CS 109B: Midterm Exam 2

April 6, 2017

Danqing Wang

Honor Code

The midterm must be completed entirely on your own, and may not be discussed with anybody else.

- You have to write your solutions entirely on your own.
- You cannot share written materials or code with anyone else.
- You may not provide or make available solutions to individuals who take or may take this course in the future.

Your submitted code will be automatically checked for plagiarism. If you are using external resources, make sure to indicate the sources.

The Harvard College Honor Code

Members of the Harvard College community commit themselves to producing academic work of integrity – that is, work that adheres to the scholarly and intellectual standards of accurate attribution of sources, appropriate collection and use of data, and transparent acknowledgement of the contribution of others to their ideas, discoveries, interpretations, and conclusions. Cheating on exams or problem sets, plagiarizing or misrepresenting the ideas or language of someone else as one's own, falsifying data, or any other instance of academic dishonesty violates the standards of our community, as well as the standards of the wider world of learning and affairs.

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).

(a) (10 points) Standardize the gene expression values, and compute the Euclidean distance between each pair of tissue samples. Apply multi-dimensional scaling to the pair-wise distances, and generate a scatter plot of tissue samples in two dimension. By visual inspection, into how many groups do the tissue samples 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)

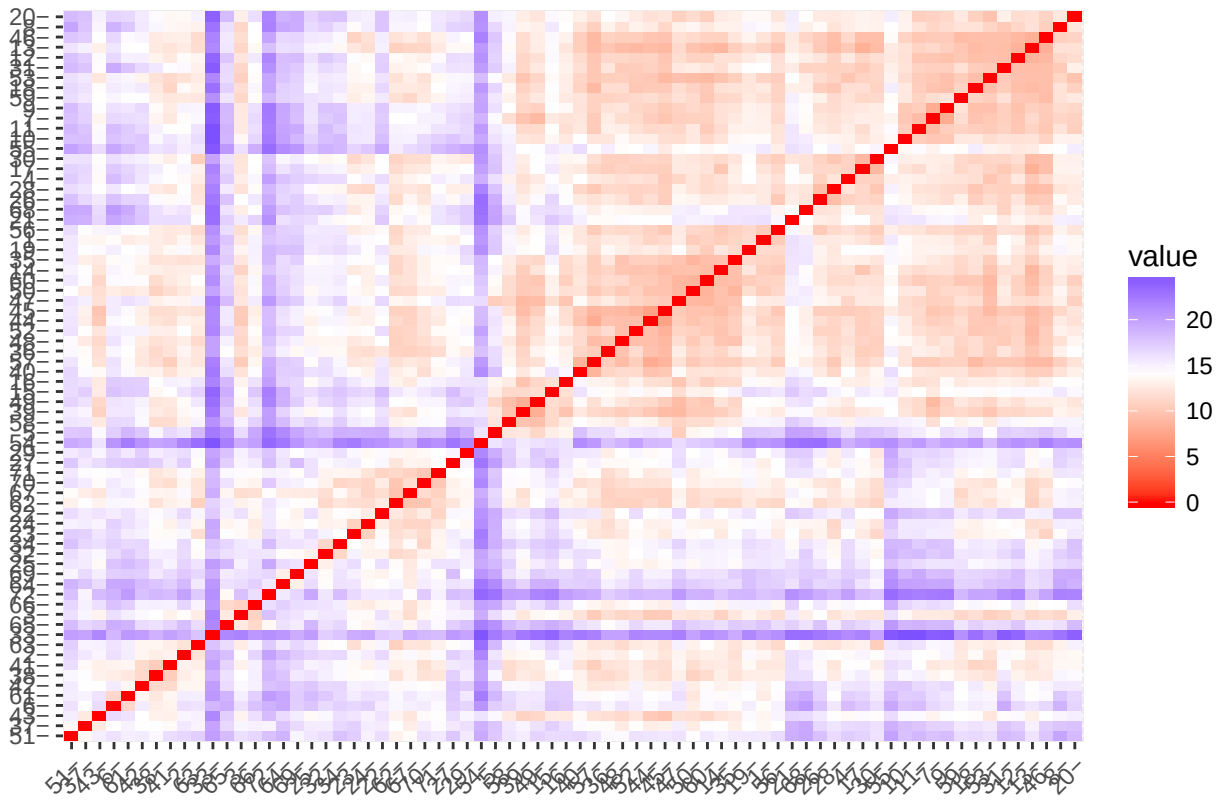
Answer

```
## Load data file, and extract relevant columns for this question
data <- read.csv("MT2_data.csv")
df <- data[, seq(5, 111)]
```

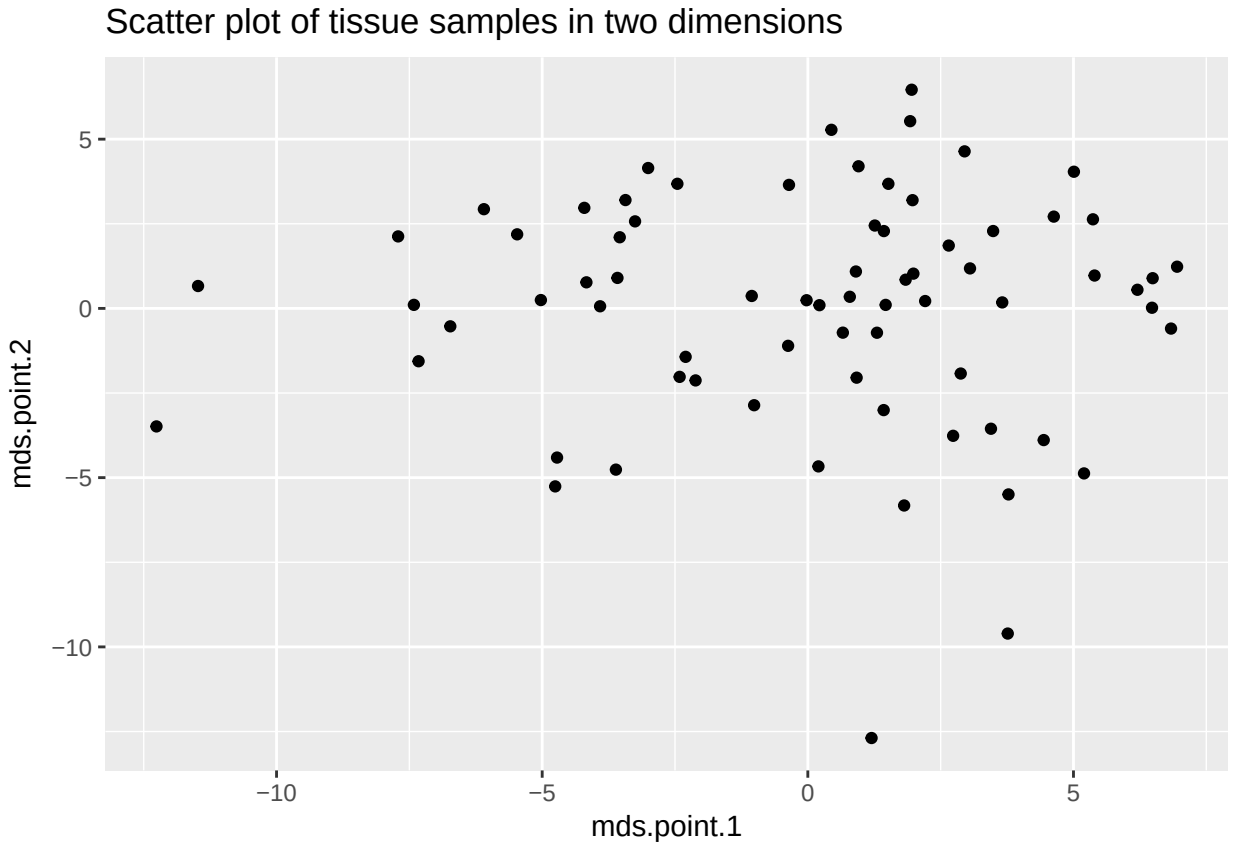
```
library(cluster)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
library(ggplot2)

## Standardize gene expression values
df.scaled <- scale(df)

## Compute the Euclidean distance between each pair of tissue samples
df.dist <- daisy(df.scaled, metric = "euclidean")

# We include a heatmap here to visualize the euclidean distances
# between pairs of tissue samples
fviz_dist(df.dist)+
  ggtitle('Heatmap showing euclidean distanes between pairs of tissue samples')
```




```
## Apply multi-dimensional scaling to the pair-wise distances,
## and generate a scatter plot of tissue samples in two dimension
df.mds <- cmdscale(df.dist)
df_mds <- data.frame(df, mds.point = df.mds)
ggplot(df_mds, mapping = aes(x = mds.point.1, y = mds.point.2)) +
  geom_point()+
  ggtitle("Scatter plot of tissue samples in two dimensions")
```



By visual inspection, into how many groups do the genes cluster?

From the graph, it looks like the data points can be grouped into two clusters: one cluster near the top left half of the plot, while the rest of the data points form another 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)

If we were to apply princiapl components analysis (PCA) to the standardized data and plot a scatter plot of data points in the first two principal components, the graph will look the same as the plot above from multidimentional scaling (MDS), up to a sign flip (i.e. the graph obtained may be a mirror flip of the above one). This is because PCA and MDS are doing essentially the same job of dimensionality reduction, and their scatter plots show the dissimilarities/distances between each data point. The possible sign flip comes from computing the loading vector, but either sign would give the same direction of most variance, so it does not matter.

PCA is a method of reducing the dimensionality of a data set by identifying a new coordinate system with fewer dimensions than before. The first principal component is chosen to capture the most amount of variance of the data, and subsequent princicpal components capture decreasing amount of variance and are orthogonal to one another. Multidimensional scaling is another dimensionality reduction method. The MDS algorithm works by taking in the n x n matrix of pairwise distances between data points, and then compute n x k matrix X with coordinate of distance, and then performing PCA on the matrix X. It is a means of visualizing a set of data points by displaying the information contained in the pairwise distane matrix of data points in the data set. Here, the MDS algorithm is placing each data point in a 2 dimensional space such that their pairwise distance information is most well preserved. It is essentially doing the same thing as PCA by capturing the most variance in the data in 2 dimensions. Hence, their graphs would be the same (up to a sign flip).

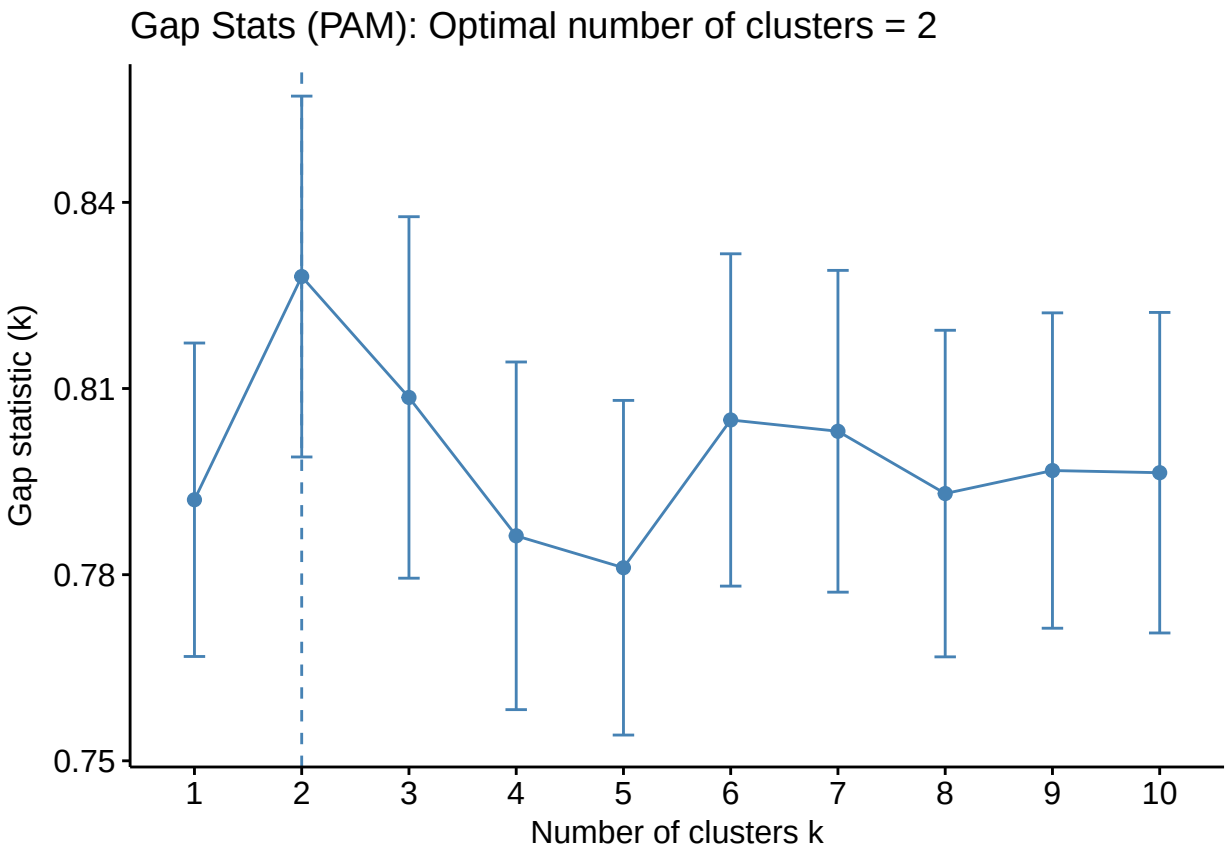
(b) (10 points) Apply **Partitioning around medoids** (PAM) to the data, selecting the optimal number of clusters based on the Gap, Elbow and Silhouette statistics – if the they disagree, select the largest number of clusters indicated by the statistics. Summarize the results of clustering using a principal components plot, and comment on the quality of clustering using a Silhouette diagnostic plot.

Answer

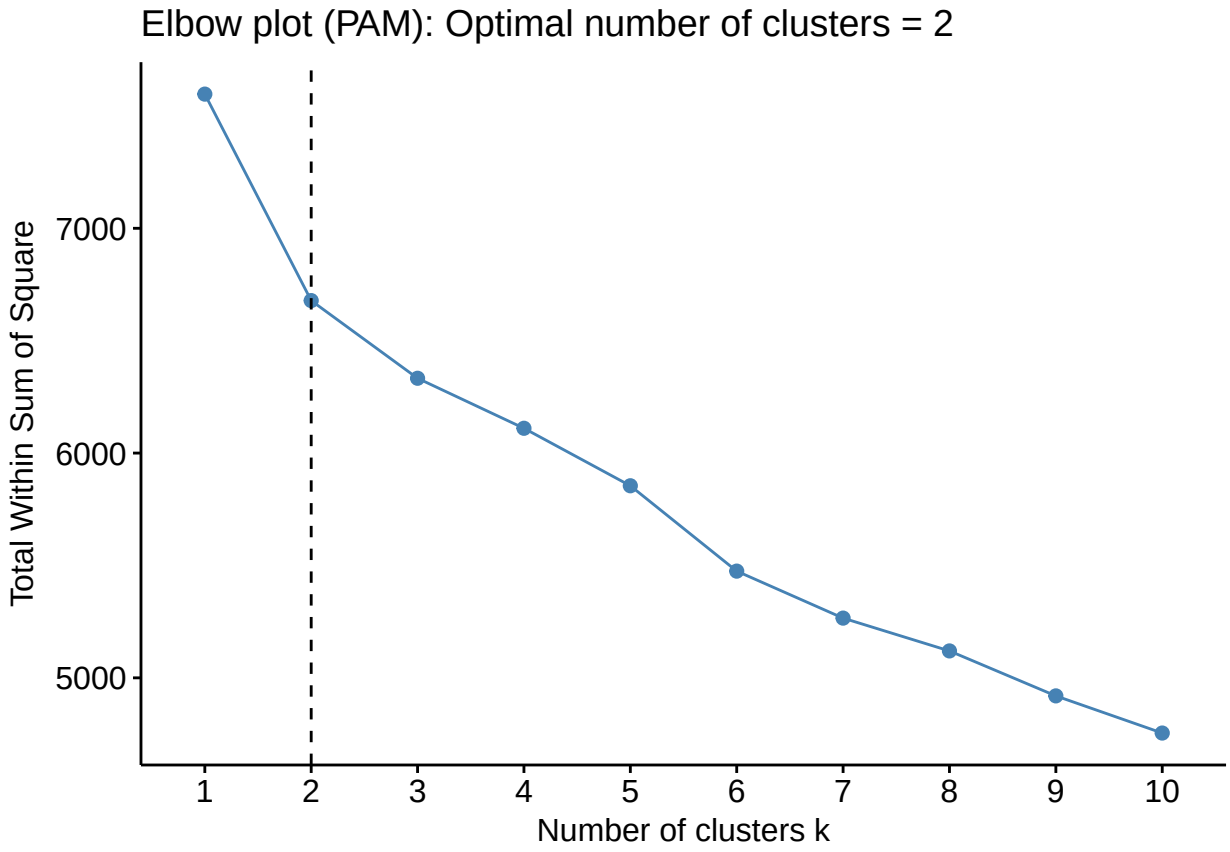
```
## Gap Statistics
set.seed(123)
gapstat <- clusGap(scale(df), FUN=pam, K.max=10, B=500, d.power=2)
print(gapstat, method="Tibs2001SEmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df), FUNcluster = pam, K.max = 10, B = 500,     d.power = 2)
## 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,] 8.242362 9.034435 0.7920736 0.02525855
##  [2,] 8.113485 8.941535 0.8280500 0.02907813
##  [3,] 8.060356 8.868919 0.8085627 0.02912886
##  [4,] 8.024586 8.810849 0.7862627 0.02801571
##  [5,] 7.981823 8.762943 0.7811195 0.02696769
##  [6,] 7.914764 8.719700 0.8049360 0.02677822
##  [7,] 7.875843 8.678955 0.8031118 0.02593064
##  [8,] 7.847721 8.640794 0.7930725 0.02631644
##  [9,] 7.807867 8.604649 0.7967818 0.02541339
## [10,] 7.773683 8.570117 0.7964349 0.02582996
```

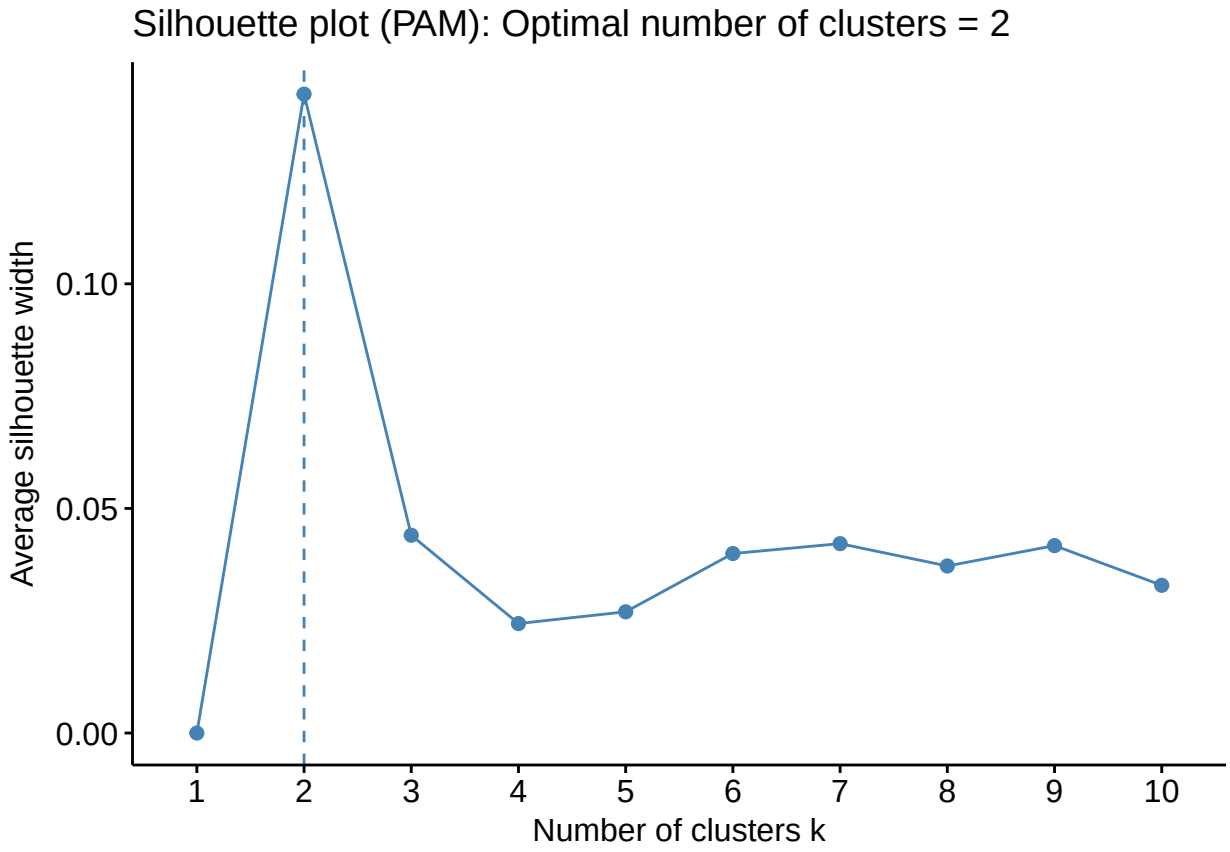
```
fviz_gap_stat(gapstat,
  maxSE=list(method="Tibs2001SEmax",SE.factor=1)) +
  ggtitle("Gap Stats (PAM): Optimal number of clusters = 2")
```




```
## Elbow method
fviz_nbclust(scale(df), pam, method="wss") +
  ggtitle("Elbow plot (PAM): Optimal number of clusters = 2")+
 geom_vline(xintercept=2,linetype=2)
```




```
## Silhouette statistics
fviz_nbclust(scale(df),pam,method="silhouette") +
  ggtitle("Silhouette plot (PAM): Optimal number of clusters = 2")
```



All three methods agree that the best number of clusters is 2. We shall use this in our PAM model below:

```
## Summarize the results of clustering using a principal components plot
set.seed(123)
df.pam = pam(scale(df), k = 2)
print(df.pam)
```

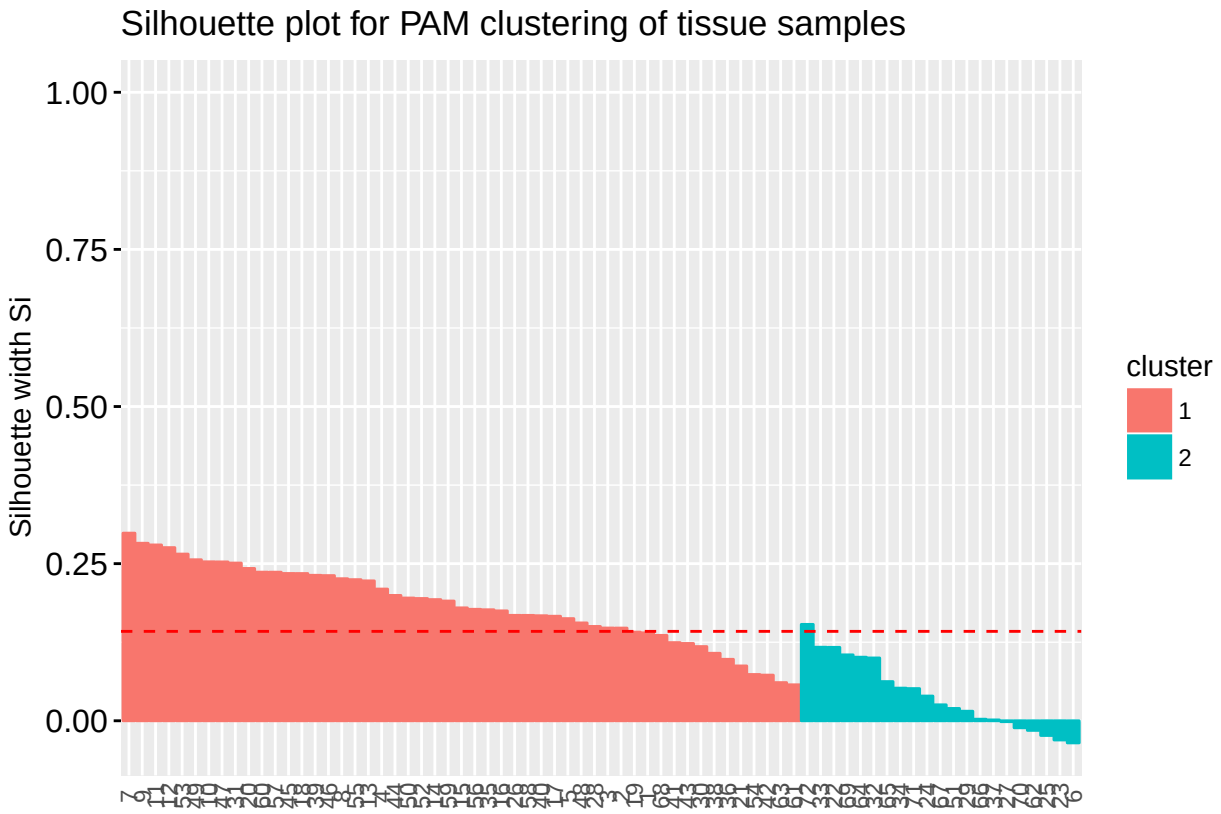
```
## Medoids:
##      ID M84371_rna1_s_at  J03779_at X82240_rna1_at U22376_cds2_s_at
## [1,] 45       -0.8658939 -0.5768048     -0.6617588       0.07864744
## [2,] 67       -0.9260392 -0.3318907     -0.6283761      -0.91338235
##       U79262_at M27396_s_at  J04948_at M19508_xpt3_s_at  M22324_at
## [1,] -0.1559333   0.4953474 -0.3697183       -0.4086798 -0.2722651
## [2,] -0.1559333   0.9132968  0.4604069        0.1533896  1.7266917
##       M23197_at U41813_at X13293_at  D50683_at  M93425_at  X70683_at
## [1,] -0.6057632 -0.588435 1.3259999  0.7052036 -1.0419159  0.7109280
## [2,] -0.2210090  1.100016 0.3358145 -0.5853228  0.3211326 -0.2055909
##       X80907_at  M63167_at   D00632_at  D00763_at  D10511_at
## [1,] -0.5161717 -0.8079706 -0.02348402 -0.3825249 -0.3228478
## [2,]  2.1891683  0.6740303 -0.17075155 -0.6735245 -0.1104762
##      AF000573_rna1_at  D14874_at D28532_at  D28915_at  D78367_at
## [1,]      -0.02381374 -0.1365293  0.293403 -0.2445392 0.05861101
## [2,]       0.57393302  2.0615074  0.194688 -0.1019464 0.21910289
##       M24439_at  D83407_at    D83735_at  D87845_at   D88213_at  D49400_at
## [1,] -0.3150173 -0.4280526 -0.476110771 -0.6245451 -0.68147467 -0.6075118
## [2,]  0.2488811  1.1881973 -0.006125687  1.1543789  0.05343722  0.6334642
##       D89052_at J05096_rna1_at AF003743_at  J00124_at X00351_f_at
## [1,] -0.7391655     -0.6280963  -0.3748789 -0.5406977   0.1485933
```

```
## [2,] -0.6158999      0.8813079  -1.3824234  0.8478197  -0.1428542
##       J02876_at   J04794_at   L00352_at  L02321_at L10102_rna1_at
## [1,] -0.6263258 -0.04365801  1.25719844 -0.4361608    -0.09429473
## [2,] -0.3936012 -0.18164069 -0.01674004  1.0728492     0.02056051
##       L29339_at  L35545_at  L36531_at L37033_at L37378_at L38608_at
## [1,]  1.1993990 -1.4641902 0.05524155 0.0501288 -0.834332 0.3202676
## [2,] -0.2100565 -0.2813806 0.33873646 0.2344947 -2.515284 0.9638852
##       L39833_at  U16861_at   L38500_at  D90084_at  L42450_at  M14091_at
## [1,]  0.6247226 -0.1065456 -0.06241768 -1.0094343 -1.4257944 -0.6274445
## [2,] -0.6062247  1.3946993 -0.06241768  0.7691311  0.2405336 -0.5759795
##       M11186_at M15353_at M19483_at  M20681_at  K03195_at M22877_at
## [1,] -0.8558435  0.584892 0.4687881 -0.4504681 -0.1010224  1.160105
## [2,] -0.4417930 -1.054207 0.1538990  2.0592484  0.6639227  1.568990
##        J04444_at  M22976_at M23114_at M28879_at M29273_at  M55047_at
## [1,] -0.01657368 -0.3437371 0.8166472 0.3064909 0.5383972  0.6016580
## [2,]  0.14891006  0.4085143 1.7559608 0.2981384 0.1139442 -0.3824432
##      M63962_rna1_at  M18185_at M64098_at  M75110_at  M76378_at  M85247_at
## [1,]      0.2073224  1.9306108 0.5511519  0.1829221 -0.7539816 -0.2270826
## [2,]      0.3060928 -0.6675944 0.2261457 -0.3841561  0.7578863  0.2148078
##       M86808_at  M90366_at   M93718_at   S71129_at  S79854_at  S76475_at
## [1,] -0.9013541 -0.5737442  0.03196313  0.05521937 -0.2004884 0.09933918
## [2,] -0.9718438  0.4143415 -1.21451833 -0.64336992 -0.9906721 0.88408756
##       S90469_at   U07882_at  U11292_at  U14588_at U18244_at U20758_rna1_at
## [1,] -0.5278510 -0.26998966 -0.2840595 -0.4112933 0.6131702     0.02693848
## [2,]  0.8690258 -0.03292557  0.3809853  0.7271560 0.5830307     0.35020025
##      U23143_at  U29680_at  U34877_at  U48408_at J02854_at M31211_s_at
## [1,] 0.2083849 -0.3232826 -0.5636787 -0.4147465 -0.404156   1.8558281
## [2,] 0.5512337  0.5991156  0.8058433  0.4660447 -1.168114  -0.3278218
##       U07139_at M19045_f_at M95623_cds1_at   X70297_at  Y00433_at
## [1,]  0.8336517  -0.6493682    -0.31535226 -0.33784730 -1.5180588
## [2,] -0.6361907  -0.2440686     0.01283624  0.05600186  0.2963622
##       U60115_at  D29963_at  M63835_at M80254_at  D88270_at   X15414_at
## [1,]  0.3764644 -0.3392307 -0.5168879  -0.39196 -0.7484410 -0.18608980
## [2,] -0.4306229  0.4130256 -0.5953364   1.11552 -0.7017633  0.01813941
##       X63527_at  X67951_at  L13278_at L14848_s_at U46006_s_at U49957_s_at
## [1,]  0.4646056 -0.1963984 -0.4777876   0.5811859  -0.4574982  0.09686789
## [2,] -0.5792965 -0.2956007 -0.6827978  -0.7465699  -0.4722933  1.13862689
##      X01677_f_at
## [1,]   0.1503522
## [2,]  -0.7311682
## Clustering vector:
##  [1] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 2 1 1 2 2 2 1
## [36] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2
## [71] 2 2
## Objective function:
##   build    swap
## 11.2242 11.2242
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

```
fviz_cluster(df.pam, # don't need to specify data for pam
  main="PAM clustering of tissue samples")
```


```
## Comment on the quality of clustering using a Silhouette diagnostic plot
fviz_silhouette(silhouette(df.pam),
  main="Silhouette plot for PAM clustering of tissue samples")+
    theme(axis.text.x = element_text(angle = 90))
```

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




```
# Compute silhouette
sil.pam = silhouette(df.pam)[, 1:3]

# Objects with negative silhouette
neg_sil_index.pam = which(sil.pam[, 'sil_width'] < 0)
print(sil.pam[neg_sil_index.pam, , drop = FALSE])
```

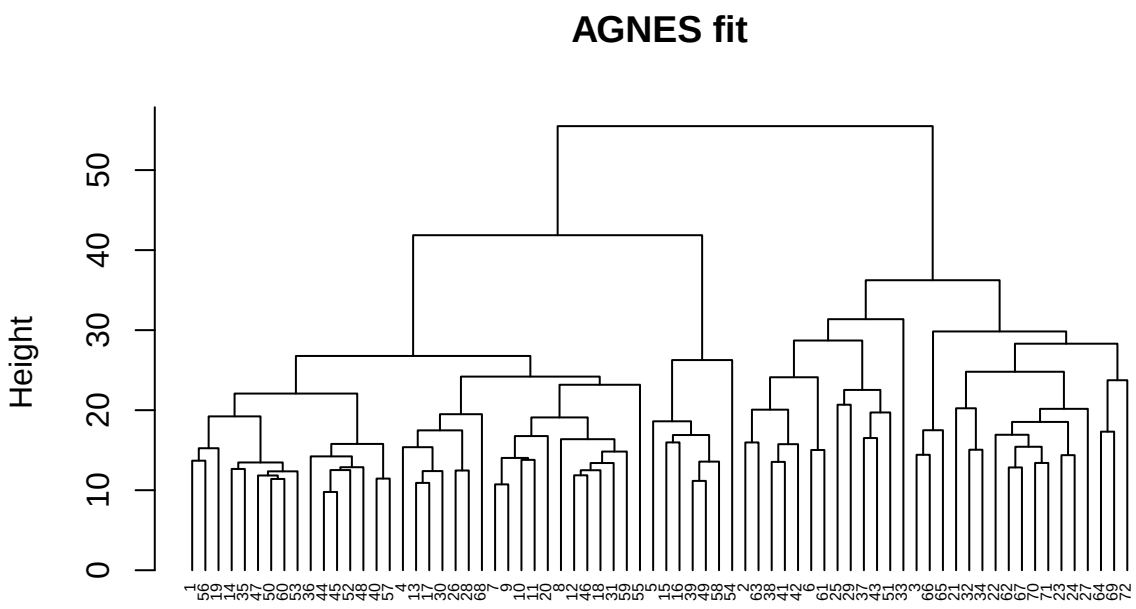
```
##    cluster neighbor    sil_width
## 27       2        1 -0.001402723
## 70       2        1 -0.011143023
## 62       2        1 -0.015290502
## 25       2        1 -0.023409205
## 23       2        1 -0.030588593
## 6        2        1 -0.035182961
```

The silhouette plot shows that the cluster 1 is fairly well clustered, with most of the observations above the average sil_width. The closer the value of a particualr sil_width is to 1, the most similar the sample is to the rest of the members in its cluster compared to members of the other cluster. However, cluster 2 data points are mostly below the average sil_width line and is not very well clustered. The smaller sil_width indicates that members of this cluster have very high dissimilarities among themselves, though not as high as the dissimilarities when compared to members of the other cluster. In cluster 2, we noticed there there are 6 tissue samples (# 6, 23, 25, 27, 62, 70) with negaive sil_widths. A negative sil_width means the dissimilarity between a particular tissue sample and the tissue samples in the other cluster is smaller than the dissimilarity between the tissue sample and the rest of the tissue samples in its own cluster. This probably indicates that this particualr tissue sample is wrongly classified. Tissue sample #6 has the most negative sil_width of -0.0352, indicating that it has a high chance of being classified wrongly.

(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.

```
## Apply **Agglomerative clustering** (AGNES) with Ward's method to the data
df.agnes = agnes(df, method="ward", stand=T)
pltree(df.agnes, cex=0.5, hang= -1,
  main="AGNES fit",
  xlab="Tissue samples",sub="")
```



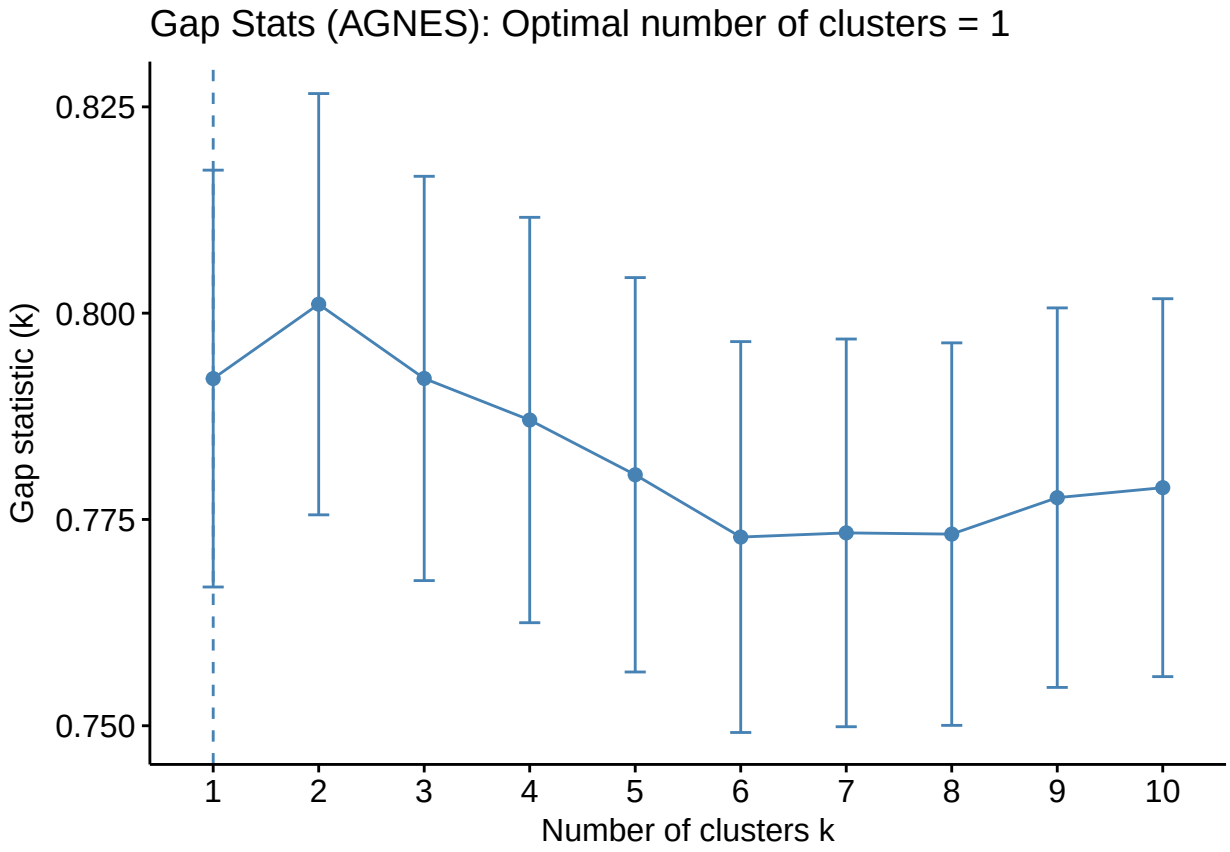

```
# Reformat agnes function for use in gap statistics
agnes.reformat<-function(x, k){
# x: Data matrix or frame, k: Number of clusters
  x.agnes = agnes(x,method="ward",stand=T)
  x.cluster = list(cluster=cutree(x.agnes,k=k))
  return(x.cluster)
}
```

```
## Determine the optimal number of clusters
# GAP Statistics
set.seed(123)
gapstat <- clusGap(scale(df), FUN=agnes.reformat, K.max=10, B=500, d.power=2)
print(gapstat, method="Tibs2001SEmax")
```

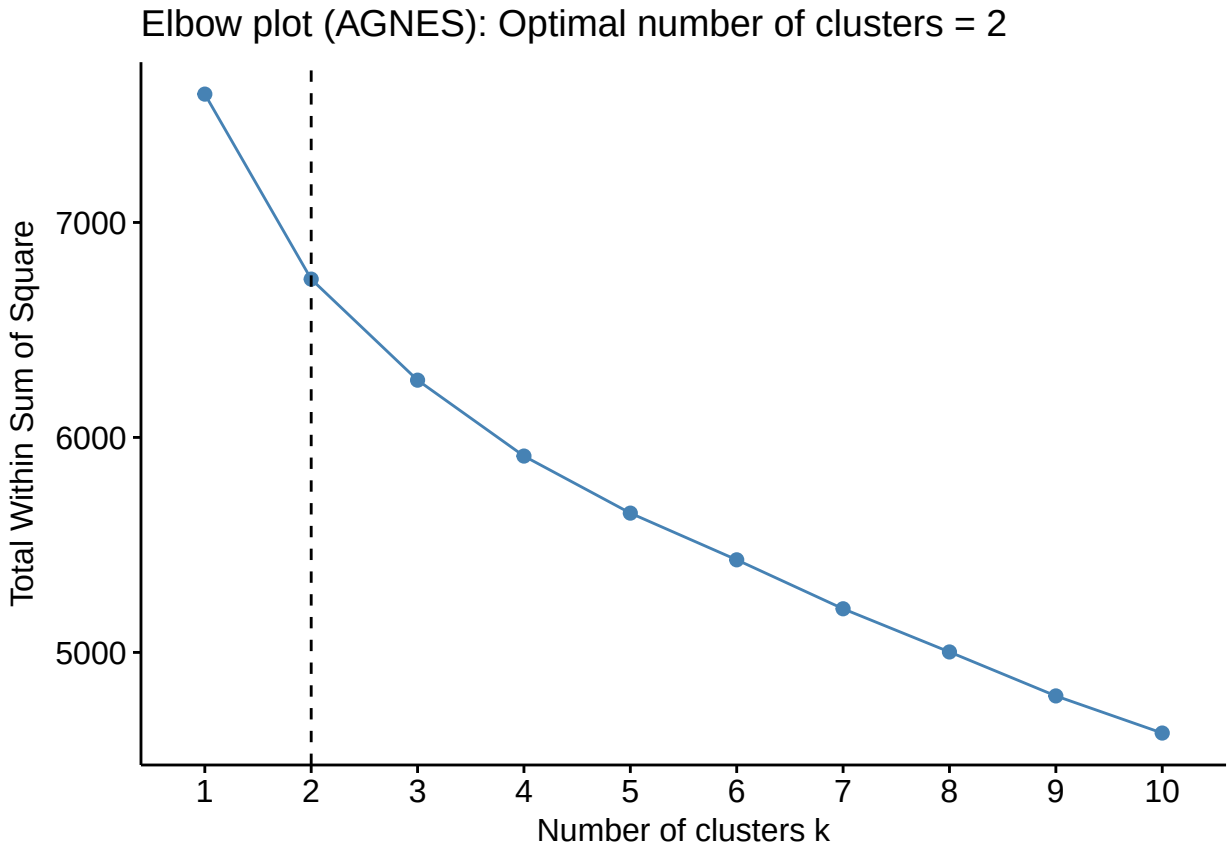
```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df), FUNcluster = agnes.reformat, K.max = 10,     B = 500, d.power = 2)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 8.242362 9.034435 0.7920736 0.02525855
##  [2,] 8.122127 8.923215 0.8010876 0.02552652
##  [3,] 8.049806 8.841894 0.7920880 0.02449844
```

```
##  [4,] 7.991815 8.778865 0.7870500 0.02456099
##  [5,] 7.945893 8.726308 0.7804144 0.02389872
##  [6,] 7.906672 8.679540 0.7728673 0.02368480
##  [7,] 7.863830 8.637204 0.7733734 0.02349186
##  [8,] 7.824460 8.597683 0.7732231 0.02317643
##  [9,] 7.782709 8.560351 0.7776419 0.02299677
## [10,] 7.746055 8.524907 0.7788522 0.02289983
```

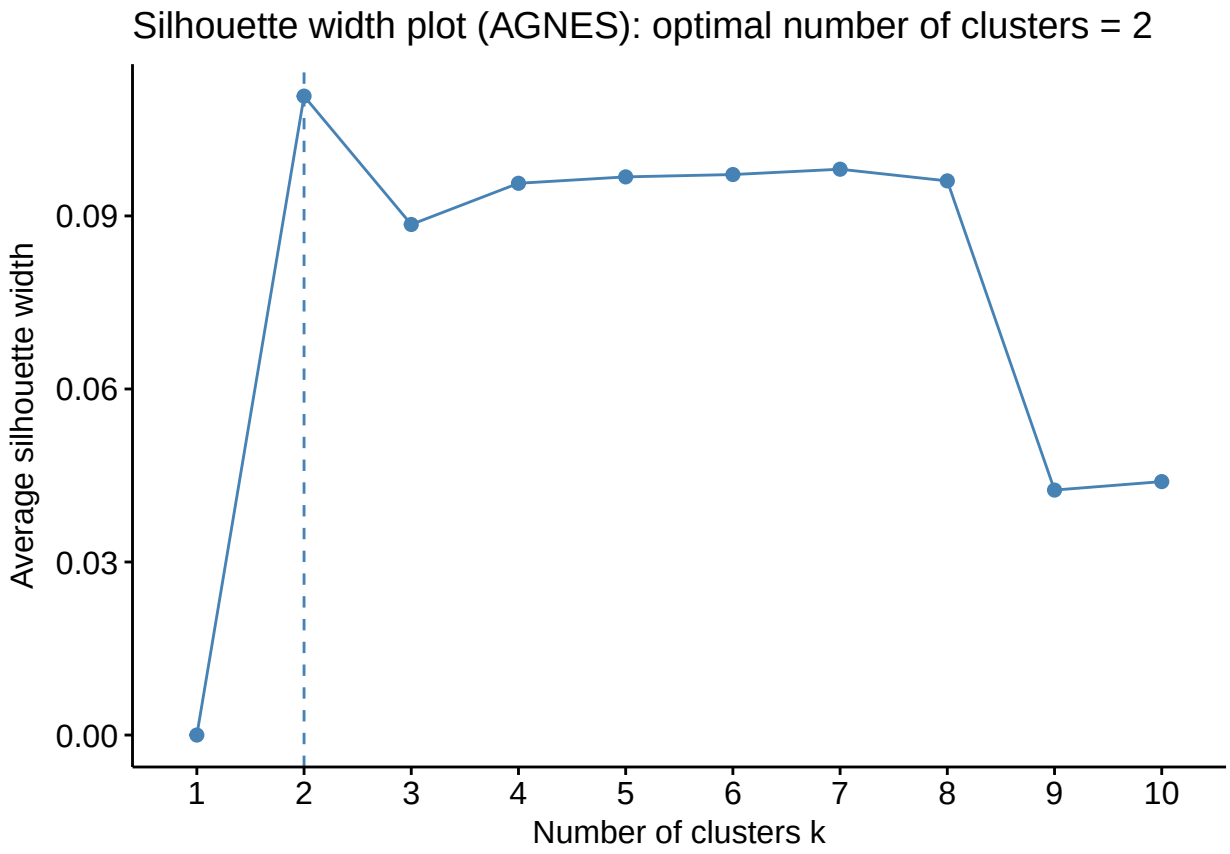
```
fviz_gap_stat(gapstat,
  maxSE=list(method="Tibs2001SEmax",SE.factor=1)) +
  ggtitle("Gap Stats (AGNES): Optimal number of clusters = 1")
```




```
# Elbow plot
fviz_nbclust(scale(df), agnes.reformat, method="wss") +
  ggtitle("Elbow plot (AGNES): Optimal number of clusters = 2") +
  geom_vline(xintercept=2,linetype=2)
```

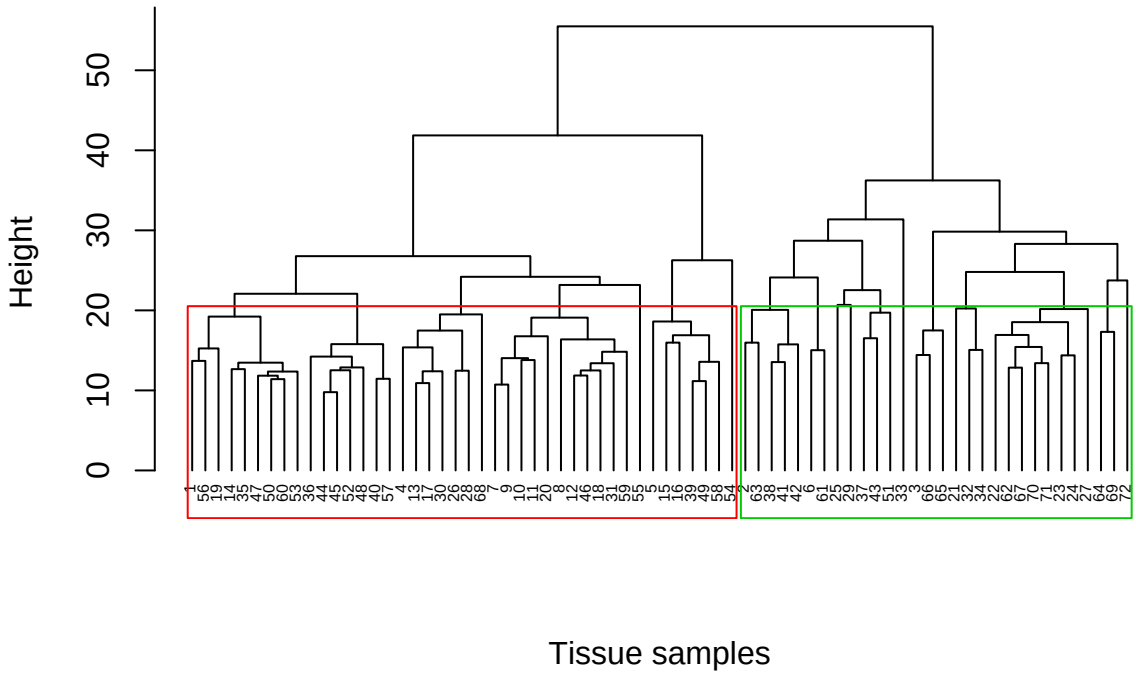



```
# Silhouette width plot
fviz_nbclust(scale(df),agnes.reformat,method="silhouette") +
  ggtitle("Silhouette width plot (AGNES): optimal number of clusters = 2")
```



Here, both the elbow method and the silhouette width plot pick the optimal number of clusters as 2, while gap statistics pick the optimal number of cluster to be 1. The reason gap statistics is picking k = 1 could be because we are using the default spaceH0 values, which compresses the data space into a hypercube. This results in a preference for in a smaller number of clusters since the algorithm ignores small differences in certain dimensions and consider possible dfferent clusters as one cluster. If we were to set spaceH0 to original, which considers all dimentions of the data space - this will result in a prefernce for a larger number of clusters, as the algorithm is able to differentiate between clusters that are slightly in and out of planes from one another. In the following, we will use k = 2 in our model fitting since it is suggested by both the elbow and the silhouette methods, and from our initial data observation, it also looks like there are two clusters.

```
## Add rectangles to the dendrograms sectioning off clusters
df.agnes = agnes(df, method="ward", stand=T)
pltree(df.agnes, cex=0.5, hang= -1,
  main="AGNES clusters",
  xlab="Tissue samples",sub="")
rect.hclust(df.agnes, k=2, border=2:3)
```

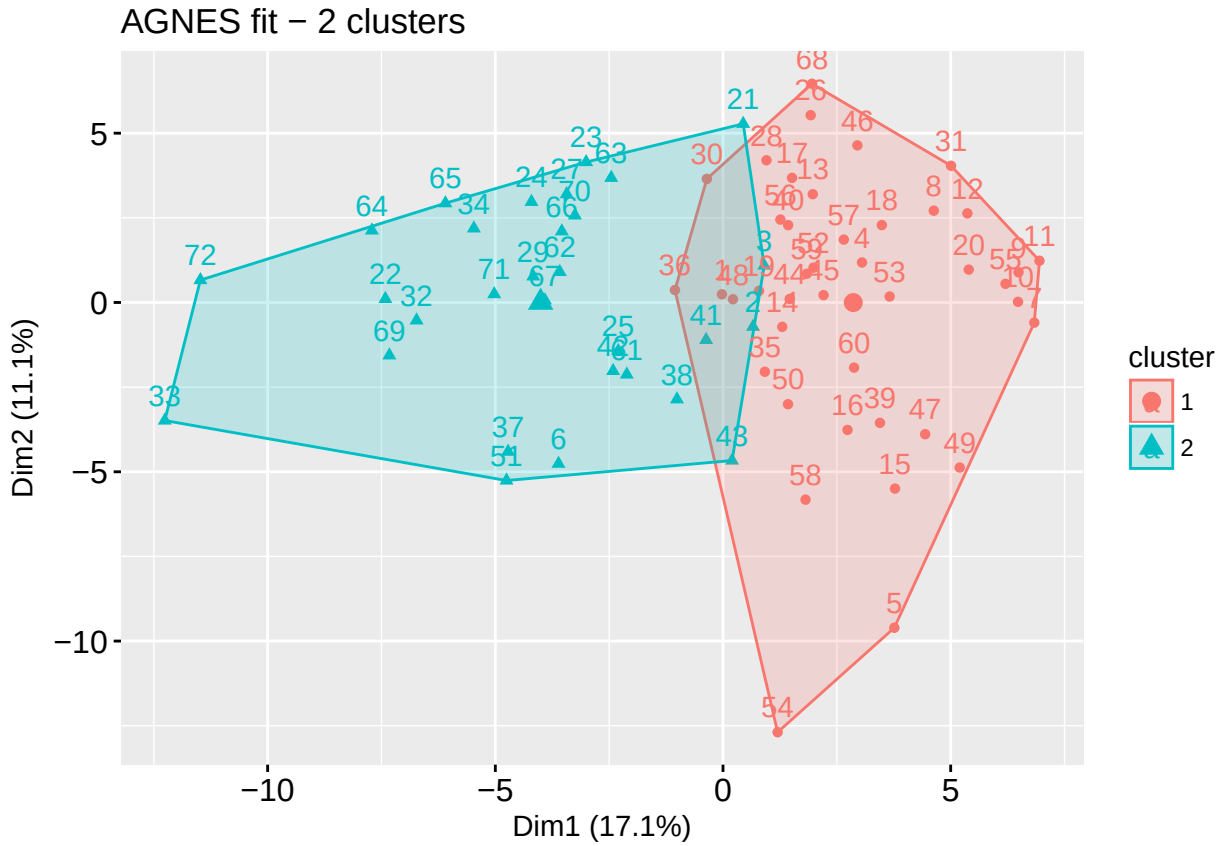


Comment on the ways (if any) the results of PAM differ from those of AGNES.

To compare the results of clustering using PAM and AGNES, is it helpful to visualize the clusters in a scatter plot. Although this is not required by the question, we plot the results of AGNES in the following. We can see that the two clusters computed by AGNES and PAM are largely similar, although AGNES has classifies some points into cluster 2 (e.g. tissue samples #2, 3, 21, 38, 41, 43, etc.) that have been classified by PAM as belonging to cluster 1. These points all lie near the interface between the two clusters. However, the classification of data points further away from the cluster sepration interface did not differ between AGNES and PAM. In general, AGNES classified more data points into cluster 2 than PAM did.

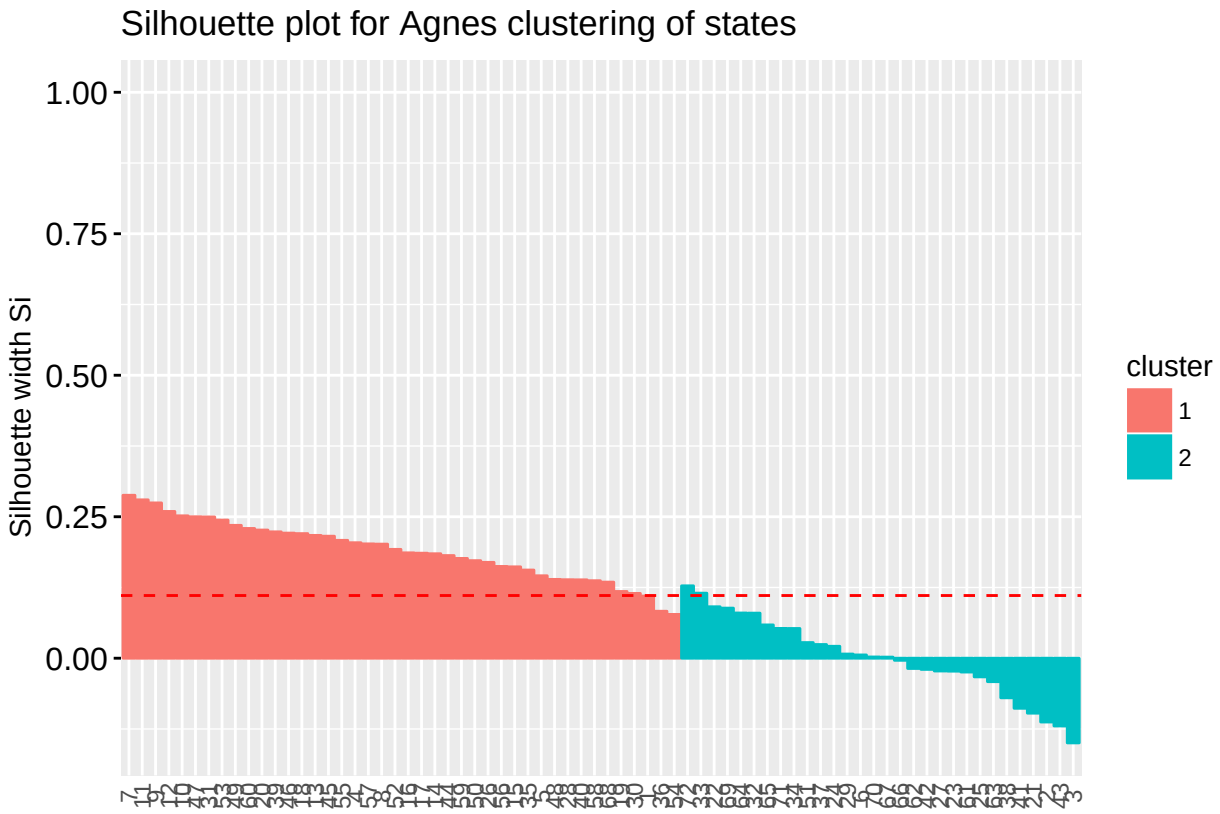
Although this is not required by the question, we graph the silhouette plot of the AGNES fit to evaluate the two clusters below. From the silhouette plot, we notice that similar to PAM, most data points in cluster have below average sil_widths, indicating that this is not a good classification. There are also multiple data points in cluster 2 that have negative sil_width, tissue samples #2, 3, 21, 38, 41, 43, which are classified as belong to cluster 2 by AGNES, but classified as belong to cluster 1 by PAM, all have negative AGNES sil_widths. This suggests that they are probably classified wrongly and are more similar to members of cluster 1 than cluster 2.

```
grp.agnes = cutree(df.agnes, k=2)
fviz_cluster(list(data = scale(df), cluster = grp.agnes),
  main="AGNES fit - 2 clusters")
```




```
df.agnes.sil <- agnes.reformat(scale(df), k = 2)
fviz_silhouette(silhouette(df.agnes.sil$cluster, dist(scale(df))),
  main="Silhouette plot for Agnes clustering of states")+
  theme(axis.text.x = element_text(angle = 90))
```

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




```
# Compute silhouette
sil.km = silhouette(df.agnes.sil$cluster, dist(scale(df)))[, 1:3]
sil.km <- data.frame(states = rownames(df), sil.km)

# Objects with negative silhouette
neg_sil_index.km = which(sil.km[, 'sil_width'] < 0)
print(sil.km[neg_sil_index.km, , drop = FALSE])
```

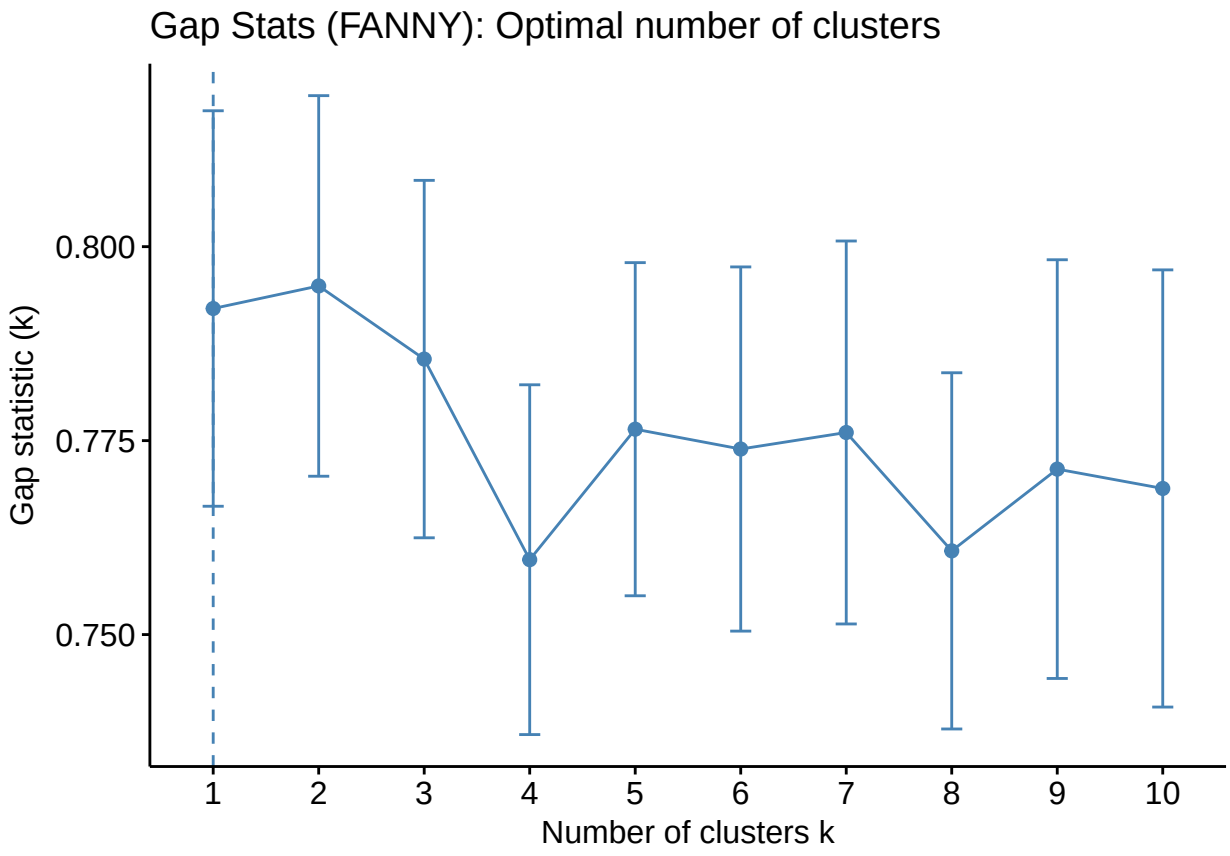
```
##    states cluster neighbor    sil_width
## 2       2       2        1 -0.112695372
## 3       3       2        1 -0.149578382
## 21     21       2        1 -0.097045433
## 23     23       2        1 -0.022839808
## 25     25       2        1 -0.032943841
## 27     27       2        1 -0.022579490
## 38     38       2        1 -0.069896052
## 41     41       2        1 -0.088440948
## 42     42       2        1 -0.019752498
## 43     43       2        1 -0.119684972
## 61     61       2        1 -0.024391819
## 62     62       2        1 -0.018090642
## 63     63       2        1 -0.041384846
## 66     66       2        1 -0.003656671
```

(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.

```r
## Determining the optimal number of clusters
# Gap statistics
set.seed(123)
gapstat <- clusGap(scale(df), FUN=fanny, K.max=10,
                   maxit = 5000, memb.exp = 1.05, d.power = 2)
print(gapstat, method="Tibs2001SEmax")
```

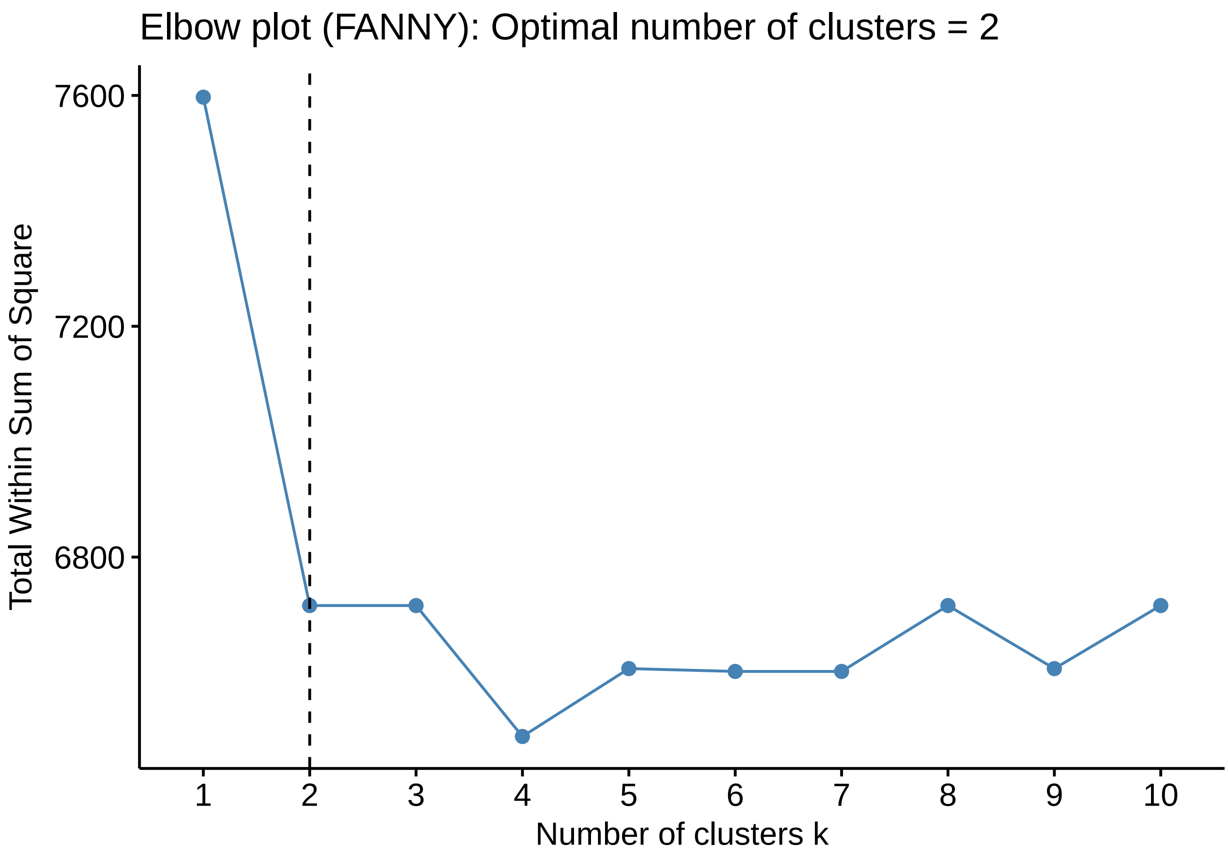
```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df), FUNcluster = fanny, K.max = 10, d.power = 2,     maxit = 5000, memb.exp = 1.05
## B=100 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 8.242362 9.034389 0.7920273 0.02548825
##  [2,] 8.110739 8.905676 0.7949370 0.02452759
##  [3,] 8.034625 8.820135 0.7855097 0.02303502
##  [4,] 7.996265 8.755919 0.7596534 0.02254220
##  [5,] 7.932486 8.708959 0.7764726 0.02146929
##  [6,] 7.895727 8.669644 0.7739169 0.02347028
##  [7,] 7.862596 8.638644 0.7760477 0.02467948
##  [8,] 7.850659 8.611448 0.7607892 0.02295633
##  [9,] 7.814209 8.585536 0.7713268 0.02698462
## [10,] 7.799665 8.568496 0.7688318 0.02817260
```

```r
fviz_gap_stat(gapstat,
  maxSE=list(method="Tibs2001SEmax",SE.factor=1)) +
  ggtitle("Gap Stats (FANNY): Optimal number of clusters")
```



```r
# Elbow plot
fviz_nbclust(scale(df), fanny, method="wss") +
  ggtitle("Elbow plot (FANNY): Optimal number of clusters = 2")+
  geom_vline(xintercept=2,linetype=2)
```

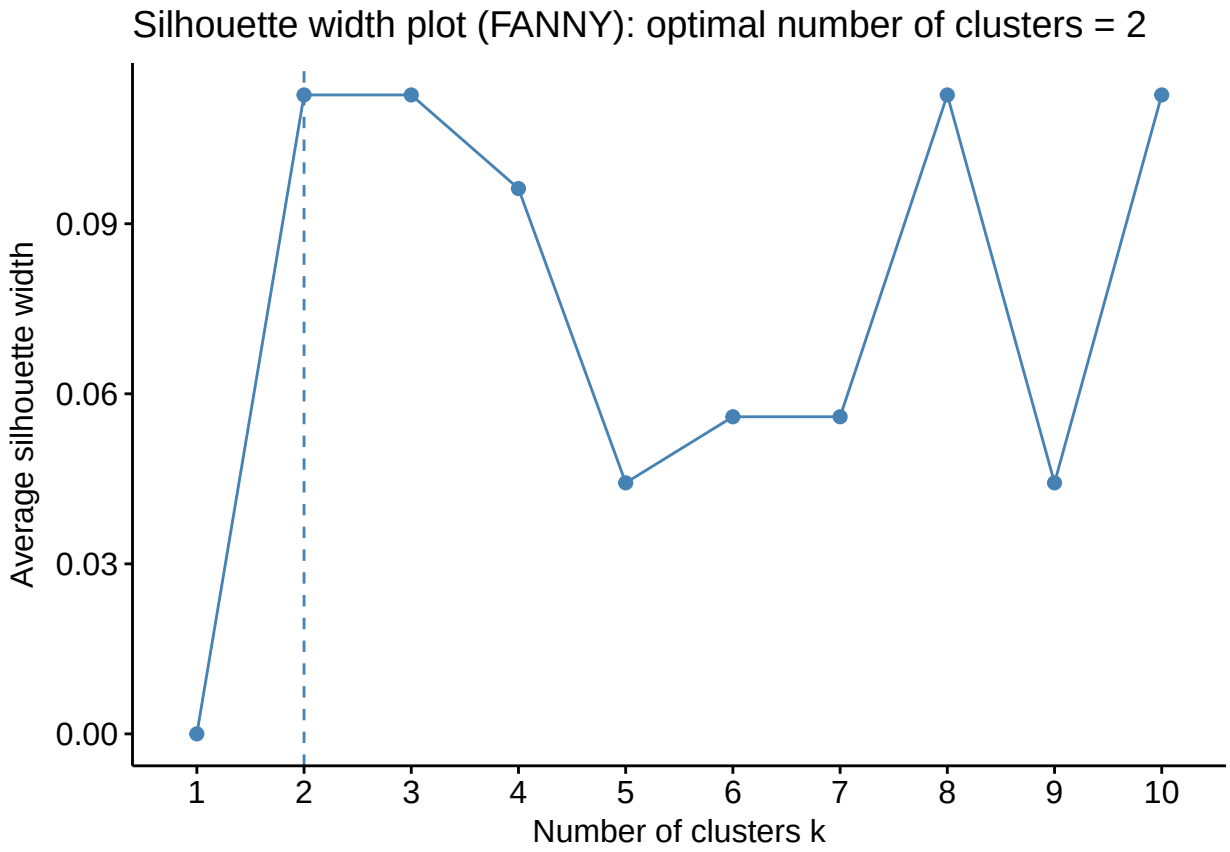
```
## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?
```




```
# Silhouette width plot
fviz_nbclust(scale(df), fanny, method="silhouette") +
  ggtitle("Silhouette width plot (FANNY): optimal number of clusters = 2")
```

```
## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?

## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?
```

```
## Warning in FUNcluster(x, i, ...): the memberships are all very close to 1/
## k. Maybe decrease 'memb.exp' ?
```



Both the elbow plot and the silhouette width method suggest the optimal cluster as 2, gap statistics suggests the optimal cluster to use as 1. We shall proceed to use k = 2 in the following by similar reasons as in the previous part.

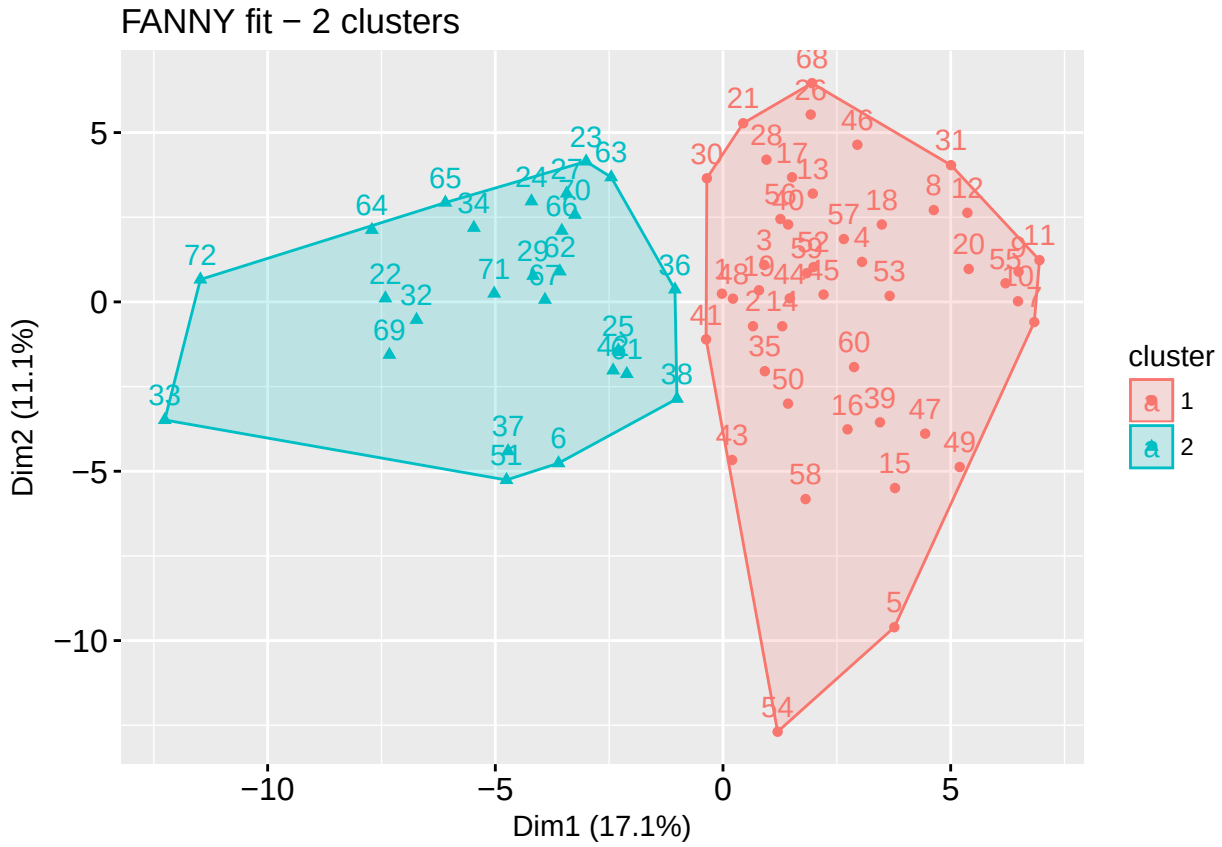
```
## Apply **Fuzzy clustering** (FANNY) to the data
df.fanny = fanny(scale(df), k=2, memb.exp = 1.05)
print((round(df.fanny$membership,3)))
```

```
##        [,1]  [,2]
##  [1,] 0.899 0.101
##  [2,] 0.915 0.085
##  [3,] 0.973 0.027
##  [4,] 0.999 0.001
##  [5,] 0.978 0.022
##  [6,] 0.017 0.983
##  [7,] 1.000 0.000
##  [8,] 0.999 0.001
##  [9,] 1.000 0.000
## [10,] 1.000 0.000
## [11,] 1.000 0.000
## [12,] 1.000 0.000
## [13,] 1.000 0.000
## [14,] 0.997 0.003
## [15,] 0.990 0.010
```

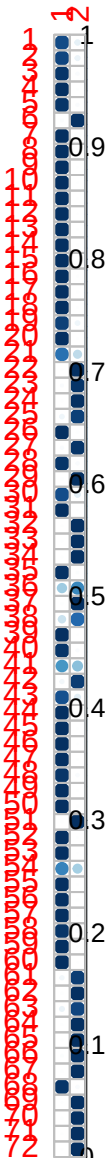
```
## [16,] 0.995 0.005
## [17,] 0.994 0.006
## [18,] 1.000 0.000
## [19,] 0.930 0.070
## [20,] 1.000 0.000
## [21,] 0.743 0.257
## [22,] 0.000 1.000
## [23,] 0.021 0.979
## [24,] 0.003 0.997
## [25,] 0.068 0.932
## [26,] 0.992 0.008
## [27,] 0.028 0.972
## [28,] 0.973 0.027
## [29,] 0.009 0.991
## [30,] 0.888 0.112
## [31,] 1.000 0.000
## [32,] 0.000 1.000
## [33,] 0.001 0.999
## [34,] 0.002 0.998
## [35,] 0.989 0.011
## [36,] 0.352 0.648
## [37,] 0.007 0.993
## [38,] 0.210 0.790
## [39,] 1.000 0.000
## [40,] 0.978 0.022
## [41,] 0.586 0.414
## [42,] 0.051 0.949
## [43,] 0.873 0.127
## [44,] 0.998 0.002
## [45,] 1.000 0.000
## [46,] 1.000 0.000
## [47,] 1.000 0.000
## [48,] 0.972 0.028
## [49,] 1.000 0.000
## [50,] 0.995 0.005
## [51,] 0.007 0.993
## [52,] 0.998 0.002
## [53,] 1.000 0.000
## [54,] 0.669 0.331
## [55,] 0.999 0.001
## [56,] 0.990 0.010
## [57,] 0.999 0.001
## [58,] 0.975 0.025
## [59,] 0.995 0.005
## [60,] 1.000 0.000
## [61,] 0.055 0.945
## [62,] 0.006 0.994
## [63,] 0.083 0.917
## [64,] 0.001 0.999
## [65,] 0.003 0.997
## [66,] 0.015 0.985
## [67,] 0.003 0.997
## [68,] 0.971 0.029
## [69,] 0.001 0.999
```

```
## [70,] 0.007 0.993
## [71,] 0.001 0.999
## [72,] 0.000 1.000
```

```
## Summarize the results using a principal components plot
fviz_cluster(df.fanny, main="FANNY fit - 2 clusters")
```




```
## A correlation plot of the cluster membership weights
library(corrplot)
corrplot(df.fanny$membership, is.corr=F)
```



Based on the cluster membership weights, do you think it makes sense to consider summarizing the results using a principal components plot? Briefly justify.

Based on the cluster membership weights, it makes sense to summarize the results using a principal components plot. This is because most data points have a high probability of belong to either of the two clusters, as indicated by a dark blue circle in the weights table. Some datapoints, such as tissue sample #36, #41, for instane, each has ligher blue circles under both cluster 1 and cluster 2, indicating a similar probability of belong to either class. Data points like these are eventually classified into the cluster that has slightly higher probability / darker blue circle. By plotting them in a scatter plot, we can see that most of these points lie near the interface between the two clusters, which makes sense. The scatter plot is a way to visualize the distances between data points which gives us an idea about how different/far away these points are from each other.

(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?

Answer

We use the results from the PAM method in our analysis below.

```
## PAM
# create a dataframe containing columns 2:4 of original data, and clustering result
pam.cluster <- df.pam$clustering
pam.all <- data.frame(data[, 2:4],
                      cluster = pam.cluster)

cluster1 <- pam.all[pam.all$cluster == 1,]
cluster2 <- pam.all[pam.all$cluster == 2,]
```

```
## what proportion of each cluster are ALL (Acute Lympohblastic Leukemia) samples?
# cluster 1
print(paste("Perportion of 'ALL' in cluster 1:", round(mean(cluster1$ALL.AML == "ALL"), 3)))
```

```
## [1] "Perportion of 'ALL' in cluster 1: 0.863"
```

```
# cluster 2
print(paste("Perportion of 'ALL' in cluster 2:", round(mean(cluster2$ALL.AML == "ALL"), 3)))
```

```
## [1] "Perportion of 'ALL' in cluster 2: 0.143"
```

86.3% of cluster 1 are ALL (Acute Lympohblastic Leukemia) samples, while ony 14.3% of cluster 2 are ALL samples. This shows that cluster 1 is made up of mostly tissue samples from ALL patients, while cluster 2 is mostly made up of tissue samples from AML patients.

```
## In each cluster, what proportion are samples belonging to female subjects?
# cluster 1
print(paste("Perportion of Female in cluster 1:", round(mean(cluster1$Gender == "F"), 3)))
```

```
## [1] "Perportion of Female in cluster 1: 0.412"
```

```
print(paste("Perportion of Male in cluster 1:", round(mean(cluster1$Gender == "M"), 3)))
```

```
## [1] "Perportion of Male in cluster 1: 0.412"
```

```
print(paste("Perportion of data with missing gender infomation in cluster 1:",
            round(mean(cluster1$Gender == ""), 3)))
```

```
## [1] "Perportion of data with missing gender infomation in cluster 1: 0.176"
```

```
# cluster 2
print(paste("Perportion of Female' in cluster 2:", round(mean(cluster2$Gender == "F"), 3)))
```

```
## [1] "Perportion of Female' in cluster 2: 0.095"
```

```
print(paste("Perportion of Male' in cluster 2:", round(mean(cluster2$Gender == "M"), 3)))
```

```
## [1] "Perportion of Male' in cluster 2: 0.238"
```

```
print(paste("Perportion of data with missing gender infomation in cluster 2:",
            round(mean(cluster2$Gender == ""), 3)))
```

```
## [1] "Perportion of data with missing gender infomation in cluster 2: 0.667"
```

```
summary(cluster2)
```

```
##  ALL.AML  BM.PB   Gender    cluster
##  ALL: 3   BM:18    :14   Min.   :2
```

```
##  AML:18   PB: 3   F: 2   1st Qu.:2
##                   M: 5   Median :2
##                          Mean   :2
##                          3rd Qu.:2
##                          Max.   :2
```

In cluster 1, 41.2% of the samples came from female patients, while equal percentage of samples came from male patients. 17.6% of the data had no information on patient gender. Since there is only a small amount of missing information, we can conclude that both female and male patients are equally likely to be in cluster 1. Combining with our conclusion from the ALL/AML analysis, we may say that both genders are equally likely in getting ALL.

In cluster 2, 9.5% of the sample came from female patients, 23.8% of the samples came from male patients, while 66.7% of the data had missing information on patient gender. Given the available data, it may seem that the number of males in cluster 2 is 2.5 times that of the number of females. However, among the 21 data points in cluster 2, we only have gender information on 7 patients, this is not quite enough data to conclude anything about if either gender is more likely to suffer from AML.

```
## In each cluster, what proportion of the samples were taken
## from bone marrow as opposed to peripheral blood?
# cluster 1
print(paste("Perportion of bone marrow sample in cluster 1:",
            round(mean(cluster1$BM.PB == "BM"), 3)))
```

```
## [1] "Perportion of bone marrow sample in cluster 1: 0.863"
```

```
# cluster 2
print(paste("Perportion of bone marrow sample in cluster 2:",
            round(mean(cluster2$BM.PB == "BM"), 3)))
```

```
## [1] "Perportion of bone marrow sample in cluster 2: 0.857"
```

86.3% of cluster 1 samples came from bone marrow, and 85.7% of cluster 2 samples came from bone marrow. There is a similar proportion of bone marrow samples in both clusters. Majority of the samples in this study came from boon marrow. Whether the sample came from bone marrow or peropheral blood does not make a difference in detecting ALL or AML.

Overall, cluster 1 seems to contain most patients with ALL, while cluster 2 seem to contain most patients with AML. Both females and males are equally likely in suffering from ALL, but we do not have enough information to make a conclusion for AML patients. In addition, bone marrow samples consists of the majority of these tissue samples, and there is a similar proportion of bone marrow samples in both clusters, indicating that method of sample extraction does not make a difference in detecting either form of leukemias.

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.

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.

(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 <- data[data$Train.Test == "Train", c(2, seq(5, 111))]
test <- data[data$Train.Test == "Test", c(2, seq(5, 111))]
```

```
## Fit two SVM models with linear and RBF kernels to the training set
library('e1071')
library('caret')
```

```
## Loading required package: lattice
```

Linear kernel

```
## Linear model
# Tune for best parameter cost
linear.tune <- tune(svm,
                ALL.AML ~ .,
                data = train,
                kernel = "linear",
                ranges = list(cost = 10^seq(-5, 0)),
                tunecontrol = tune.control(cross = 5))

linear.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0
```

```
# linear svm evaluation
str(linear.tune$performances)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ cost      : num  1e-05 1e-04 1e-03 1e-02 1e-01 1e+00
##  $ error     : num  0.286 0.286 0.286 0.025 0 ...
##  $ dispersion: num  0.2016 0.2016 0.2016 0.0559 0 ...
```

```
# best model
cat('Best Model:\n')
```

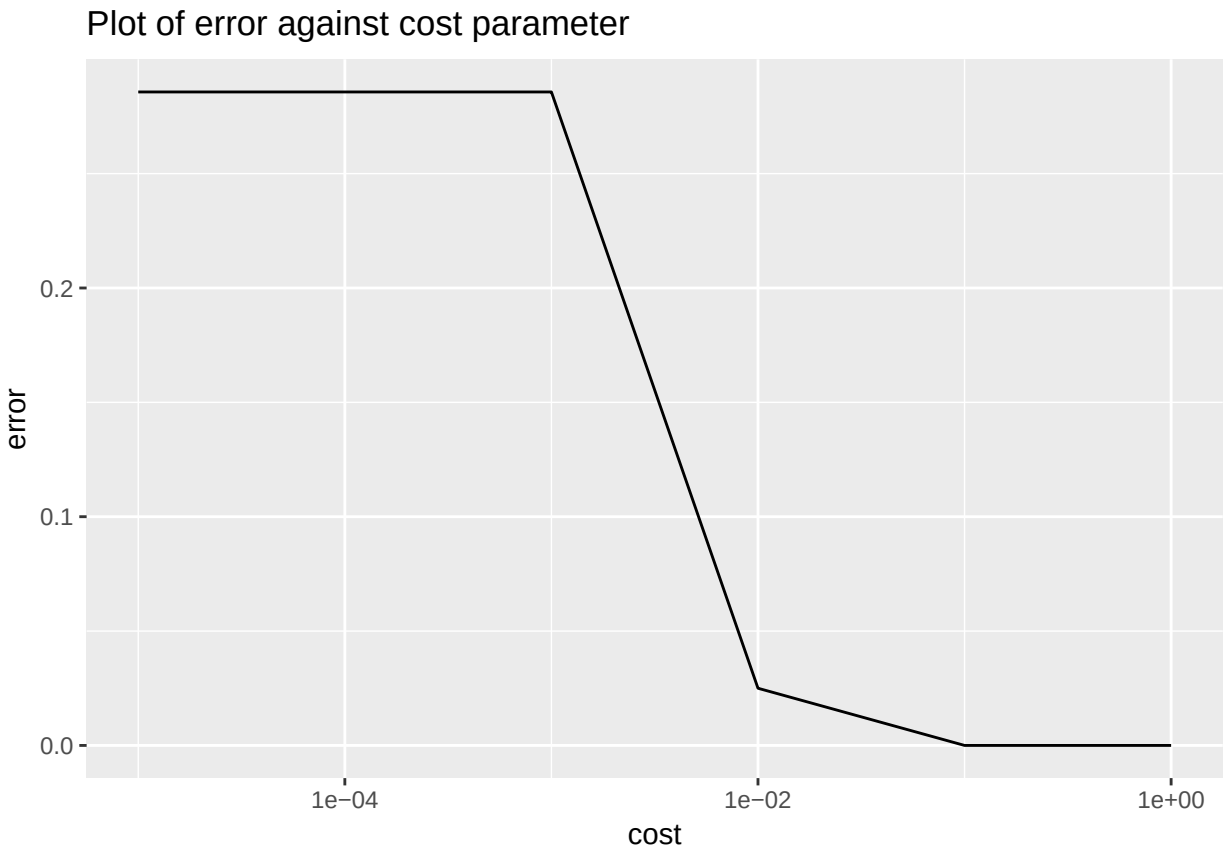
```
## Best Model:
```

```
linear.tune$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = ALL.AML ~ ., data = train,
##     ranges = list(cost = 10^seq(-5, 0)), tunecontrol = tune.control(cross = 5),
##     kernel = "linear")
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.009345794
##
## Number of Support Vectors:  22
```

```
# visualization
ggplot(linear.tune$performances, mapping = aes(x = cost, y = error)) +
  geom_line()+
  scale_x_log10()+
  ggtitle("Plot of error against cost parameter")
```




```
# linear svm prediction
pred.linear.test <- predict(linear.tune$best.model, newdata = test)
confusionMatrix(pred.linear.test, test$ALL.AML)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ALL AML
##        ALL  20   4
##        AML   0  10
##
##                Accuracy : 0.8824
##                  95% CI : (0.7255, 0.967)
```

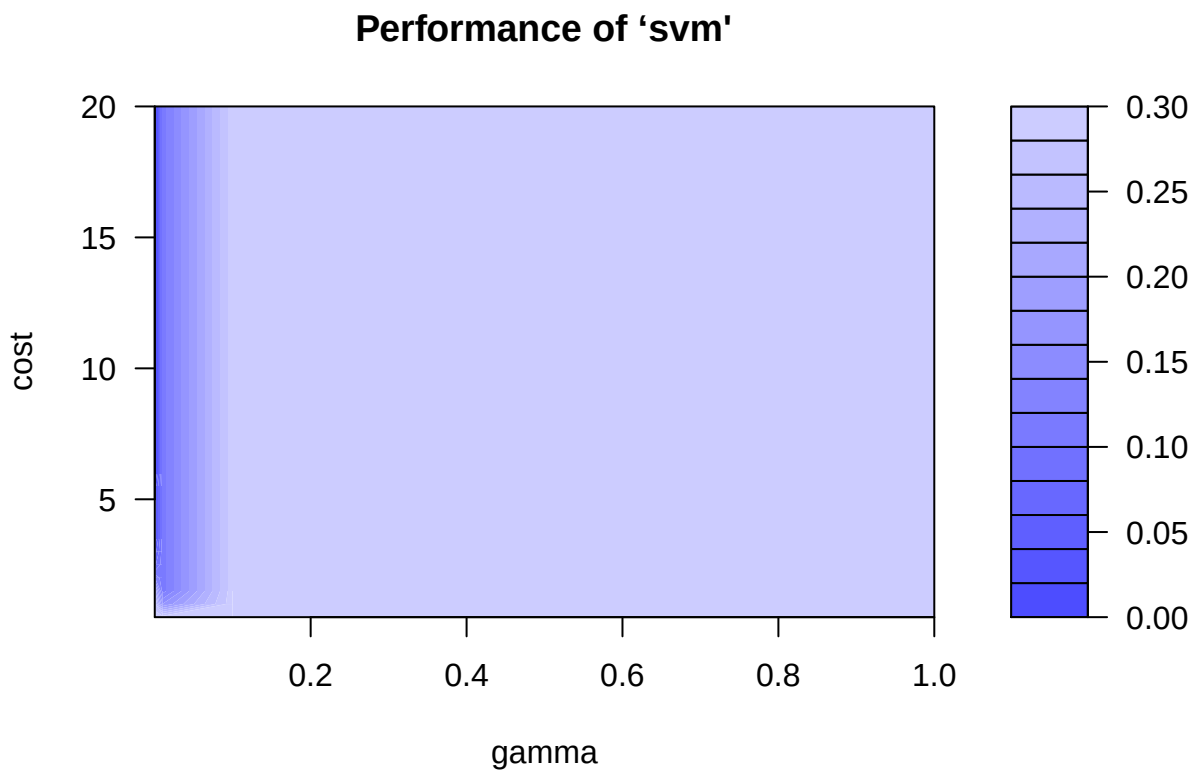
```
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : 0.0001971
##
##                   Kappa : 0.7463
##  Mcnemar's Test P-Value : 0.1336144
##
##             Sensitivity : 1.0000
##             Specificity : 0.7143
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5882
##          Detection Rate : 0.5882
##    Detection Prevalence : 0.7059
##       Balanced Accuracy : 0.8571
##
##        'Positive' Class : ALL
##
```

RBF kernel

```
## RBF tune
rbf.tune <- tune(svm,
                 ALL.AML ~ .,
                 data = train,
                 kernel = "radial",
                 ranges = list(gamma = 10^seq(-6, 0),
                               cost = seq(0.5, 20, by = 0.5)),
                 tunecontrol = tune.control(cross = 5))
rbf.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  gamma cost
##  0.001    6
##
## - best performance: 0
```

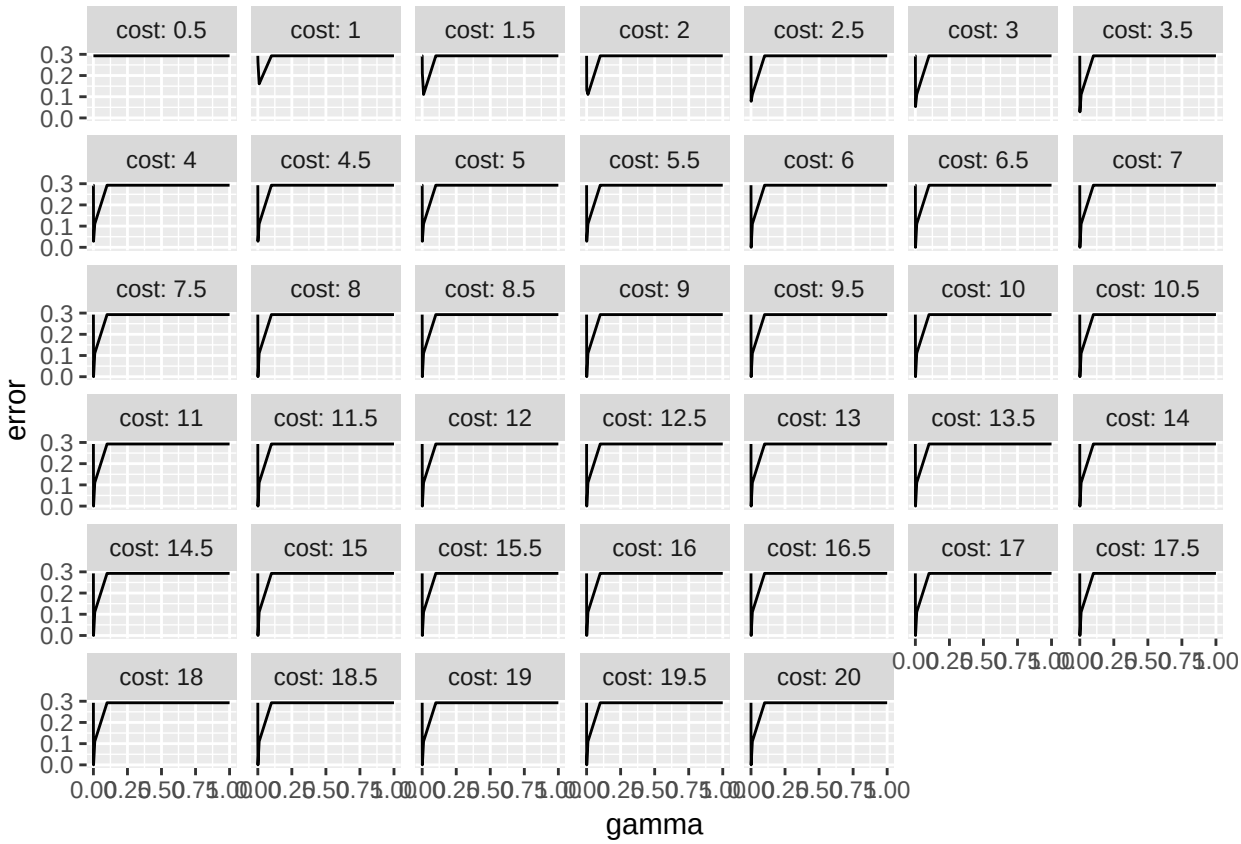
```
# rbf svm evaluation
plot(rbf.tune)
```




```
# best parameters
str(rbf.tune$performances)
```

```
## 'data.frame':    280 obs. of  4 variables:
##  $ gamma     : num  1e-06 1e-05 1e-04 1e-03 1e-02 1e-01 1e+00 1e-06 1e-05 1e-04 ...
##  $ cost      : num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 1 1 ...
##  $ error     : num  0.293 0.293 0.293 0.293 0.293 ...
##  $ dispersion: num  0.162 0.162 0.162 0.162 0.162 ...
```

```
# visualization
ggplot(rbf.tune$performances,
       mapping = aes(x = gamma, y = error)) +
  geom_line() +
  facet_wrap(~cost, labeller = label_both)
```




```
# best model
cat('Best Model:\n')
```

```
## Best Model:
```

```
rbf.tune$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = ALL.AML ~ ., data = train,
##     ranges = list(gamma = 10^seq(-6, 0), cost = seq(0.5, 20,
##         by = 0.5)), tunecontrol = tune.control(cross = 5), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  6
##       gamma:  0.001
##
## Number of Support Vectors:  23
```

```
# rbf svm prediction
pred.rbf.test <- predict(rbf.tune$best.model, newdata = test)
confusionMatrix(pred.rbf.test, test$ALL.AML)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction ALL AML
##        ALL  20   4
##        AML   0  10
##
##                Accuracy : 0.8824
##                  95% CI : (0.7255, 0.967)
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : 0.0001971
##
##                   Kappa : 0.7463
##  Mcnemar's Test P-Value : 0.1336144
##
##             Sensitivity : 1.0000
##             Specificity : 0.7143
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5882
##          Detection Rate : 0.5882
##    Detection Prevalence : 0.7059
##       Balanced Accuracy : 0.8571
##
##        'Positive' Class : ALL
##
```

Explain in words how linear and RBF kernels differ as part of the SVM.

Both the linear kernel and the rbf kernel are functions that quantify the similarity of two observations. The linear kernel is when the support vector classifier is linear in the features. The linear kernel quantifies the similarity of a pair of observations using the Pearson correlation. The rbf kernel is a nonlinear kernel, where when predicting on a test observation, training observations that are far from the test observation play essentially no role in the predicted class label for this test observation, while training observations nearer to the test observation play a much larger role. This means that the radial kernel has very local behavior, in the sense that only nearby training observations have an effect on the class label of a test observation. Other than that, the linear kernel computes faster than the rbf kernel, but is never more accurate than the rbf kernel.

When using the linear kernel, we tune the parameter cost. A larger cost parameter gives emphasis on the accuracy of the fit and creates a softer margin. However, when cost is too large, the model will overfit to the training data.

When using the rbf kernel, we tune both the `cost` parameter and the `gamma` parameter. We can think of gamma as a parameter for the radial function that sits on top of each training data points that provides information when we try to classify a new point. The larger the value of gamma, the narrower and more peaked is the radial function, this means that it has a smaller radius of influence for nearby points. A large gamma with narrow peaks will fit well on the training data set but provides little information on areas that are far away from the peaks or in between the peaks. Hence, when using the model to fit on the test dataset, the error increases for we have little information on most of the areas. On the other hand, when gamma is small, the radial functions are broad and have a greater radius of influence on the nearby area. However, each of these functions sitting on top of different training data points may interfere with one another, for a point that is sitting in between two peaks, both have a moderate effect on the point, this cause the model to break down at some point and the error will increase as gamma becomes smaller and smaller.

Here, our 5-fold corss validation picks the best cost of 0.1 for the linear kernel, and picks the best gamma of 0.001 and a cost of 4. This gives an overall accuracy of 88.24%, with 100% accuracy on ALL and 71.4% accuracy on AML for both kernels. This goes to show that a linear kernel is probably good enough in this case, since it gives the same accuracy as the rbf kernel.

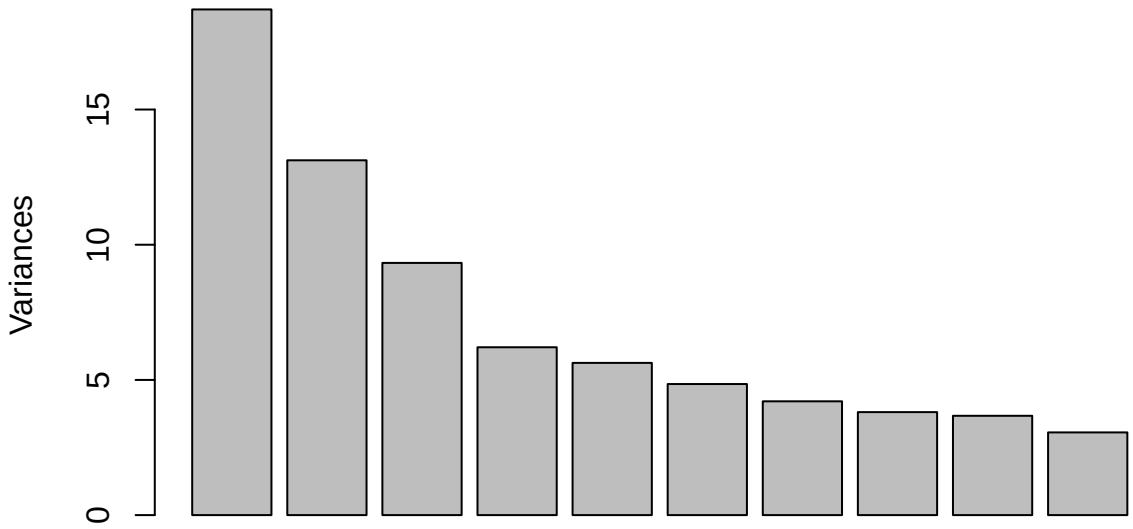
(b) (10 points) Apply principal component analysis (PCA) to the genetic expression values in the training set, and retain the minimal number of PCs that capture at least 90% of the variance in the data. How does the number of PCs identified compare with the total number of gene expression values? Apply to the test data the rotation that resulted in the PCs in the training data, and keep the same set of PCs.

Answer

```
## 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.
train.prc <- prcomp(train[, -1], center = TRUE,scale = TRUE)
summary(train.prc)
```

```
## Importance of components:
##                            PC1    PC2     PC3     PC4     PC5    PC6
## Standard deviation     4.3245 3.6224 3.05397 2.49145 2.37256 2.2016
## Proportion of Variance 0.1748 0.1226 0.08717 0.05801 0.05261 0.0453
## Cumulative Proportion  0.1748 0.2974 0.38458 0.44259 0.49520 0.5405
##                            PC7    PC8     PC9    PC10    PC11    PC12
## Standard deviation     2.05147 1.9517 1.91651 1.74869 1.70042 1.61958
## Proportion of Variance 0.03933 0.0356 0.03433 0.02858 0.02702 0.02451
## Cumulative Proportion  0.57983 0.6154 0.64976 0.67833 0.70536 0.72987
##                           PC13    PC14    PC15    PC16    PC17    PC18
## Standard deviation     1.59408 1.50671 1.41424 1.35926 1.33340 1.30885
## Proportion of Variance 0.02375 0.02122 0.01869 0.01727 0.01662 0.01601
## Cumulative Proportion  0.75362 0.77484 0.79353 0.81080 0.82741 0.84342
##                           PC19    PC20    PC21    PC22   PC23    PC24
## Standard deviation     1.25983 1.24862 1.17165 1.15853 1.1140 1.05631
## Proportion of Variance 0.01483 0.01457 0.01283 0.01254 0.0116 0.01043
## Cumulative Proportion  0.85826 0.87283 0.88566 0.89820 0.9098 0.92023
##                           PC25    PC26    PC27    PC28    PC29    PC30
## Standard deviation     1.04208 0.98638 0.96212 0.89937 0.88640 0.83620
## Proportion of Variance 0.01015 0.00909 0.00865 0.00756 0.00734 0.00653
## Cumulative Proportion  0.93037 0.93947 0.94812 0.95568 0.96302 0.96956
##                           PC31    PC32    PC33    PC34    PC35    PC36
## Standard deviation     0.81490 0.74759 0.73278 0.68477 0.64268 0.58092
## Proportion of Variance 0.00621 0.00522 0.00502 0.00438 0.00386 0.00315
## Cumulative Proportion  0.97576 0.98099 0.98600 0.99039 0.99425 0.99740
##                           PC37     PC38
## Standard deviation     0.5274 1.92e-15
## Proportion of Variance 0.0026 0.00e+00
## Cumulative Proportion  1.0000 1.00e+00
```

```
# Plotting the first ten principal components
plot(train.prc)
```




```
# 90% cumulative variance
cumvar <- summary(train.prc)$importance[3,]
cumvar[cumvar<=0.9]
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
## 0.17478 0.29741 0.38458 0.44259 0.49520 0.54050 0.57983 0.61543 0.64976
##    PC10    PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18
## 0.67833 0.70536 0.72987 0.75362 0.77484 0.79353 0.81080 0.82741 0.84342
##    PC19    PC20    PC21    PC22
## 0.85826 0.87283 0.88566 0.89820
```

```
print('Retain 23 PC componenets.')
```

```
## [1] "Retain 23 PC componenets."
```

```
# reduced train data in coordinates of the top 23 PCs
train.reduced <- data.frame(ALL.AML = train$ALL.AML,
                            train.prc$x[, seq(1,23)])
```

How does the number of PCs identified compare with the total number of gene expression values?

The number of PC s identified to capture 90% of variance is only 23, which is much smaller tha the total number of gene expression values of 107. We have used PCA to reduce the dimension down by 4.65 times. Since we only have 72 observations, 107 features is clearly too many dimensions and may result in overfitting. With only 23 PCs, the number of features is more reasonable.

```
## Apply to the test data the rotation that resulted in the PCs
## in the training data, and keep the same set of PCs
test.reduced <- data.frame(ALL.AML = test$ALL.AML,
  scale(test[, -1]) %*% train.prc$rotation[, seq(1, 23)])
```

(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.

Answer

Linear kernel

```
# Tune for best parameter cost
set.seed(123)
reduced.linear.tune <- tune(svm,
                ALL.AML ~ .,
                data = train.reduced,
                kernel = "linear",
                ranges = list(cost = 10^seq(-5, 0)),
                tunecontrol = tune.control(cross = 5))

reduced.linear.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.05357143
```

```
# linear svm evaluation
str(reduced.linear.tune$performances)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ cost      : num  1e-05 1e-04 1e-03 1e-02 1e-01 1e+00
##  $ error     : num  0.2893 0.2893 0.2893 0.2893 0.0536 ...
##  $ dispersion: num  0.1351 0.1351 0.1351 0.1351 0.0736 ...
```

```
# best model
cat('Best Model:\n')
```

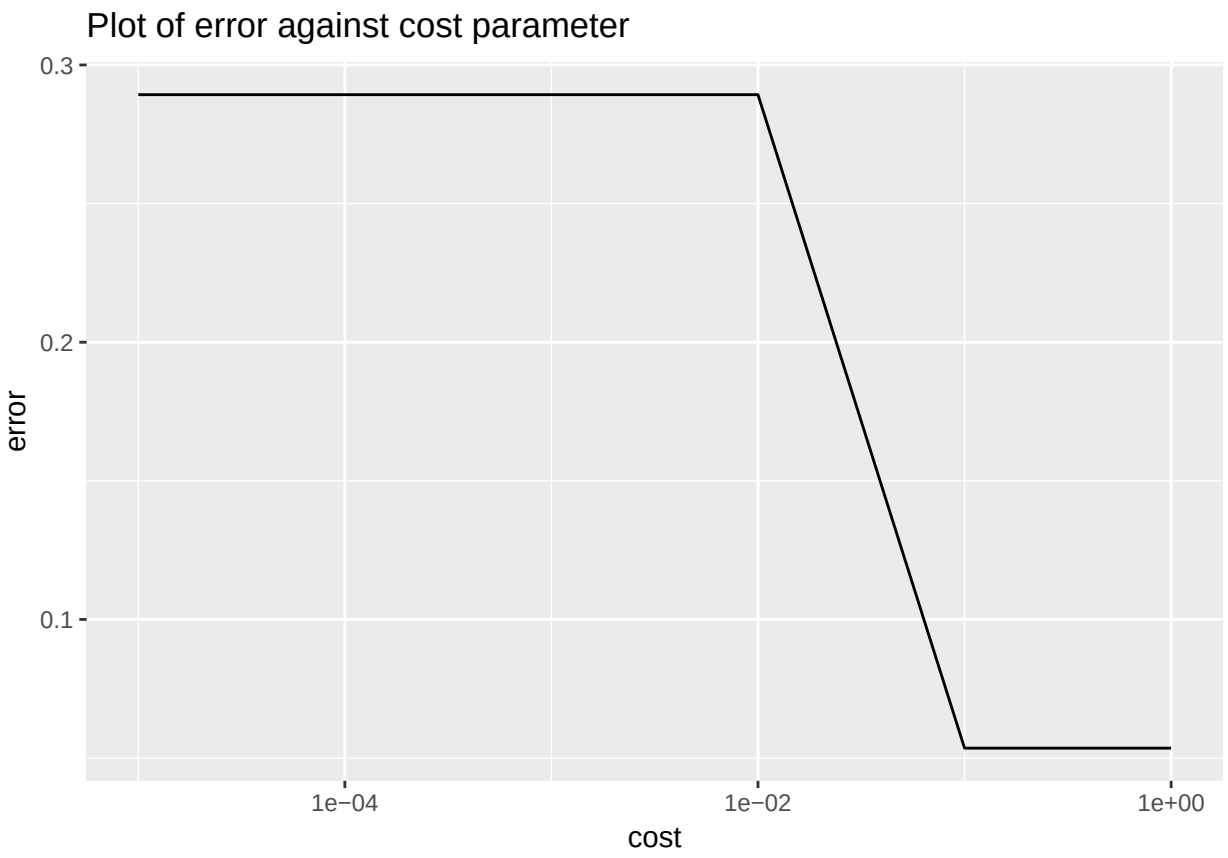
```
## Best Model:
```

```
reduced.linear.tune$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = ALL.AML ~ ., data = train.reduced,
##     ranges = list(cost = 10^seq(-5, 0)), tunecontrol = tune.control(cross = 5),
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.04347826
```

```
##
## Number of Support Vectors:  23
```

```
# visualization
ggplot(reduced.linear.tune$performances, mapping = aes(x = cost, y = error)) +
  geom_line()+
  scale_x_log10()+
  ggtitle("Plot of error against cost parameter")
```




```
# linear svm prediction
pred.reduced.linear.test <- predict(reduced.linear.tune$best.model, newdata = test.reduced)
confusionMatrix(pred.reduced.linear.test, test.reduced$ALL.AML)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ALL AML
##        ALL  20   6
##        AML   0   8
##
##                Accuracy : 0.8235
##                  95% CI : (0.6547, 0.9324)
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : 0.003193
##
##                   Kappa : 0.6107
##  Mcnemar's Test P-Value : 0.041227
##
```

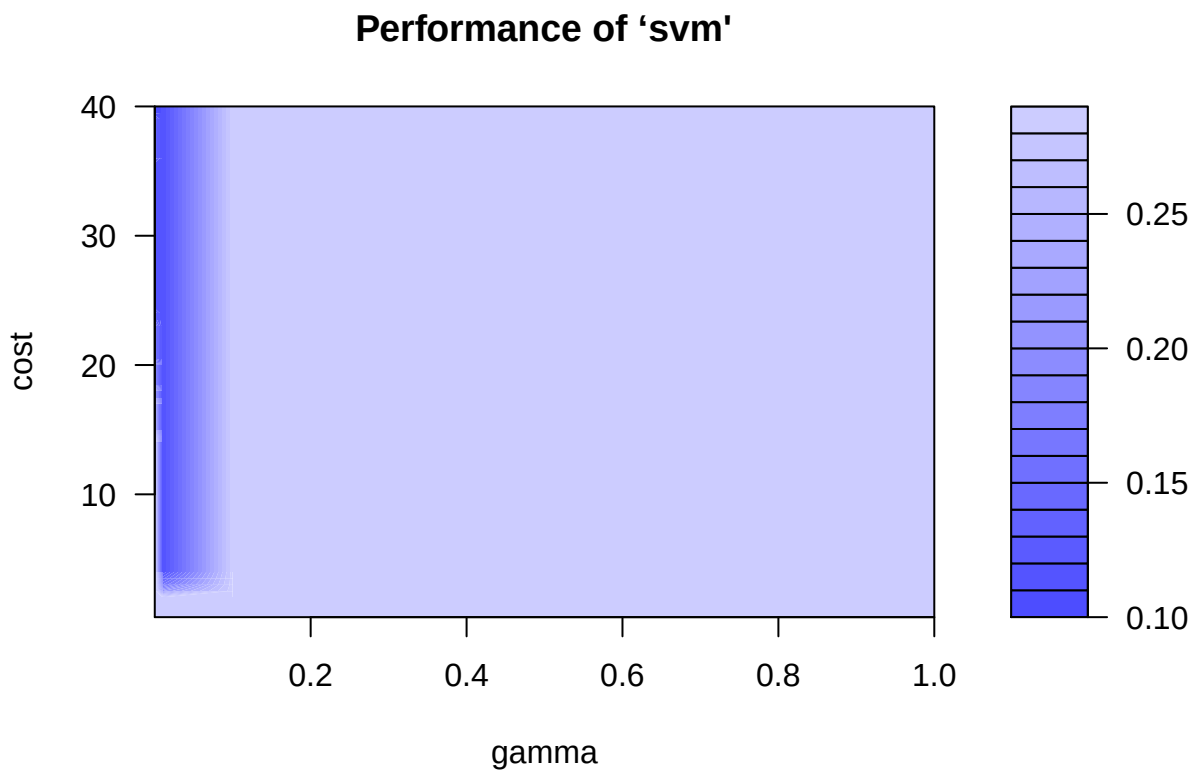
```
##             Sensitivity : 1.0000
##             Specificity : 0.5714
##          Pos Pred Value : 0.7692
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5882
##          Detection Rate : 0.5882
##    Detection Prevalence : 0.7647
##       Balanced Accuracy : 0.7857
##
##        'Positive' Class : ALL
##
```

RBF kernel

```
reduced.rbf.tune <- tune(svm,
                ALL.AML ~ .,
                data = train.reduced,
                kernel = "radial",
                ranges = list(gamma = 10^seq(-6, 0),
                              cost = seq(0.5, 40, by = 0.5)),
                tunecontrol = tune.control(cross = 5))
reduced.rbf.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  gamma cost
##  0.001 39.5
##
## - best performance: 0.1071429
```

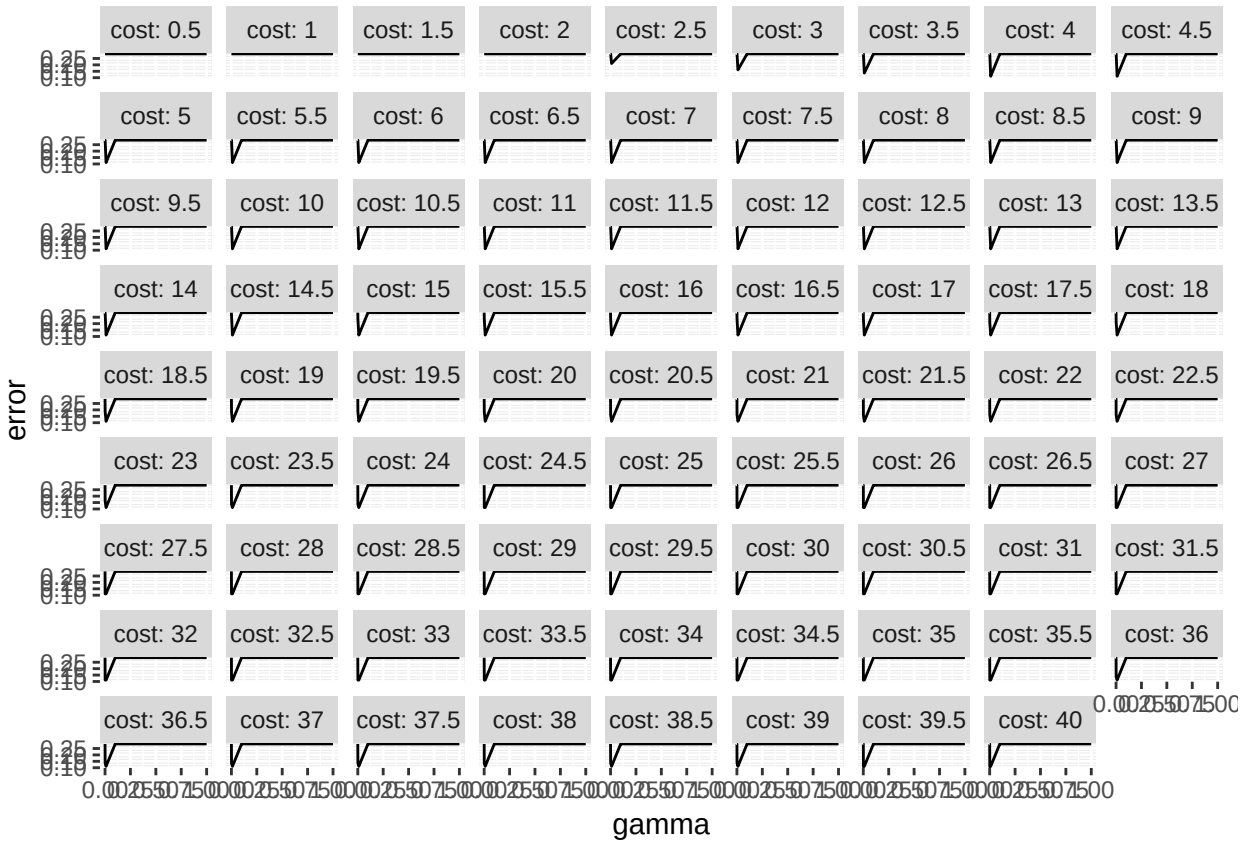
```
# rbf svm evaluation
plot(reduced.rbf.tune)
```




```
# best parameters
str(reduced.rbf.tune$performances)
```

```
## 'data.frame':    560 obs. of  4 variables:
##  $ gamma     : num  1e-06 1e-05 1e-04 1e-03 1e-02 1e-01 1e+00 1e-06 1e-05 1e-04 ...
##  $ cost      : num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 1 1 ...
##  $ error     : num  0.286 0.286 0.286 0.286 0.286 ...
##  $ dispersion: num  0.213 0.213 0.213 0.213 0.213 ...
```

```
# visualization
ggplot(reduced.rbf.tune$performances,
       mapping = aes(x = gamma, y = error)) +
  geom_line() +
  facet_wrap(~cost, labeller = label_both)
```




```
# best model
cat('Best Model:\n')
```

```
## Best Model:
```

```
reduced.rbf.tune$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = ALL.AML ~ ., data = train.reduced,
##     ranges = list(gamma = 10^seq(-6, 0), cost = seq(0.5, 40,
##         by = 0.5)), tunecontrol = tune.control(cross = 5), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  39.5
##       gamma:  0.001
##
## Number of Support Vectors:  26
```

```
# rbf sum prediction
pred.reduced.rbf.test <- predict(reduced.rbf.tune$best.model, newdata = test.reduced)
confusionMatrix(pred.reduced.rbf.test, test.reduced$ALL.AML)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction ALL AML
##        ALL  20   6
##        AML   0   8
##
##                Accuracy : 0.8235
##                  95% CI : (0.6547, 0.9324)
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : 0.003193
##
##                   Kappa : 0.6107
##  Mcnemar's Test P-Value : 0.041227
##
##             Sensitivity : 1.0000
##             Specificity : 0.5714
##          Pos Pred Value : 0.7692
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5882
##          Detection Rate : 0.5882
##    Detection Prevalence : 0.7647
##       Balanced Accuracy : 0.7857
##
##        'Positive' Class : ALL
##
```

The overall test accuracies for both kernels using the reduced data is 82.4%, while both kernels classify ALL with 100% acuracy, and AML with 57.1% accuracy. The accuracies are lower in the reduced set compared to the complete set in part (a), as only 90% of variance is captured in the reduced data set, the classification is therefore less accurate. This also suggests that the remaining 10% of variance captures important information especially of the AML class.