be properly seperated on the first two PC's. Cluster one appears to be bigger than cluster 2.

Misclassified genes

```
# Calculate
sil_pam <- silhouette(sel_pam)[, 3]

# Print missclassified genes
print("PAM misclassified genes:")
```

```
## [1] "PAM misclassified genes:"
```

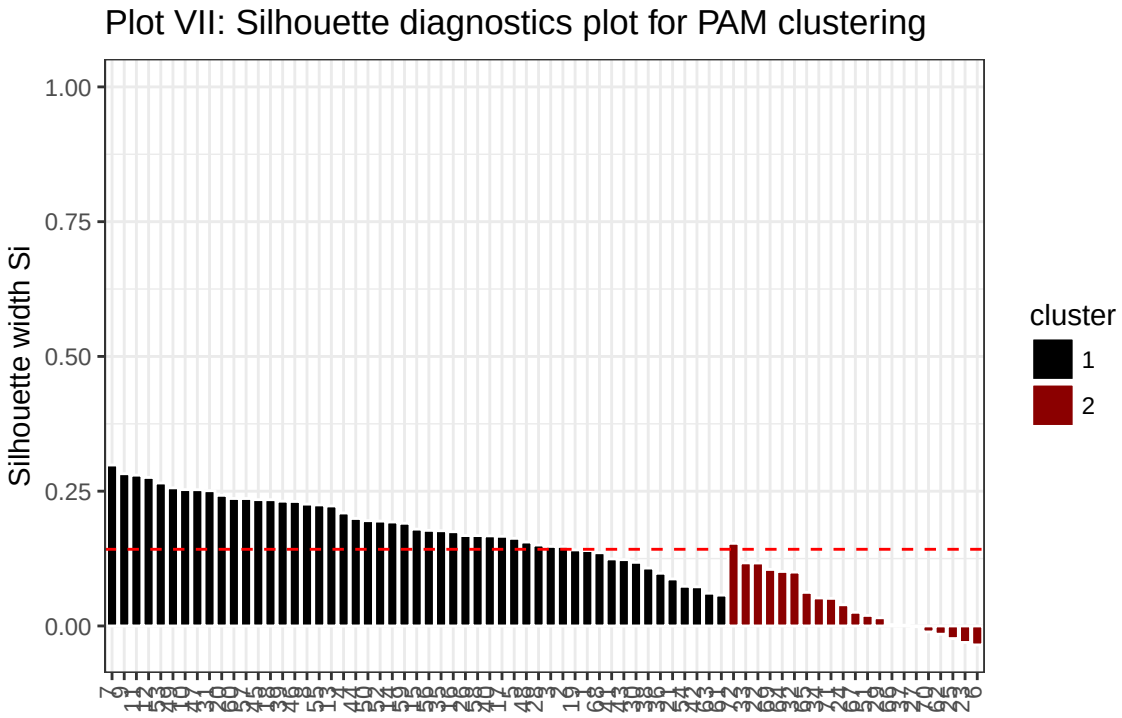
```
print(names(which(sil_pam < 0)))
```

```
## [1] "27" "70" "62" "25" "23" "6"
```

PAM silhouette

```
fviz_silhouette(silhouette(sel_pam),
                main="Plot VII: Silhouette diagnostics plot for PAM clustering") +
  theme_bw() +
  scale_fill_manual(values=c("black", "darkred")) +
  scale_color_manual(values=c("white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

```
##   cluster size ave.sil.width
## 1       1   51          0.18
## 2       2   21          0.04
```



The above plot shows, that the PAM algorithm probably misclassified the genes 27, 70, 62, 25, 23 and 6 into the wrong clusters. The first cluster appears to be the stronger of the two with the majority of its point lying well above the dotted line. The case for cluster two appears to be slightly weaker as a portion is likely to be misclassified. Furthremore, the majority of the points are lying below the dotted line.

Question C.) (10 points)

Apply **Agglomerative clustering** (AGNES) with Ward's method to the data. Summarize the results using a dendrogram. Determine the optimal number of clusters in a similar way as in (b), and add rectangles to the dendrograms sectioning off clusters. Comment on the ways (if any) the results of PAM differ from those of AGNES.

Agglomerative clustering

```
gapstat_agnes <- clusGap(scale(df_clust),
                         FUN=agnes.reformat,
                         K.max=10, B=500)
print(gapstat_agnes, method = "Tibs2001SEmax")
```

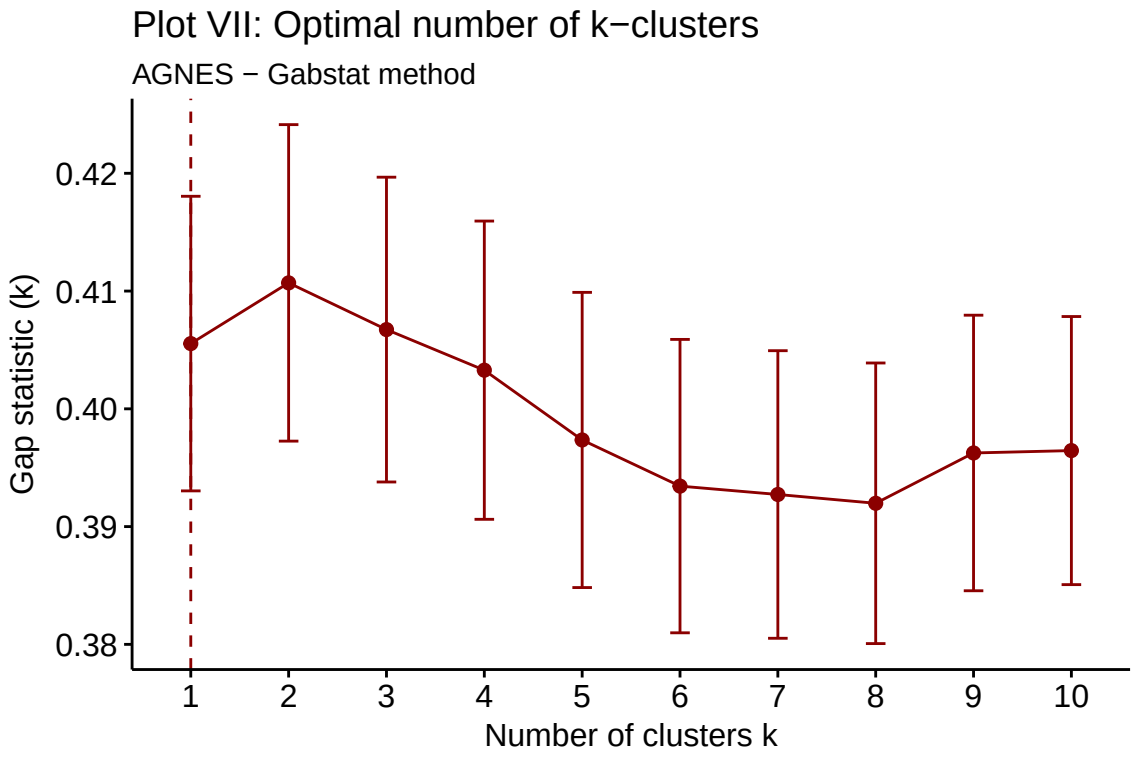
```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_clust), FUNcluster = agnes.reformat, K.max = 10,     B = 500)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
```

```
##  [1,] 5.539804 5.945337 0.4055337 0.01240160
##  [2,] 5.472407 5.883168 0.4107619 0.01248618
##  [3,] 5.429605 5.835972 0.4063668 0.01198693
##  [4,] 5.395149 5.797625 0.4024762 0.01202767
##  [5,] 5.367678 5.764305 0.3966266 0.01174772
##  [6,] 5.340844 5.733742 0.3928977 0.01164013
##  [7,] 5.313070 5.705214 0.3921446 0.01156530
##  [8,] 5.286846 5.677965 0.3911190 0.01144051
##  [9,] 5.256546 5.651651 0.3951054 0.01136246
## [10,] 5.231040 5.626128 0.3950874 0.01133356
```

According to the Tibshirani metric the optimal amount of clusters is 1. This can also be visualized.

Agnes gap statistic

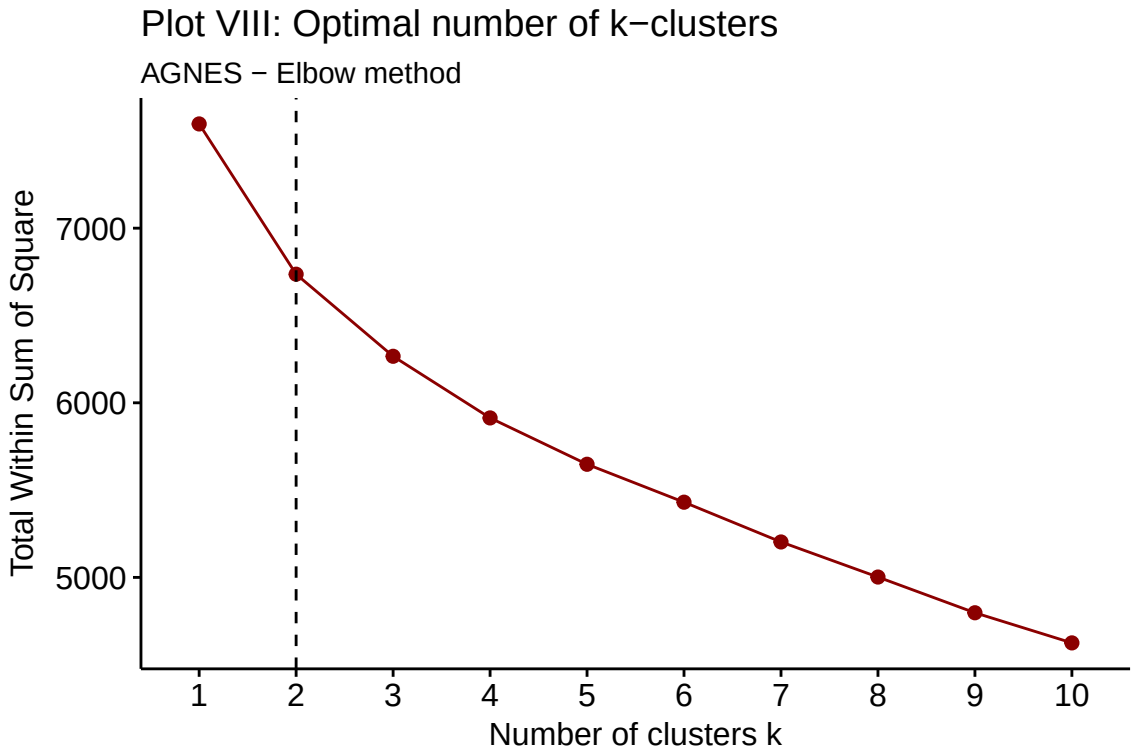
```
# Calculation
agnes_clust_gap <- fviz_nbclust(scale(df_clust), agnes.reformat,
                 method="gap_stat", linecolor="darkred",
                 k.max=10)

agnes_clust_gap +
  labs(title="Plot VII: Optimal number of k-clusters",
  subtitle="AGNES - Gabstat method")
```



The above plot shows, that the optimal number of clusters according to the gap statistic is 1.

Elbow plot

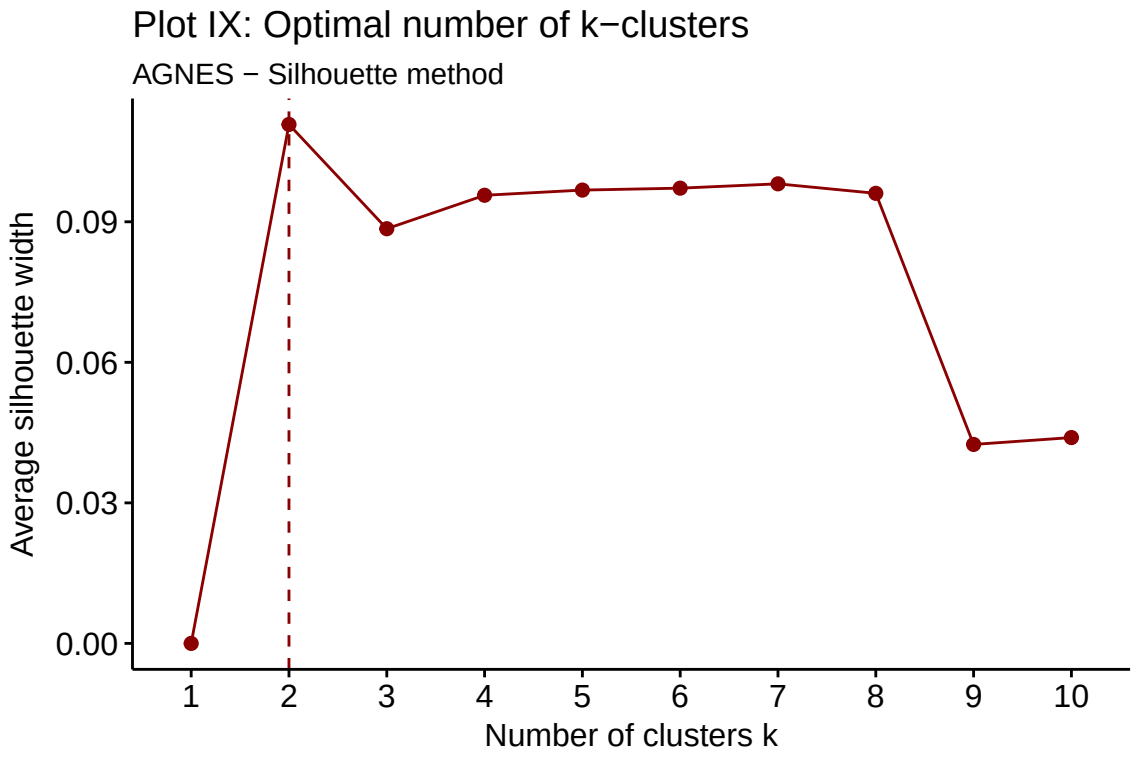
```
# Visualization
fviz_nbclust(scale(df_clust), agnes.reformat, method="wss", linecolor="darkred") +
  labs(title="Plot VIII: Optimal number of k-clusters",
  subtitle="AGNES - Elbow method") +
  geom_vline(xintercept=2,linetype=2)
```



The above plot shows, that the optimal number of clusters according to the elbow plot is 2.

Agnes silouhette

```
# Calculation
agnes_clust_sil <- fviz_nbclust(scale(df_clust), agnes.reformat,
                 method="silhouette", linecolor="darkred")

# Visualization
agnes_clust_sil +
  labs(title="Plot IX: Optimal number of k-clusters",
  subtitle="AGNES - Silhouette method")
```



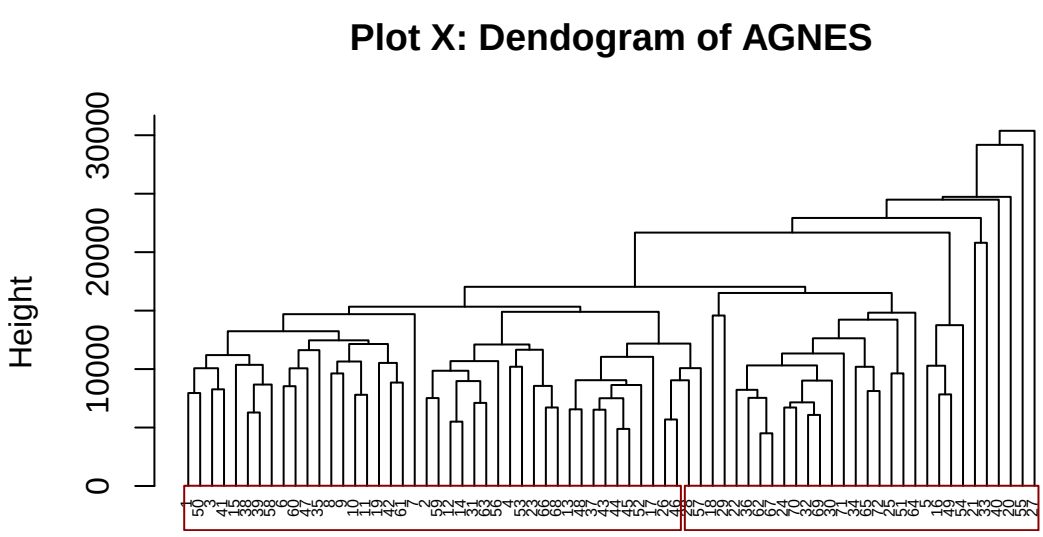
According to the silhouette plot above the optimal number of clusters is again at 2.

Summary

The different methods for the pam algorithm don't agree on the optimal amount of clusters. In order to have a sensible amount of clusters the highest number with 2 clusters is chosen.

Ward method

```
sel_agnes <- agnes(scale(df_clust),
                   method="ward",
                   stand=T)

pltree(agnes(df_clust), cex=0.5, hang=-0.01, main="Plot X: Dendogram of AGNES")
rect.hclust(sel_agnes, k=2, border="darkred")
```




```
table(agnes(df_clust)$order[43:length(agnes(df_clust)$order)] %in%
  as.numeric(names(sil_pam[52:length(sil_pam)])))
```

```
##
## FALSE  TRUE
##    13    17
```

The dendogram for the AGNES clustering shows the two selected clusters. There appears to be some overlap between the clusters of AGNES and PAM. However, a sizeable number is classified into a different cluster than seen before by the PAM algorithm.

Calculations

```
out_agnes <- as.integer((agnes.reformat(scale(df_clust), 2))[[1]])
sil_agnes <- silhouette(out_agnes, dist(scale(df_clust)))[, 3]
index_agnes_neg_sil <- which(sil_agnes < 0)
print("Agnes Misclassified Genes:")
```

```
## [1] "Agnes Misclassified Genes:"
```

```
print(row.names(df_clust)[index_agnes_neg_sil])
```

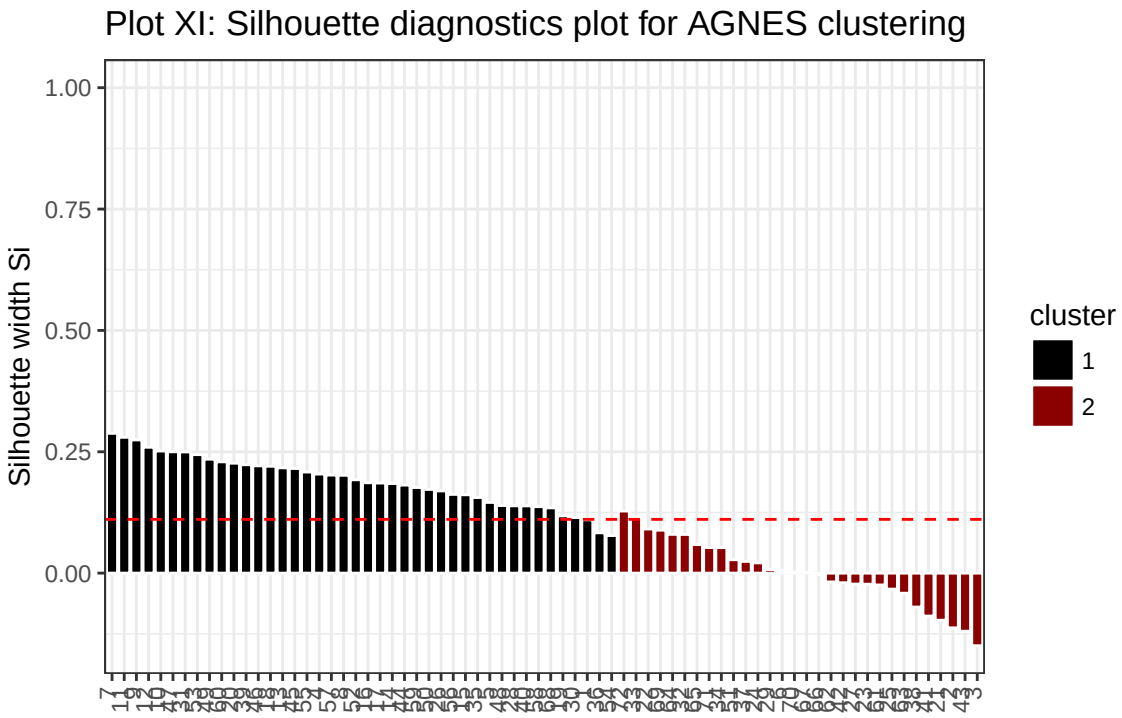
```
##  [1] "2"  "3"  "21" "23" "25" "27" "38" "41" "42" "43" "61" "62" "63" "66"
```

Visualization

```
fviz_silhouette(silhouette(out_agnes, dist(scale(df_clust))),
                main="Plot XI: Silhouette diagnostics plot for AGNES clustering") +
```

```
  theme_bw() +
  scale_fill_manual(values=c("black", "darkred")) +
  scale_color_manual(values=c("white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

```
##   cluster size ave.sil.width
## 1       1   42          0.19
## 2       2   30          0.00
```



The silhouette diagnostics plot above shows that the following genes are probably misclassified: 2, 3, 21, 23, 25, 27, 38, 41, 42, 43, 61, 62, 63, 66 into cluster 2. Cluster one appears to be rather solid with most of the points being classified above the dotted line. However, as seen before, the second cluster appears to be less stable with most of the points being below the dotted line.

As already commented upon, both clustering algorithm appear to choose a more stable cluster one and a less stable cluster two. There appears to be some overlap between the two clustering methods, nevertheless, a sizeable amount of genes are classified into different clusters between the two algorithms. Cluster one for the PAM algorithm is bigger than for the Agnes algorithm. Furthermore, Agnes appears to have a higher rate of possible misclassified genes than PAM.

Question D.) (10 points)

Apply **Fuzzy clustering** (FANNY) to the data, determining the optimal number of clusters as in (b). Summarize the results using both a principal components plot, and a correlation plot of the cluster membership weights. Based on the cluster membership weights, do you think it makes sense to consider summarizing the results using a principal components plot? Briefly justify.

Fuzzy Clustering

Gap statistics fanny

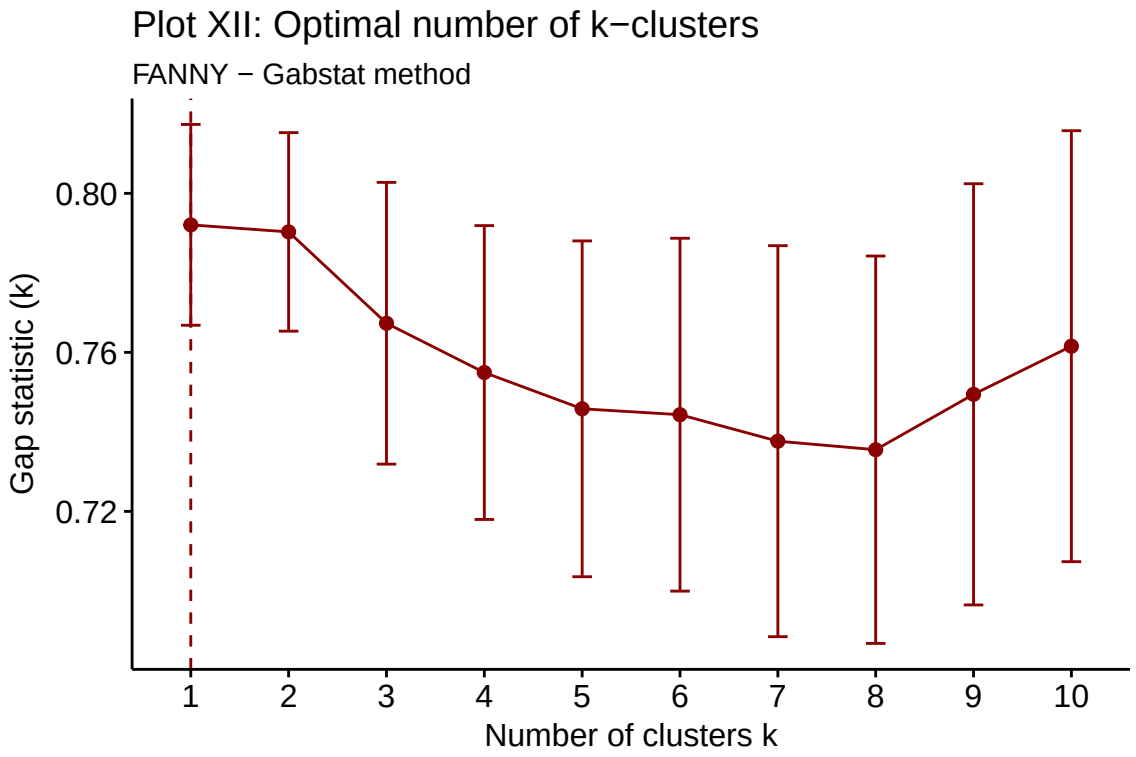
```
gapstat_fanny <- clusGap(scale(df_clust),
                         FUN=fanny,
                         K.max=10,
                         B=500,
                         memb.exp=1.5,
                         maxit=1000,
                         d.power=2)
print(gapstat_fanny, method = "Tibs2001SEmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_clust), FUNcluster = fanny, K.max = 10,     B = 500, d.power = 2, memb.exp = 1.5
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 8.242362 9.034435 0.7920736 0.02525855
##  [2,] 8.119112 8.909416 0.7903043 0.02496744
##  [3,] 8.119112 8.886434 0.7673222 0.03543940
##  [4,] 8.119112 8.874035 0.7549225 0.03696263
##  [5,] 8.119112 8.864909 0.7457968 0.04224178
##  [6,] 8.119112 8.863436 0.7443239 0.04438671
##  [7,] 8.119112 8.856765 0.7376526 0.04918112
##  [8,] 8.119112 8.854619 0.7355071 0.04872669
##  [9,] 8.103470 8.852911 0.7494418 0.05297755
## [10,] 8.086654 8.848205 0.7615510 0.05420477
```

According to the Tibshirani metric the optimal amount of clusters is 1. This can also be visualized.

Fuzzy gap statistic

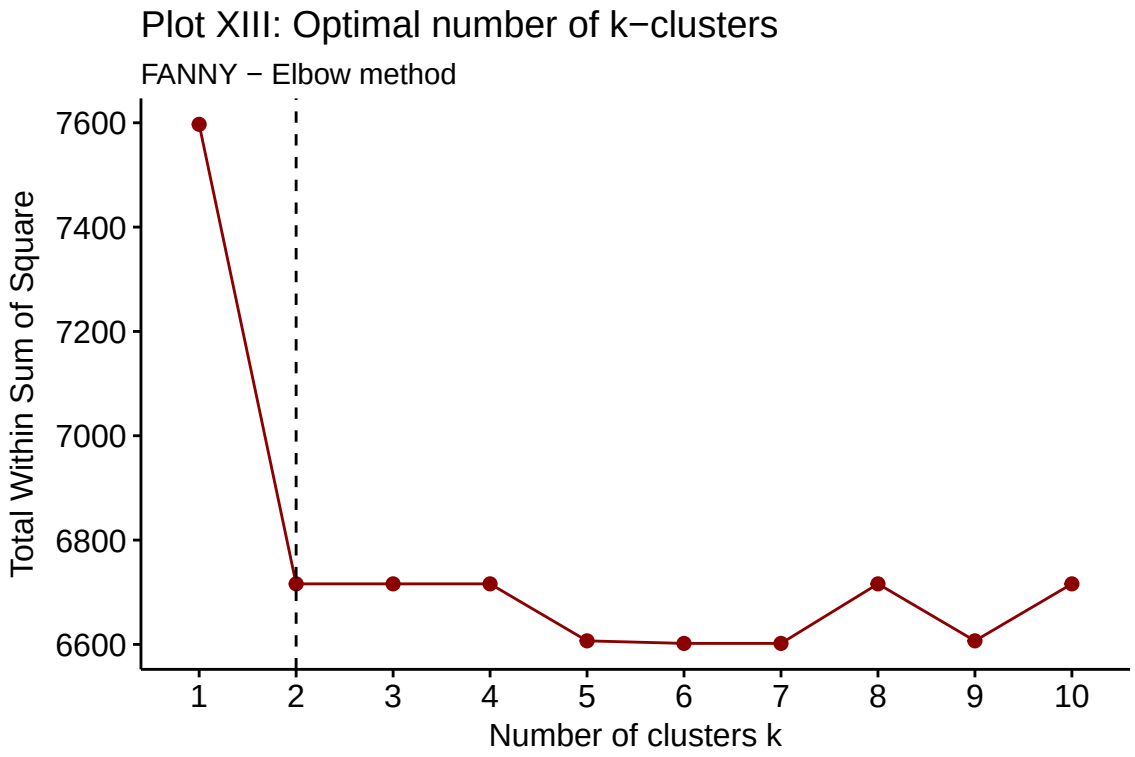
```
# Calculation
fanny_clust_gap <- fviz_gap_stat(gapstat_fanny,
                                 linecolor="darkred")

fanny_clust_gap +
  labs(title="Plot XII: Optimal number of k-clusters",
  subtitle="FANNY - Gabstat method")
```



The above plot shows, that the optimal number of clusters according to the gap statistic is 1.

Elbow plot

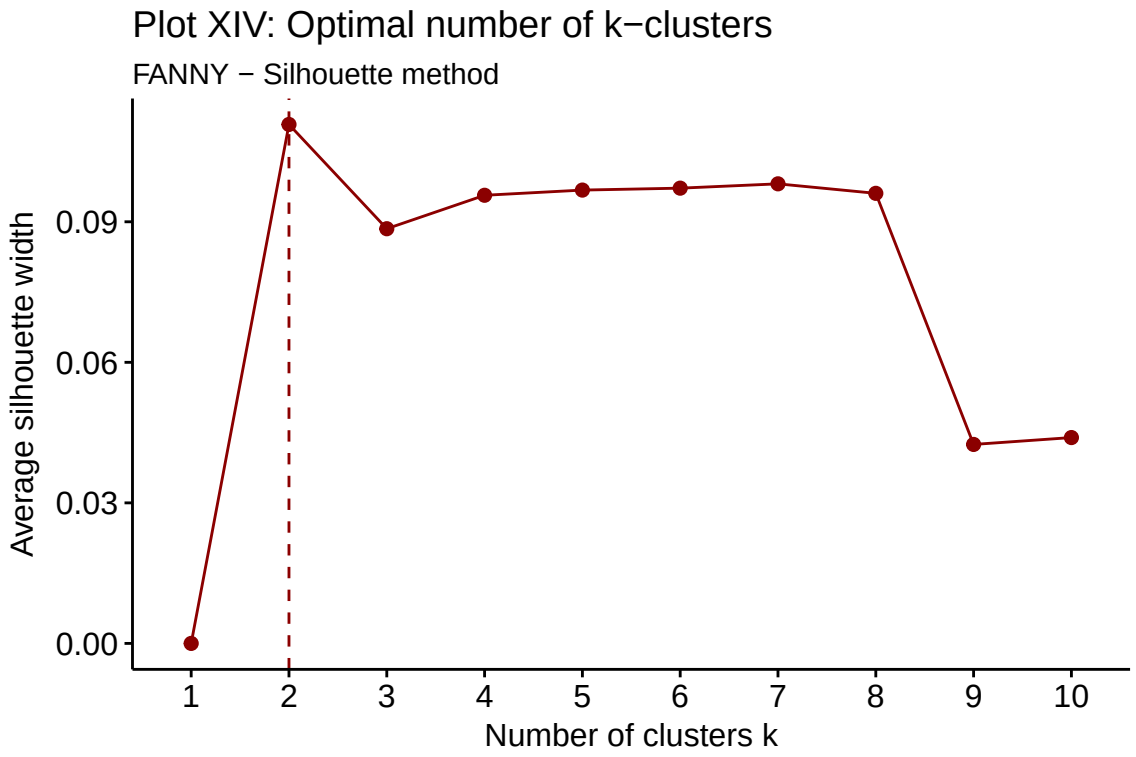
```
# Visualization
fviz_nbclust(scale(df_clust), fanny, method="wss", linecolor="darkred") +
  labs(title="Plot XIII: Optimal number of k-clusters",
  subtitle="FANNY - Elbow method") +
  geom_vline(xintercept=2,linetype=2)
```



The above plot shows, that the optimal number of clusters according to the elbow plot is 2.

Agnes silouhette

```
# Calculation
fanny_clust_sil <- fviz_nbclust(scale(df_clust), fanny,
                 method="silhouette", linecolor="darkred")

# Visualization
agnes_clust_sil +
  labs(title="Plot XIV: Optimal number of k-clusters",
  subtitle="FANNY - Silhouette method")
```



According to the silhouette plot above the optimal number of clusters is again at 2.

Summary

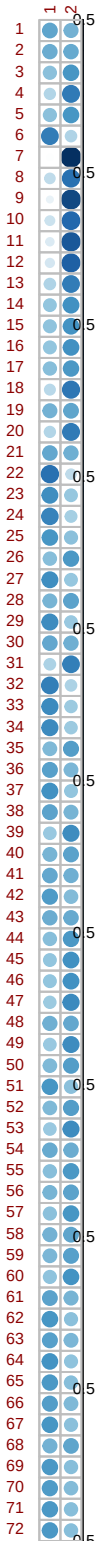
The different methods for the pam algorithm don't agree on the optimal amount of clusters. In order to have a sensible amount of clusters the highest number with 2 clusters is chosen.

Correlation plot for fanny

```
sel_fanny <- fanny(scale(df_clust), 2)
pandoc.header("Plot XV: Correlation plot for FANNY", level=4)
```

```
##
## #### Plot XV: Correlation plot for FANNY
```

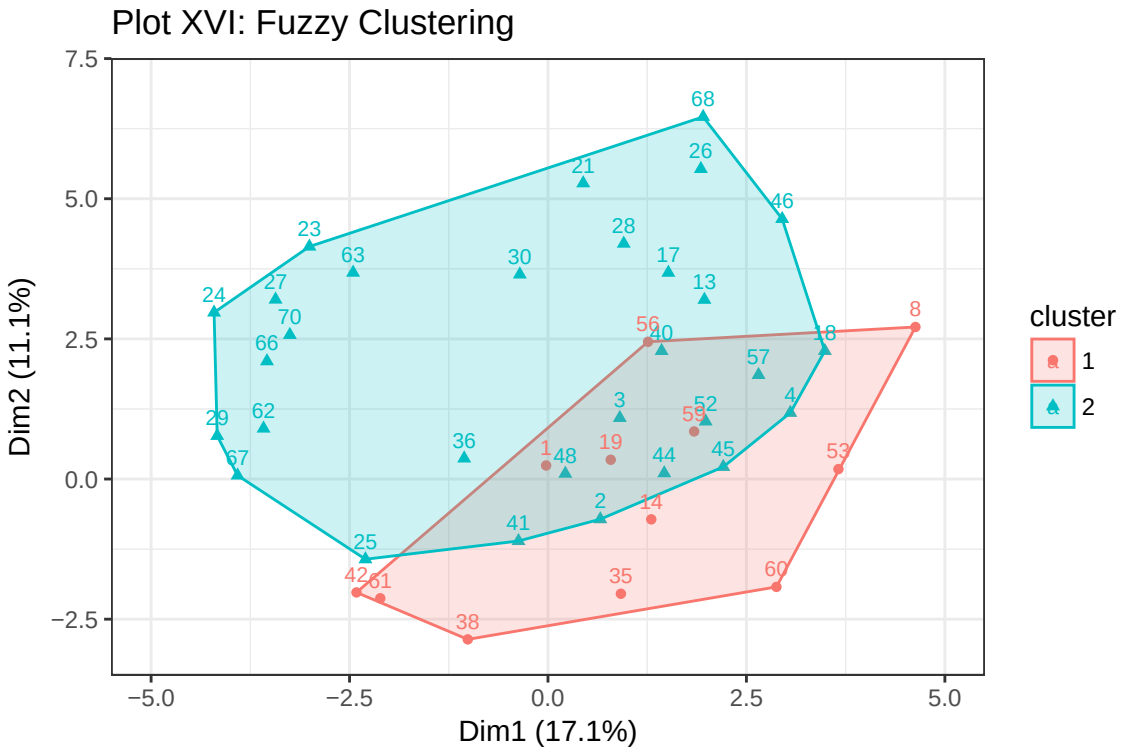
```
corrplot(sel_fanny$membership,
         is.corr=FALSE,
         tl.cex=0.5, tl.col="darkred", cl.cex=0.5)
```



For the case of fuzzy clustering, it appears that the correlation plot indicates difficulties distinguishing between the two clusters. Creating a principal components plot would visualize this situation. It might be helpful in further indicating which cases are borderline and which aren't.

Visualization

```
sel_fanny <- fanny(df_clust, 2)
fviz_cluster(sel_fanny,
             data=scale(df_clust),
             main="Plot XVI: Fuzzy Clustering",
             labelsize=8) +
  theme_bw() +
  xlim(-5, 5) +
  ylim(-3, 7)
```



As can bee seen above, the genes, 41, 2, 44, 45, 4, 16, 40, 3, 48, 52 as well as 56, 1, 19, 59, 14 are borderline cases. This could already be obseverd in the correlation plot above. The fuzzy algorithm appears to produce results which are not clear cut cases.

Fanny silhouette plot

```
# Calculations
sil_fanny <- silhouette(sel_fanny)[, 3]
print("Fuzzy misclassified states:")
```

```
## [1] "Fuzzy misclassified states:"
```

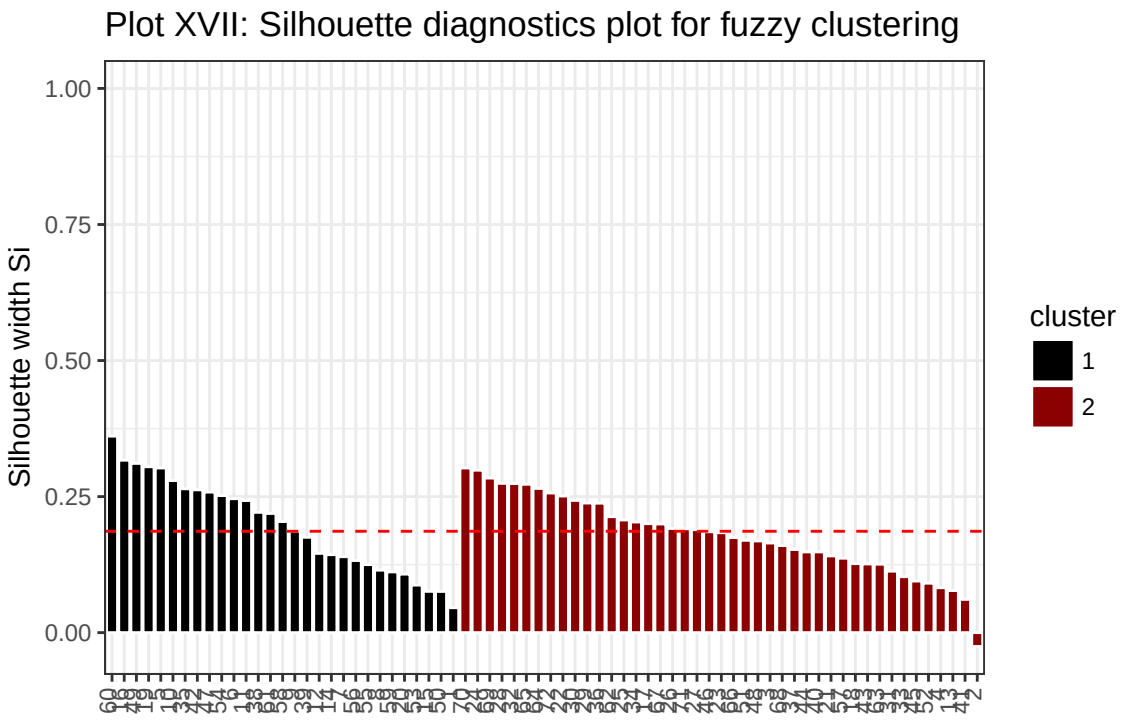
```
print(names(which(sil_fanny < 0)))
```

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

Visualization

```
fviz_silhouette(silhouette(sel_fanny),
                main="Plot XVII: Silhouette diagnostics plot for fuzzy clustering") +
  theme_bw() +
  scale_fill_manual(values=c("black", "darkred")) +
  scale_color_manual(values=c("white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

```
##   cluster size ave.sil.width
## 1       1   29          0.20
## 2       2   43          0.18
```



The above diagnostics plot shows, that there is an indication of one missclassified case. However, in both clusters a lot of genes are lying below the dotted line. This further shows, that the fuzzy algorithm appears to be having dificutlties in distinguising between the two clusters.

Question E.) (20 points)

For the clusters found in parts (b)-(d), select just one of the clusterings, preferably with the largest number of clusters. For this clustering, what proportion of each cluster are ALL (Acute Lympohblastic Leukemia) samples? In each cluster, what proportion are samples belonging to female subjects? In each cluster, what proportion of the samples were taken from bone marrow as opposed to peripheral blood? What, if anything, does this analysis imply about the clusters you discovered?

Prepare data

No clustering algorithm selects more than 2 clusters. Out of all the applied methods the PAM appears to be the most promising. See (b)-(d) for the details. This is why this algorithm is chosen for the further analyis. First we're attaching the clustering to the data.

```
df_clust2 <- df_genes
df_clust2$cluster <- sel_pam$clustering
```

Calculate all.aml frequency

```
tabl_all.aml <- table(df_clust2$cluster, df_clust2$all.aml)
pander(list("Frequency/Counts table:", tabl_all.aml))
```

- Frequency/Counts table:

ALL	AML
44	7
3	18

-

```
pander(list("Proportion table I (Colums):", round(prop.table(tabl_all.aml, 2), 3)))
```

- Proportion table I (Colums):

ALL	AML
0.936	0.28
0.064	0.72

-

```
pander(list("Proportion table II (Rows):", round(prop.table(tabl_all.aml, 1), 3)))
```

- Proportion table II (Rows):

ALL	AML
0.863	0.137
0.143	0.857

-

```
print(paste0("Accuracy: ",
             round(sum(diag(tabl_all.aml))/sum(tabl_all.aml) * 100, 1),
             "%"))
```

```
## [1] "Accuracy: 86.1%"
```

The dataset includes 47 ALL and 25 AML samples. Looking at the different clusters shows, that cluster one comprises 44 out of all 47 (93.6%) ALL samples and cluster two 18 out of 25 (72%) AML samples. Cluster one has 51 samples, out of that 86.3% are ALL and 13.7% are AML samples. Cluster two has 21 samples,

out of that 14.3% are ALL and 85.7% are AML (Answer to question 1).

Using the PAM clusters as predictors for the ALL and AML leads to an overall accuracy rate of 86.1%. This means the PAM algorithm more or less clusters around the ALL and AML samples and is a good predictor for them.

Calculate gender frequency

```
df_clust2$gender[is.na(df_clust2$gender)] <- "missing"
tabl_gender <- table(df_clust2$cluster, df_clust2$gender)
pander(list("Frequency/Counts table:", tabl_gender))
```

- Frequency/Counts table:

F	M	missing
21	21	9
2	5	14

-

```
pander(list("Proportion table I (Colums):", round(prop.table(tabl_gender, 2), 3)))
```

- Proportion table I (Colums):

F	M	missing
0.913	0.808	0.391
0.087	0.192	0.609

-

```
pander(list("Proportion table II (Rows):", round(prop.table(tabl_gender, 1), 3)))
```

- Proportion table II (Rows):

F	M	missing
0.412	0.412	0.176
0.095	0.238	0.667

-

```
print(paste0("Accuracy: ",
             round(sum(diag(tabl_gender))/sum(tabl_gender) * 100, 1),
             "%"))
```

```
## [1] "Accuracy: 36.1%"
```

There are a lot of missing values present in the gender variable. This makes an analyis of the gender distribution in each cluster more difficult. As seen before, cluster 1 counts 51 samples, out of that 41.2% are female, 41.2% are male and 17.6% are missing values. cluster 2 counts 21 samples, out of that 9.5% are female, 23.8% are male and 66.7% are missing values. (Answer to question 2)

Overall cluster one has 91.3% of all females, 80.8% of all males and 39.1% of all missing values. Cluster 2 has

8.7% of all females, 19.2% of all males and 60.9% of all missing values.

The overall accuracy rate when using the clusters for predictors for the three categories (female, male, missing) is at 36.1% which is only slightly above chance. The PAM cluster algorithm doesn't appear to have clustered around the gender of the participants.

Calculate bm.pb frequency

```
tabl_bm.pb <- table(df_clust2$cluster, df_clust2$bm.pb)
pander(list("Frequency/Counts table:", tabl_bm.pb))
```

- Frequency/Counts table:

BM	PB
44	7
18	3

-

```
pander(list("Proportion table I (Colums):", round(prop.table(tabl_bm.pb, 2), 3)))
```

- Proportion table I (Colums):

BM	PB
0.71	0.7
0.29	0.3

-

```
pander(list("Proportion table II (Rows):", round(prop.table(tabl_bm.pb, 1), 3)))
```

- Proportion table II (Rows):

BM	PB
0.863	0.137
0.857	0.143

-

```
print(paste0("Accuracy: ",
             round(sum(diag(tabl_bm.pb))/sum(tabl_bm.pb) * 100, 1),
             "%"))
```

```
## [1] "Accuracy: 65.3%"
```

Looking at the bm.pb variable it appears that roughly 2/3 of all BM and BP data is in cluster one and 1/3 of all BM and BP is in cluster 2. Using the clusters as a predictor would lead to an overall accuracy rate of 65.3%. Even when beeing well above chance (50%) it doesn't appear that the PAM clusters are good predictors for the two different sampling methods.

In cluster 1 the proportion of BM 86.3% vs. 13.7% PB and in cluster 2 it is 85.7% BM vs. 14.3% BP. (Answer to question 3)

Summary: The PAM algorithm appears to cluster more or less around the ALL and AML samples and is a good predictor for them. This isn't the case for the gender and/or bm.pb variable. (Answer to question 4)

Problem 2: Classification [40 points]

For the following problem, we will not be using the general information about the sample due to missing values. Subset the columns keeping only the ALL.AML and the 107 genetic expression values. Then split the samples into two datasets, one for training and one for testing, according to the indicator in the first column. There should be 38 samples for training and 34 for testing.

Data selection

```
df_train <- df_genes[df_genes$train.test == "Train", c(2, 5:length(df_genes))]
df_test <- df_genes[df_genes$train.test == "Test", c(2, 5:length(df_genes))]

df_train$all.aml <- as.factor(df_train$all.aml)
df_test$all.aml <- as.factor(df_test$all.aml)
```

The following questions essentially create a diagnostic tool for predicting whether a new patient likely has Acute Lymphoblastic Leukemia or Acute Myeloid Leukemia based only on their genetic expression values.

Question A.) (15 points)

Fit two SVM models with linear and RBF kernels to the training set, and report the classification accuracy of the fitted models on the test set. Explain in words how linear and RBF kernels differ as part of the SVM. In tuning your SVMs, consider some values of `cost` in the range of 1e-5 to 1 for the linear kernel and for the RBF kernel, `cost` in the range of 0.5 to 20 and `gamma` between 1e-6 and 1. Explain what you are seeing.

Train a svm with 5 fold cv

```
set.seed(109)
# Set parameters
costs_lin <- seq(from=1e-5, to=1, by=0.05)
costs_rbf <- seq(from=0.5, to=20, by=0.75)
gammas <- seq(from=1e-6, to=1, by=0.05)

# Optimal paramers (found trough iterations)
costs_lin <- 0.05001
costs_rbf <- 0.5
gammas <- 0.000001

# Fit model
fit_svm_cv_lin <- tune(svm, all.aml ~ .,
                       data=df_train,
                       tunecontrol=tune.control(cross=5),
                       ranges=list(cost=costs_lin,
                                   kernel='linear'))

fit_svm_cv_rad <- tune(svm, all.aml ~ .,
```

```
                        data=df_train,
                        tunecontrol=tune.control(cross=5),
                        ranges=list(cost=costs_rbf, gamma=gammas,
                                    kernel='radial'))
```

Predict accuracy

```
# Best SVM
pred_svm_lin <- predict(fit_svm_cv_lin$best.model, df_test)
pred_svm_rad <- predict(fit_svm_cv_rad$best.model, df_test)

# Confusion matrix
cm_svm_lin <- confusionMatrix(table(pred_svm_lin, df_test$all.aml))
cm_svm_rad <- confusionMatrix(table(pred_svm_rad, df_test$all.aml))

# Output
pander(cm_svm_lin$table)
```

	ALL	AML
ALL	20	4
AML	0	10

```
pander(cm_svm_lin$overall)
```

Table 11: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.8824	0.7463	0.7255	0.967	0.5882

AccuracyPValue	McnemarPValue
0.0001971	0.1336

```
pander(cm_svm_rad$table)
```

	ALL	AML
ALL	20	14
AML	0	0

```
pander(cm_svm_rad$overall)
```

Table 14: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.5882	0	0.407	0.7535	0.5882

AccuracyPValue	McnemarPValue
0.5729	0.000512

It appears that the linear SVM kernel is much better in detecting AML cases. The confusion matrices show, that the radial kernel classifies all observations into the ALL class. This results in an overall test accuracy of 58.8%. The Linear kernel on the other hand reaches an overall test accuracy of 88.2%. The linear kernel appears to be better able in detecting the AML class. It detects 10 AML cases and only missclassifies 4 cases as ALL.

The difference between a linear kernel and a RBF kernel is that in essence the linear kernel expects a linear model. The RBF uses a normal curve around its data points, and sums them in such a way that the decision boundary can be defined by a type of topology condition. According to http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.880&rep=rep1&type=pdf it can be shown that the linear kernel is a simple case of RBF. This indicates that the above RBF kernel is not properly tuned.

Question B.) (10 points)

Apply principal component analysis (PCA) to the genetic expression values in the training set, and retain the minimal number of PCs that capture at least 90% of the variance in the data. How does the number of PCs identified compare with the total number of gene expression values? Apply to the test data the rotation that resulted in the PCs in the training data, and keep the same set of PCs.

Appling principal component analysis (PCA)

```
# Fit the model
fit_pca <- prcomp(df_train[, 2:length(df_train)], scale=TRUE, center=TRUE)

# Get the amount on variance
pca_variance <- fit_pca$sdev^2/sum(fit_pca$sdev^2)
```

Calculate the pc's retaining 90% of the variation

```
top_pca <- min(which(cumsum(pca_variance) >= 0.90))
print(max(top_pca))
```

```
## [1] 23
```

Instead of using 108 Columns only 23 are sufficient in order to retain 90% of the variability in the data. This helps migigating the *curse of dimensionality* problem in the data. The curse of dimensionality refers to the phenomena that arise when analyzing data in a very high-dimensional space. The problem is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. More specifically, the amount of data needed to support the result grows exponentially with the dimensionality. Furthermore, it helps in decreasing the calculation time when performing various optimization methods.

Transforming the test and training data

Next we're applying to all the data (train & test) the rotation that resulted in the PCs in the training data, and keep the same set of PCs.

```
pca_90_vectors <- fit_pca$rotation[, 1:top_pca]

# Rescale
df_pca_train <- scale(df_train[, 2:length(df_train)],
                      center=fit_pca$center,
                      scale=fit_pca$scale)

df_pca_test <- scale(df_test[, 2:length(df_test)],
                      center=fit_pca$center,
                      scale=fit_pca$scale)

df_pca_train <- df_pca_train %*% fit_pca$rotation[, 1:top_pca]
df_pca_test <- df_pca_test %*% fit_pca$rotation[, 1:top_pca]

# Put together in df
df_pca_train <- data.frame(all.aml=df_train$all.aml, df_pca_train)
df_pca_test <- data.frame(all.aml=df_test$all.aml, df_pca_test)
```

Question C.) (15 points)

Fit a SVM model with linear and RBF kernels to the reduced training set, and report the classification accuracy of the fitted models on the reduced test set. Do not forget to tune the regularization and kernel parameters by cross-validation. How does the test accuracies compare with the previous models from part (a)? What does this convey? *Hint*: You may use similar ranges for tuning as in part (a), but for the RBF kernel you may need to try even larger values of `cost`, i.e. in the range of 0.5 to 40.

Model tuning: 5-Fold cross-validation

```
set.seed(109)
# Set parameters
costs_lin <- seq(from=1e-5, to=1, by=0.05)
costs_rbf <- seq(from=0.5, to=20, by=0.75)
gammas <- seq(from=1e-6, to=1, by=0.05)

# Optimal paramers (found trough iterations)
costs_lin <- 0.10001
costs_rbf <- 0.5
gammas <- 0.000001

# Fit model
fit_svm_cv_lin_red <- tune(svm, all.aml ~ .,
                           data=df_pca_train,
                           tunecontrol=tune.control(cross=5),
                           ranges=list(cost=costs_lin,
                                       kernel='linear'))

fit_svm_cv_rad_red <- tune(svm, all.aml ~ .,
                           data=df_pca_train,
                           tunecontrol=tune.control(cross=5),
                           ranges=list(cost=costs_rbf, gamma=gammas,
                                       kernel='radial'))
```

Predict accuracy

```
# Best SVM
pred_svm_lin_red <- predict(fit_svm_cv_lin_red$best.model, df_pca_test)
pred_svm_rad_red <- predict(fit_svm_cv_rad_red$best.model, df_pca_test)

# Confusion matrix
cm_svm_lin_red <- confusionMatrix(table(pred_svm_lin_red, df_pca_test$all.aml))
cm_svm_rad_red <- confusionMatrix(table(pred_svm_rad_red, df_pca_test$all.aml))

# Output
pander(cm_svm_lin_red$table)
```

	ALL	AML
ALL	20	5
AML	0	9

```
pander(cm_svm_lin_red$overall)
```

Table 17: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.8529	0.6792	0.6894	0.9505	0.5882

AccuracyPValue	McnemarPValue
0.0008805	0.07364

```
pander(cm_svm_rad_red$table)
```

	ALL	AML
ALL	20	14
AML	0	0

```
pander(cm_svm_rad_red$overall)
```

Table 20: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.5882	0	0.407	0.7535	0.5882

AccuracyPValue	McnemarPValue
0.5729	0.000512

The above statistics show, that the linear kernel appears to perform slightly worse than before, one more

AML case was missclassified as ALL. This means that the test accuracy decreased to 85.2%. The radial kernel on the other hand performs now better than before. It now reaches an overall accuracy of 79.4% and is able to correctly identify 7 AML cases. Nevertheless, it still performs worse than the linear kernel applied on the untransformed data. The loss on accuracy in the linear kernel indicates, that there is a sligh loss of information in the data. Furthermore, the performance of the RBF kernel indicates, first, the RBF kernel might have had a problem with hyperdimensionality and second, more parameter tuning should be performed as it's performance isn't yet on par with the linear kernel.

Tune RBF further1

```
set.seed(109)
# Set parameters
costs_rbf2 <- seq(from=1, to=40, by=0.5)
gammas2 <- seq(from=1e-6, to=1, by=0.00001)

# Optimal paramers (found trough iterations)
costs_rbf2 <- 33
gammas2 <- 0.001171

# Fit model
fit_svm_cv_rad_red2 <- tune(svm, all.aml ~ .,
                            data=df_pca_train,
                            tunecontrol=tune.control(cross=5),
                            ranges=list(cost=costs_rbf2, gamma=gammas2,
                                        kernel='radial'))
```

Predict accuracy

```
# Best SVM
pred_svm_rad_red2 <- predict(fit_svm_cv_rad_red2$best.model, df_pca_test)

# Confusion matrix
cm_svm_rad_red2 <- confusionMatrix(table(pred_svm_rad_red2, df_pca_test$all.aml))

# Output
pander(cm_svm_rad_red2$table)
```

	ALL	AML
ALL	20	5
AML	0	9

```
pander(cm_svm_rad_red2$overall)
```

Table 23: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.8529	0.6792	0.6894	0.9505	0.5882

AccuracyPValue	McnemarPValue
0.0008805	0.07364

The overall test accuracy for the RBF kernel lies now at 85.2%, which is on par with the performance of the linear kernel performed on the reduced dataset. All 20 'ALL' cases and 9 out of the 14 'AML' cases are correctly identified. The model with the variable reduction falls only 1 missclassified AML short of performing on par with the full model. Nevertheless, because of the importance of correct prediction (Leukemia), if feasible in practice, I would opt for the more accurate model with the full data.