Stat 437 Project 1

Samantha Gregoryk (11559189)

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

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

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

## corrplot 0.84 loaded

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

Data set and its description

Please download the data set "TCGA-PANCAN-HiSeq-801x20531.tar.gz" from the website https://archive.ics.uci.edu/ml/machine-learning-databases/00401/. A brief description of the data set is given at https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq.

You need to decompress the data file since it is a .tar.gz file. Once uncompressed, the data files are "labels.csv" that contains the cancer type for each sample, and "data.csv" that contains the "gene expression profile" (i.e., expression measurements of a set of genes) for each sample. Here each sample is for a subject and is stored in a row of "data.csv". In fact, the data set contains the gene expression profiles for 801 subjects, each with a cancer type, where each gene expression profile contains the gene expressions for the same set of 20531 genes. The cancer types are: "BRCA", "KIRC", "COAD", "LUAD" and "PRAD". In both files "labels.csv" and "data.csv", each row name records which sample a label or observation is for.

Loading Data.

Task A. Clustering

For this task, you need to apply k-means and hierarchical clustering to cluster observations into their associated cancer types, and report your findings scientifically and professionally. Your laptop may not have sufficient computational power to implement k-means and hierarchical clustering on the whole data set, and genes whose expressions are zero for most of the subjects may not be so informative of a cancer type.

Please use `set.seed(123)` for random sampling via the command `sample`, random initialization of `kmeans`, implementing the gap statistic, and any other process where artificial randomization is needed.

(**Task A1**) Complete the following data processing steps:

- Filter out genes (from "data.csv") whose expressions are zero for at least 300 subjects, and save the filtered data as R object "gexp2".
- Use the command `sample` to randomly select 1000 genes and their expressions from "gexp2", and save the resulting data as R object "gexp3".
- Use the command `sample` to randomly select 30 samples and their labels from the file "labels.csv", and save them as R object "labels1". For these samples, select the corresponding samples from "gexp3" and save them as R object "gexpProj1".
- Use the command `scale` to standard the gene expressions for each gene in "gexpProj1", so that they have sample standard deviation 1. Save the standardized data as R object "stdgexpProj1".

```r
set.seed(123)

# filter expressions
gexp2 <- data[, (colSums(data == 0, na.rm = TRUE) < 300), drop = TRUE]

# sample 1000 genes
gexp3 <- sample(gexp2, 1000)
gexp3$X <- gexp2[, 1]

# sample 30 labels
labels1 <- labels[sample(nrow(labels), 30), ]

# select the corresponding samples from 'gexp3'
gexpProj1 <- merge(gexp3, labels1, by = "X")

gexpProj1 <- gexpProj1[, c(1, 1002, 2:1001)]

# standardize gene expression
gp <- gexpProj1[, -c(1, 2)]
stdgexpProj1 <- scale(gp)
```

```r
stdgexpProj1 <- as.data.frame(stdgexpProj1)
stdgexpProj1$X <- gexpProj1[, 1]
stdgexpProj1 <- stdgexpProj1[, c(1001, 2:1000)]
```

(**Task A2**)

(**Part 1 of Task A2**) Randomly pick 50 genes and their expressions from "stdgexpProj1", and do the following to these expressions: apply the "gap statistic" to estimate the number of clusters, apply K-means clustering with the estimated number of clusters given by the gap statistic, visualize the classification results using techniques given by "LectureNotes3_notes.pdf.pdf", and provide a summary on classification errors. You may use the command `table` and "labels1" to obtain classification errors. Note that the cluster numbering given by `kmeans` will usually be coded as follows:

```
#   Class  label
#     PRAD  5
#     LUAD  4
#     BRCA  1
#     KIRC  3
#     COAD  2
```

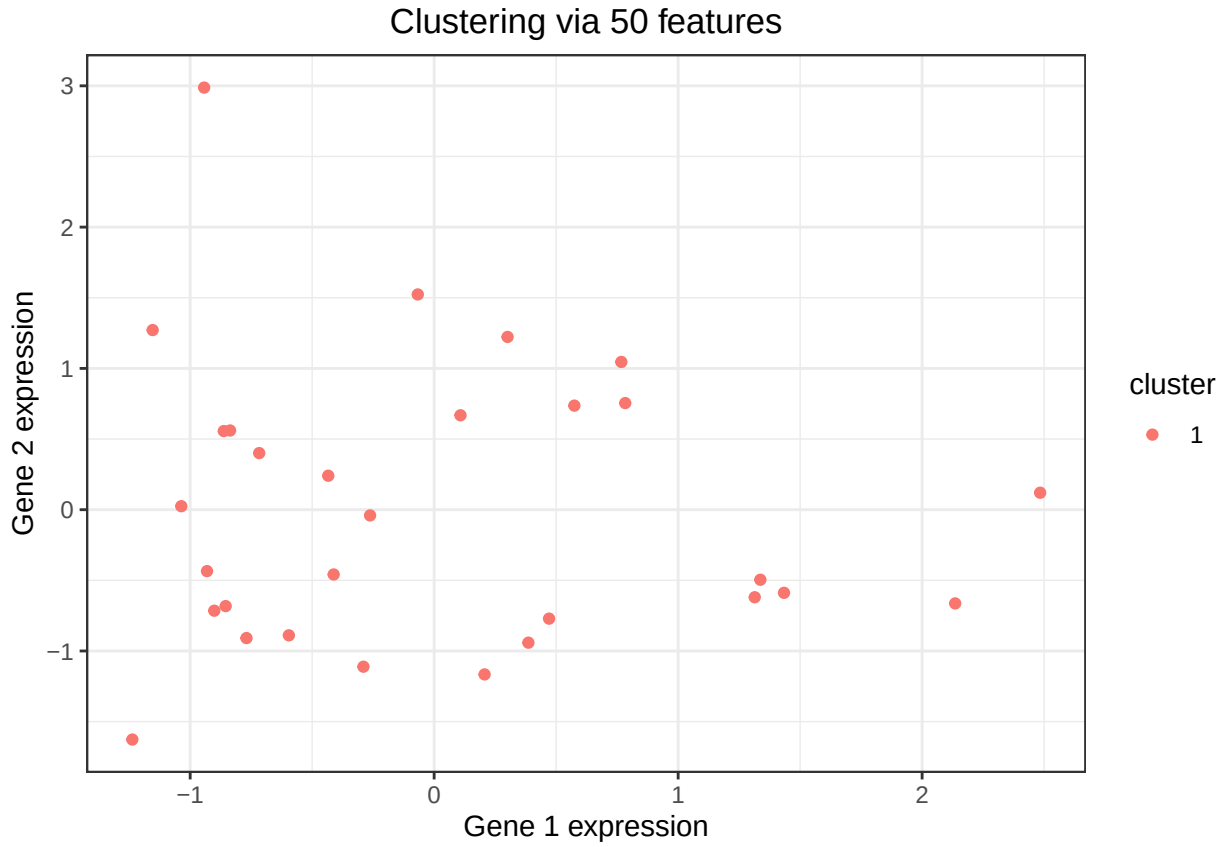
When you apply `clusGap`, please use arguments `K.max=10, B=200,iter.max=100`, and when you use `kmeans`, please use arguments `iter.max = 100, nstart=25, algorithm = c("Hartigan-Wong")`.

```r
set.seed(123)

# random 50 genes
rand.50 <- stdgexpProj1[, sample(ncol(stdgexpProj1), 50)]

# apply the 'gap statistic' to estimate the number of
# clusters
gap <- clusGap(rand.50, kmeans, K.max = 10, B = 200, iter.max = 100)

# apply K-means clustering with the estimated number of
# clusters given by the gap statistic
k <- maxSE(gap$Tab[, "gap"], gap$Tab[, "SE.sim"], method = "Tibs2001SEmax")
k
```

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

```r
# visualize the classification results using techniques given
# by 'LectureNotes3_notes.pdf.pdf'
mtx.50 <- as.matrix(rand.50)

km.out <- kmeans(mtx.50, 1, iter.max = 100, nstart = 25, algorithm = c("Hartigan-Wong"))
```

```
rand.50$cluster <- factor(km.out$cluster)
rand.50$X <- stdgexpProj1[, 1]

p1 <- ggplot(rand.50, aes(rand.50[, 1], rand.50[, 2])) + xlab("Gene 1 expression") +
    ylab("Gene 2 expression") + theme_bw() + geom_point(aes(color = cluster),
    na.rm = T) + theme(legend.position = "right") + ggtitle("Clustering via 50 features") +
    theme(plot.title = element_text(hjust = 0.5))
p1
```




```
table(labels1)
```

```
##             Class
## X            BRCA COAD KIRC LUAD PRAD
##   sample_179    0    0    1    0    0
##   sample_200    1    0    0    0    0
##   sample_206    1    0    0    0    0
##   sample_210    0    0    0    1    0
##   sample_24     0    0    1    0    0
##   sample_265    0    0    0    0    1
##   sample_290    0    0    1    0    0
##   sample_292    1    0    0    0    0
```

```
##    sample_303    0    0    0    1    0
##    sample_33     1    0    0    0    0
##    sample_342    1    0    0    0    0
##    sample_365    0    0    0    0    1
##    sample_380    0    0    1    0    0
##    sample_41     0    0    0    0    1
##    sample_424    0    0    0    1    0
##    sample_432    1    0    0    0    0
##    sample_538    0    0    1    0    0
##    sample_546    1    0    0    0    0
##    sample_605    1    0    0    0    0
##    sample_640    0    0    0    0    1
##    sample_668    1    0    0    0    0
##    sample_680    0    0    0    0    1
##    sample_693    0    1    0    0    0
##    sample_702    0    0    0    1    0
##    sample_712    1    0    0    0    0
##    sample_75     0    0    0    1    0
##    sample_777    0    0    1    0    0
##    sample_782    1    0    0    0    0
##    sample_85     0    0    0    0    1
##    sample_99     1    0    0    0    0
```

Classification errors for K means assumes that there is always k amount of clusters. It also assumes all clusters have the same sum squares (SSE) and have the same importance for all cluster groups. It should also be noted that the sample of data used should be tested over different types of classification methods in order to find the best fit for the data. In this case, each cluster class was assigned to one of the 50 samples.

(**Part 2 of of Task A2**) Upon implementing `kmeans` with k as the number of clusters, we will obtain the "total within-cluster sum of squares" $W(k)$ from the output `tot.withinss` of `kmeans`. If we try a sequence of $k = 1, 2, 3, ..., 10$, then we get $W(k)$ for each k between 1 and 10. Let us look at the difference $\Delta_k = W(k) - W(k+1)$ for k ranging from 1 to 9. The K^* for which

$$\{\Delta_k : k < K^*\} \gg \{\Delta_k : k \geq K^*\}$$

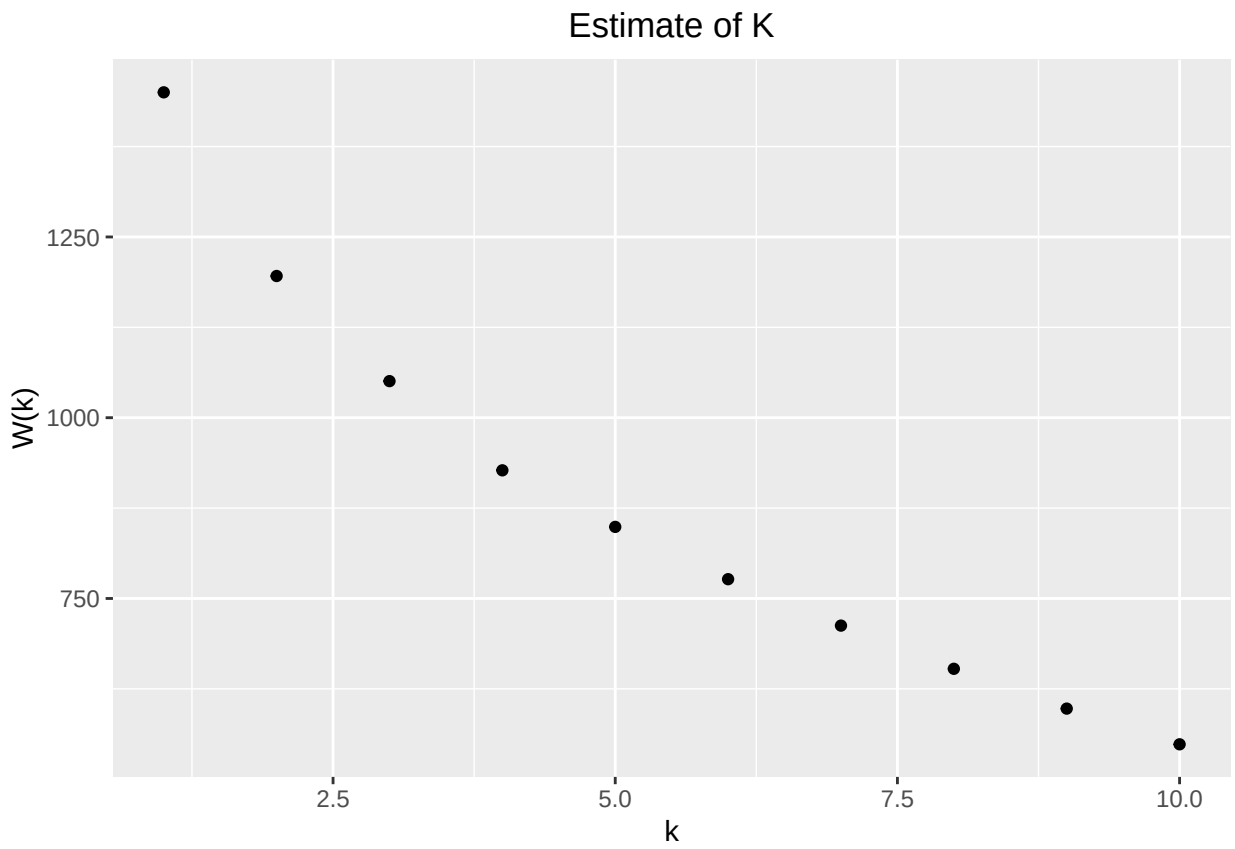
is an estimate of the true number K of clusters in the data, where $\gg$ means "much larger". Apply this method to obtain an estimate of K for the data you created in **Part 1**, and provide a plot of $W(k)$ against k for each k between 1 and 10. Compare this estimate with the estimate obtained in **Part 1** given by the gap statistic, comment on the accuracy of the two estimates, and explain why they are different.

```r
set.seed(123)

# Apply method to obtain an estimate of K for the data you
# created in Part 1
k.means <- function(df, x) {
    km <- kmeans(df, x, iter.max = 100, nstart = 25, algorithm = c("Hartigan-Wong"))
```

```
}

for (i in 1:10) {
    k <- k.means(mtx.50, i)
    print(k$tot.withinss)
}
```

```
## [1] 1450
## [1] 1227.117
## [1] 1046.286
## [1] 902.0388
## [1] 792.8026
## [1] 725.7802
## [1] 661.3111
## [1] 608.1006
## [1] 557.3356
## [1] 515.3775
```

```
est.df <- data.frame(K = c(1:10), Estimate1 = c(1450, 1195.97,
    1050.569, 927.2279, 848.9938, 776.6594, 712.4604, 652.7215,
    597.6664, 548.3005))

# Provide a plot of W(k) against k for each k between 1 and
# 10
p2 <- ggplot(est.df, aes(K, Estimate1)) + xlab("k") + ylab("W(k)") +
    geom_point() + ggtitle("Estimate of K") + theme(plot.title = element_text(hjust = 0.5))
p2
```




```
# Compare this estimate with the estimate obtained in Part 1
# given by the gap statistic (k = 1)
for (i in 1:10) {
    km.outk <- k.means(mtx.50, i)
    print(km.outk$tot.withinss)
}
```

```
## [1] 1450
## [1] 1227.117
## [1] 1046.286
## [1] 902.0388
## [1] 792.8026
## [1] 725.7802
## [1] 661.3111
## [1] 608.1006
## [1] 557.3356
## [1] 511.9412
```

```
est.df$Estimate2 <- c(1450, 1227.117, 1046.286, 902.0388, 792.8026,
    725.7802, 661.3111, 609.967, 557.3356, 515.7921)
est.df
```

```
##     K Estimate1 Estimate2
```

```
## 1    1 1450.0000 1450.0000
## 2    2 1195.9700 1227.1170
## 3    3 1050.5690 1046.2860
## 4    4  927.2279  902.0388
## 5    5  848.9938  792.8026
## 6    6  776.6594  725.7802
## 7    7  712.4604  661.3111
## 8    8  652.7215  609.9670
## 9    9  597.6664  557.3356
## 10 10  548.3005  515.7921
```

The estimate of K provided by the gap statistics estimates k based on differences between within-cluster sums of squares. The estimate of k provided by W(k) estimates k as the number of clusters obtained by the TOTAL within-cluster sum of squares.

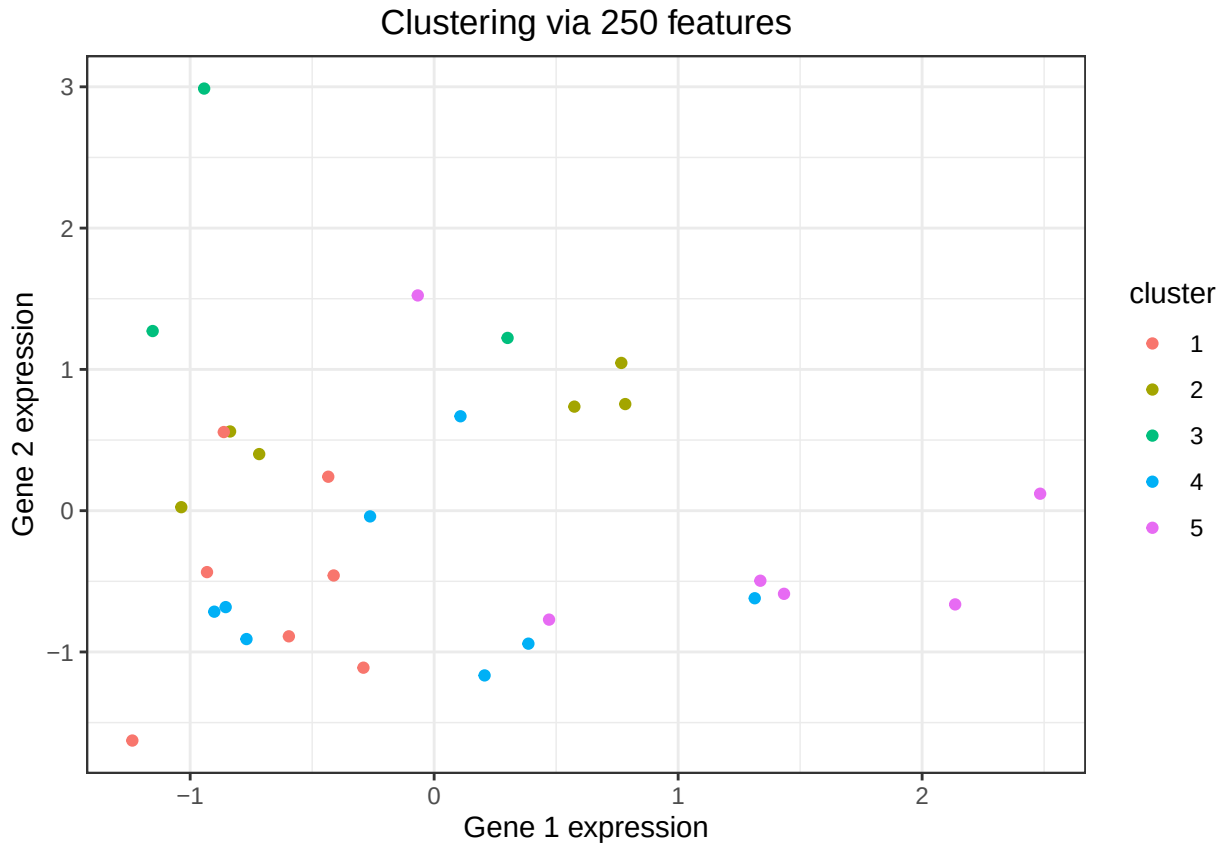
(**Part 3 of of Task A2**) Randomly pick 250 genes and their expressions from "stdgexpProj1", and for these expressions, do the analysis in **Part 1** and **Part 2**. Report your findings, compare your findings with those from **Part 1** and **Part 2**; if there are differences between these findings, explain why. Regard using more genes as using more features, does using more features necessarily give more accurate clutering or classification results?

```
set.seed(123)

# random 250 genes
rand.250 <- stdgexpProj1[, sample(ncol(stdgexpProj1), 250)]

# Part 1 analysis

# apply the 'gap statistic' to estimate the number of
# clusters
gap.1 <- clusGap(na.omit(rand.250), kmeans, K.max = 10, B = 250,
    iter.max = 100)

# apply K-means clustering with the estimated number of
# clusters given by the gap statistic
k.1 <- maxSE(gap.1$Tab[, "gap"], gap.1$Tab[, "SE.sim"], method = "Tibs2001SEmax")
k.1
```

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

```
# visualize the classification results using techniques given
# by 'LectureNotes3_notes.pdf.pdf'
mtx.250 <- as.matrix(rand.250)

km.out.1 <- kmeans(na.omit(mtx.250), 5, iter.max = 100, nstart = 25,
    algorithm = c("Hartigan-Wong"))
```

```
rand.250$cluster <- factor(km.out.1$cluster)
rand.250$X <- stdgexpProj1[, 1]

p.1 <- ggplot(rand.250, aes(rand.250[, 1], rand.250[, 2])) +
    xlab("Gene 1 expression") + ylab("Gene 2 expression") + theme_bw() +
    geom_point(aes(color = cluster), na.rm = T) + theme(legend.position = "right") +
    ggtitle("Clustering via 250 features") + theme(plot.title = element_text(hjust = 0.5))
p.1
```

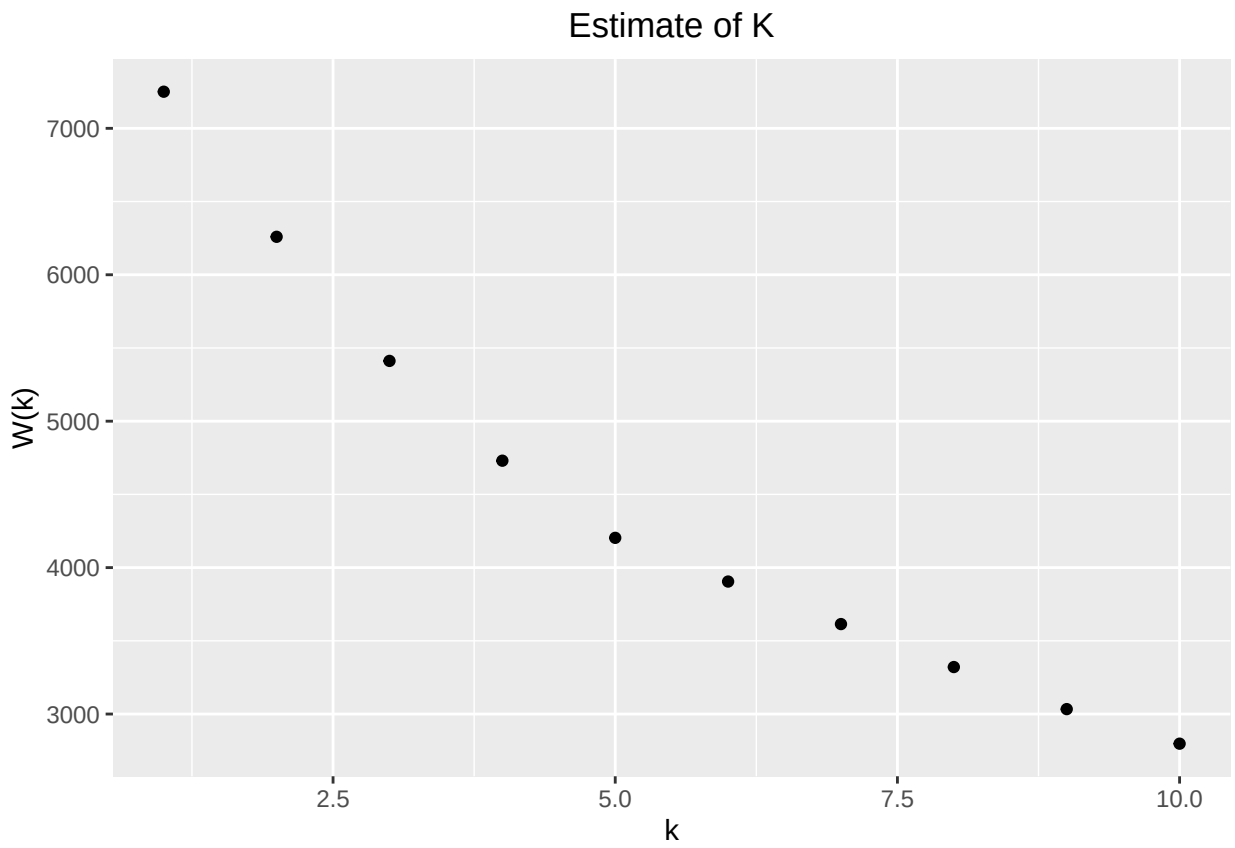



```
set.seed(123)

# Part 2 analysis

# Apply method to obtain an estimate of K for the data you
# created in Part 1
for (i in 1:10) {
    k <- k.means(mtx.250, i)
    print(k$tot.withinss)
}
```

```
## [1] 7250
## [1] 6134.221
```

```
## [1] 5342.334
## [1] 4549.335
## [1] 4081.71
## [1] 3752.054
## [1] 3443.701
## [1] 3143.002
## [1] 2887.669
## [1] 2695.234
```

```
est.df.2 <- data.frame(K = c(1:10), Estimate1 = c(7250, 6259.198,
    5411.601, 4730.2, 4203.374, 3904.876, 3614.462, 3321.071,
    3033.697, 2797.635))

# Provide a plot of W(k) against k for each k between 1 and
# 10
p.2 <- ggplot(est.df.2, aes(K, Estimate1)) + xlab("k") + ylab("W(k)") +
    geom_point() + ggtitle("Estimate of K") + theme(plot.title = element_text(hjust = 0.5))
p.2
```




```
# Compare this estimate with the estimate obtained in Part 1
# given by the gap statistic
for (i in 1:10) {
```

```
    km.outk <- k.means(mtx.250, i)
    print(km.outk$tot.withinss)
}
```

```
## [1] 7250
## [1] 6134.221
## [1] 5342.334
## [1] 4549.335
## [1] 4081.71
## [1] 3752.054
## [1] 3443.701
## [1] 3143.002
## [1] 2887.669
## [1] 2670.278
```

```
est.df.2$Estimate2 <- c(7250, 6134.221, 5342.334, 4549.335, 4081.71,
    3752.054, 3443.701, 3143.002, 2887.669, 2670.278)
est.df.2
```

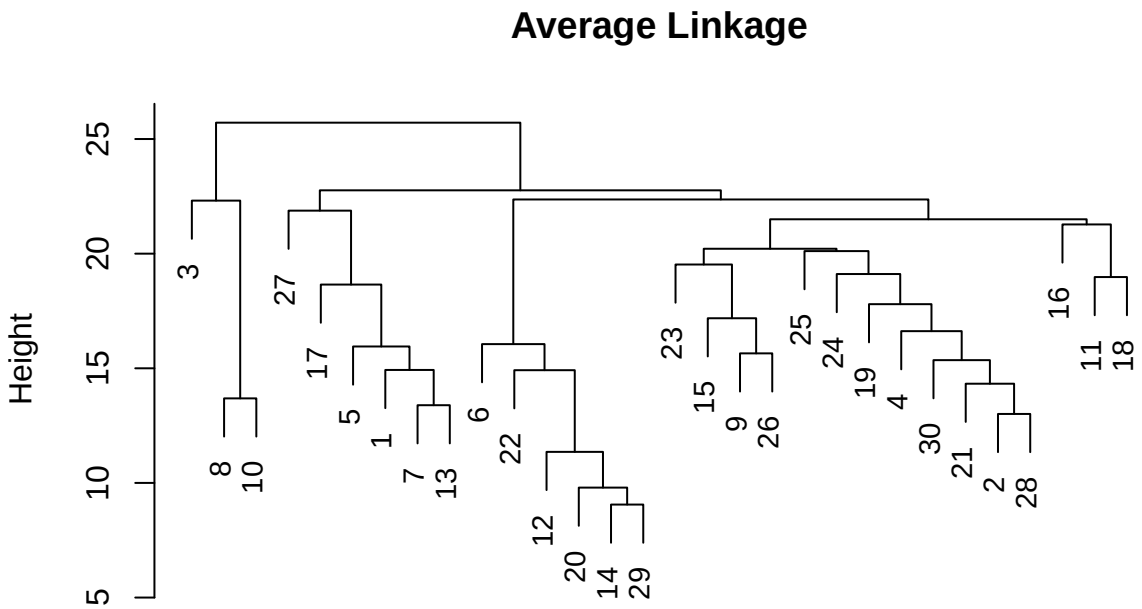
```
##     K Estimate1 Estimate2
## 1   1  7250.000  7250.000
## 2   2  6259.198  6134.221
## 3   3  5411.601  5342.334
## 4   4  4730.200  4549.335
## 5   5  4203.374  4081.710
## 6   6  3904.876  3752.054
## 7   7  3614.462  3443.701
## 8   8  3321.071  3143.002
## 9   9  3033.697  2887.669
## 10 10  2797.635  2670.278
```

Just like the 50 sample matrix, the estimate of K provided by the gap statistics estimates k based on differences between within-cluster sums of squares while the estimate of k provided by W(k) estimates k as the number of clusters obtained by the TOTAL within-cluster sum of squares. Therefore, each k estimate will differ except for when k = 1 because it is only one cluster.

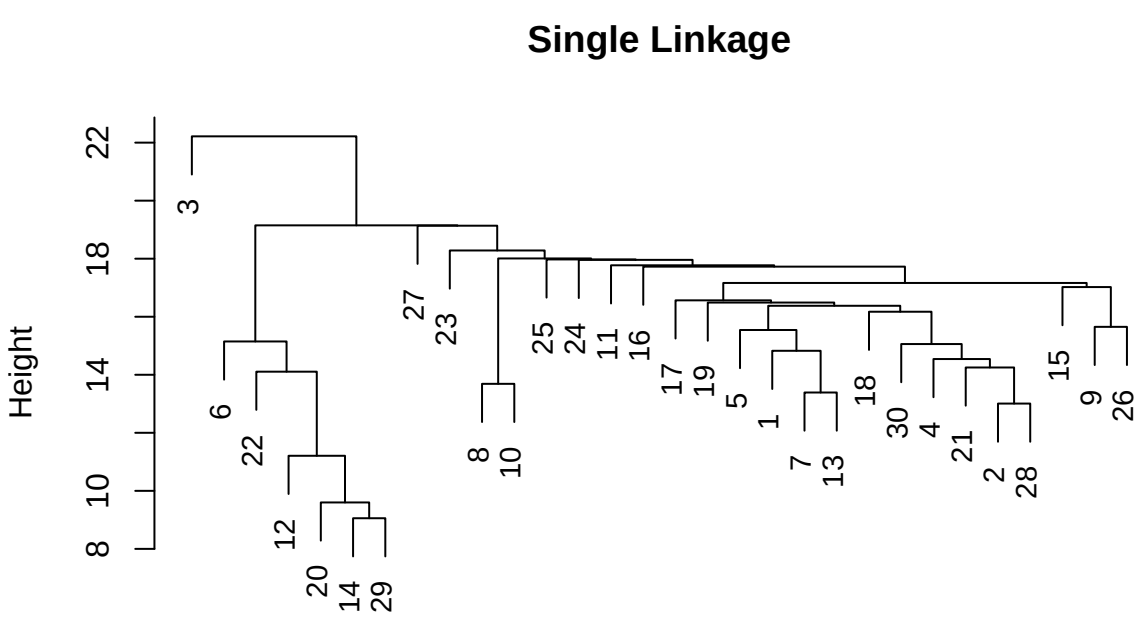
(**Task A3**) Randomly pick 250 genes and their expressions from "stdgexpProj1", and for these expressions, do the following: respectively apply hierarchical clustering with average linkage, single linkage, and complete linkage to cluster subjects into groups, and create a dendrogram. For the dendrogram obtained from average linkage, find the height at which cutting the dendrogram gives the same number of groups in "labels1", and comment on the clustering results obtained at this height by comparing them to the truth contained in "labels1".

```
set.seed(123)

# Randomly pick 250 genes and their expressions from
```

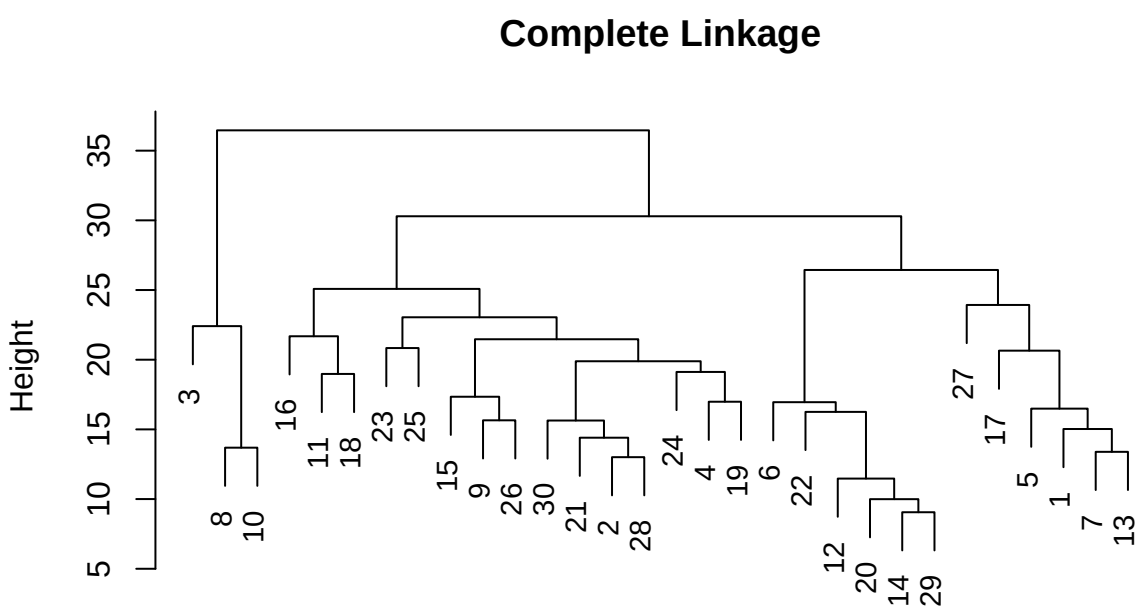
```r
# 'stdgexpProj1'
pick.250 <- stdgexpProj1[, sample(ncol(stdgexpProj1), 250)]

# Respectively apply hierarchical clustering to cluster
# subjects into groups

## average linkage
hc.average <- hclust(dist(pick.250), method = "average")
plot(hc.average, main = "Average Linkage", xlab = "", sub = "",
    cex = 0.9)
```



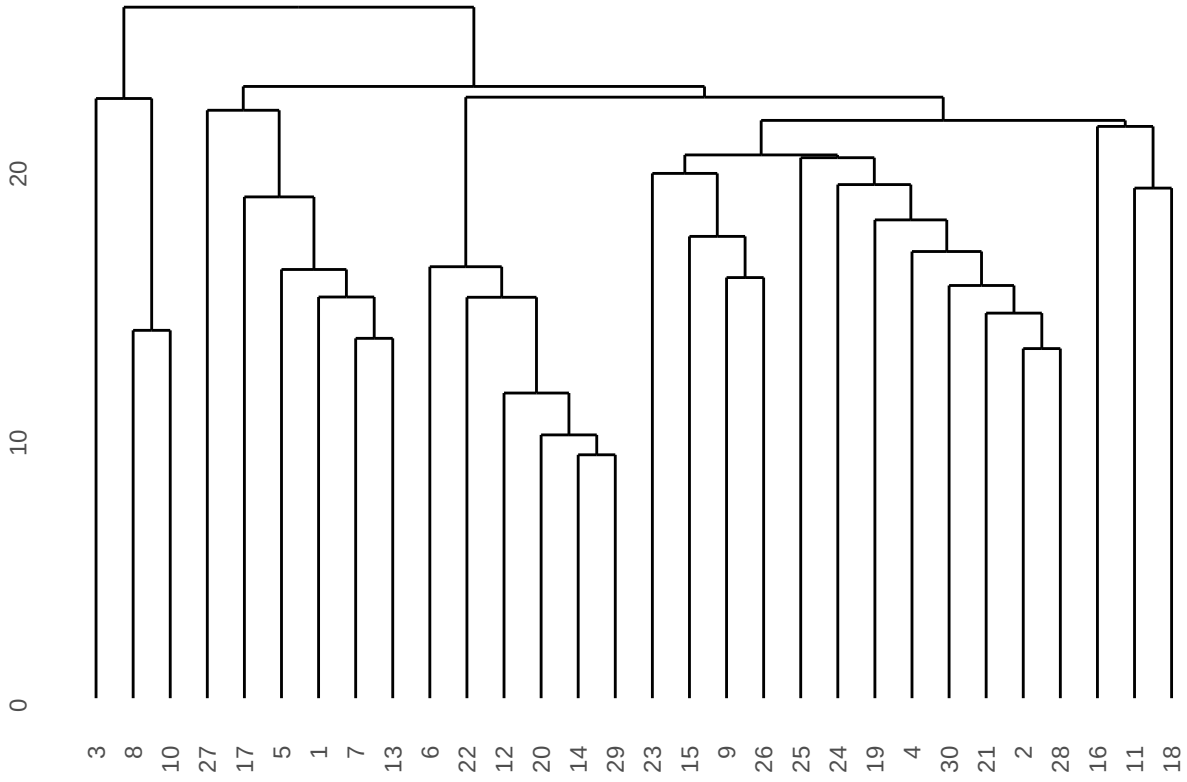

```r
## single linkage
hc.single <- hclust(dist(pick.250), method = "single")
plot(hc.single, main = "Single Linkage", xlab = "", sub = "",
    cex = 0.9)
```



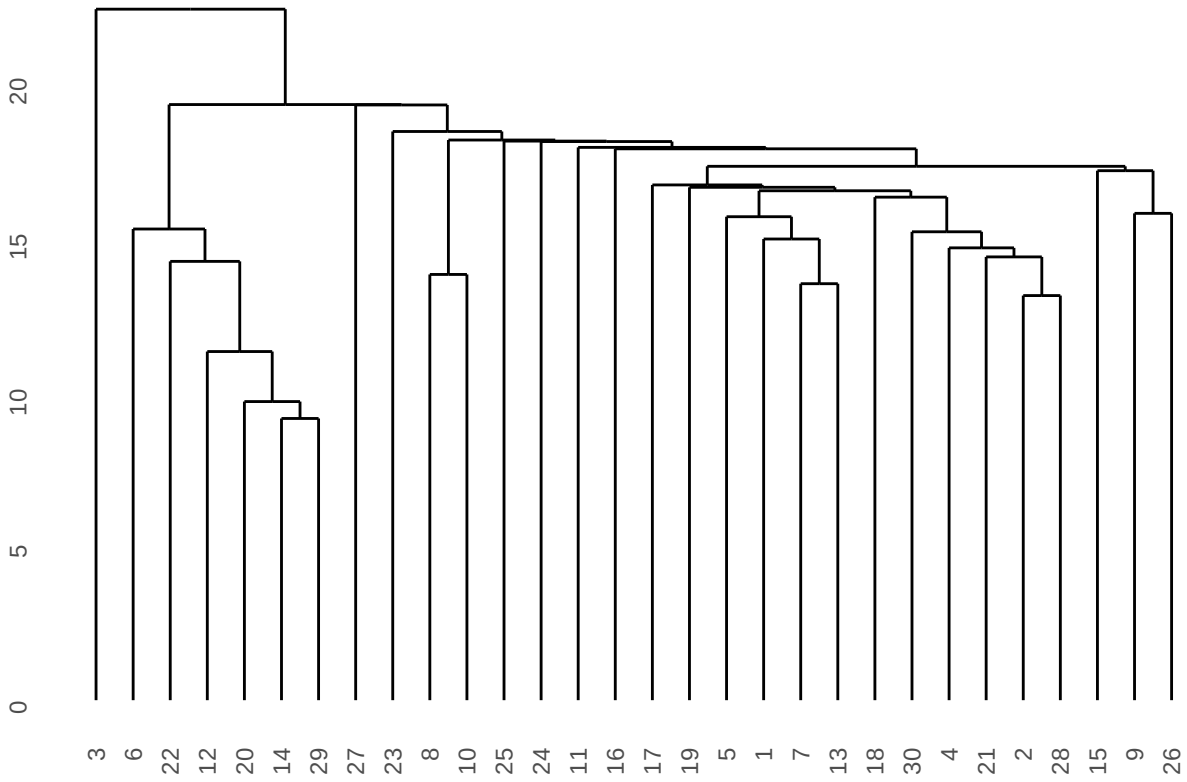

```
## complete linkage
hc.complete <- hclust(dist(pick.250), method = "complete")
plot(hc.complete, main = "Complete Linkage", xlab = "", sub = "",
    cex = 0.9)
```



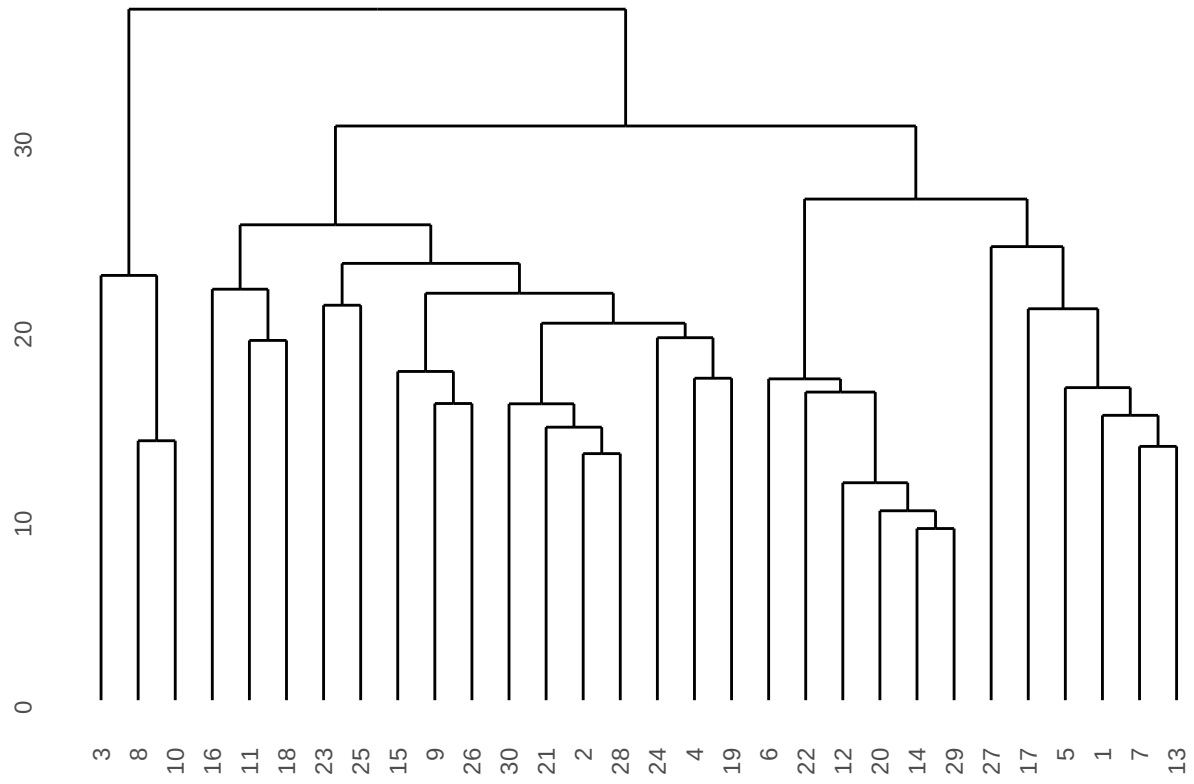

```
# Create a dendrogram for each linkage

## average linkage
ggdendrogram(hc.average, leaf_labels = TRUE, rotate = FALSE)
```




```
## single linkage
ggdendrogram(hc.single, leaf_labels = TRUE, rotate = FALSE)
```




```
## complete linkage
ggdendrogram(hc.complete, leaf_labels = TRUE, rotate = FALSE)
```




```
# For the dendrogram obtained from average linkage, find the
# height at which cutting the dendrogram gives the same
# number of groups in 'labels1'

## average linkage dendrogram
hc.average$height
```

```
##  [1]  9.056425  9.796263 11.353939 13.007835 13.389332 13.687368 14.327630
##  [8] 14.917214 14.927270 15.355431 15.651064 15.950890 16.054617 16.619755
## [15] 17.182378 17.796659 18.651967 18.979536 19.110654 19.526194 20.111020
## [22] 20.213734 21.274592 21.499039 21.876014 22.313135 22.363078 22.761849
## [29] 25.712501
```

```
# height with 5 clusters
ch.average <- hc.average$height[length(hc.average$height) - 4]
ch.average
```

```
## [1] 21.87601
```

```
# cutting dendrogram
ct.a.2 <- cutree(hc.average, h = hc.average$height[(length(hc.average$height)) -
    4])

# comparing groups to find the right height
table(labels1$Class)
```

```
##
## BRCA COAD KIRC LUAD PRAD
##   12    1    6    5    6
```

```
table(ct.a.2)
```

```
## ct.a.2
##  1  2  3  4  5
##  6 15  1  6  2
```

The results from the average linkage dendrogram height does not match with the truth contained in "labels1". Each cluster has a different result.

Task B. Classification

For this task, we will use the same data set you would have downloaded. Please use `set.seed(123)` for random sampling via the command `sample` and any other process where artificial randomization is needed.

(**Task B1**) After you obtain "labels.csv" and "data.csv", do the following:

- Filter out genes (from "data.csv") whose expressions are zero for at least 300 subjects, and save the filtered data as R object "gexp2".
- Use the command `sample` to randomly select 1000 genes and their expressions from "gexp2", and save the resulting data as R object "gexp3".
- Pick the samples from "labels.csv" that are for cancer type "LUAD" or "BRCA", and save them as object "labels2". For these samples, pick the corresponding gene expressions from "gexp3" and save them as object "stdgexp2"

```
set.seed(123)

# pick the samples from 'labels.csv' that are for cancer type
# 'LUAD' or 'BRCA'
labels2 <- filter(labels, labels$Class == "LUAD" | labels$Class ==
    "BRCA")
```

```
gexp3$X <- labels$X
gexp3 <- gexp3[, c(1001, 1:1000)]

# for these samples, pick the corresponding gene expressions
# from 'gexp3'
stdgexp2 <- merge(labels2, gexp3, by = "X")
```

(**Taks B2**) The assumptions of linear or quadratic discriminant analysis requires that each observation follows a Gaussian distribution given the class or group membership of the observation, and that each observation follows a Gaussian mixture model. In our settings here, each observation (as a row) within a group would follow a Gaussian with dimensionality equal to the number of genes (i.e., number of entries of the row). So, the more genes whose expressions we use for classification, the higher the dimension of these Gaussian distributions. Nonetheless, you need to check if the Gaussian mixture assumption is satisfied. Note that we only consider two classes "LUAD" and "BRCA", for which the corresponding Gaussian mixture has 2 components and hence has 2 bumps when its density is plotted.

Do the following and report your findings on classification:

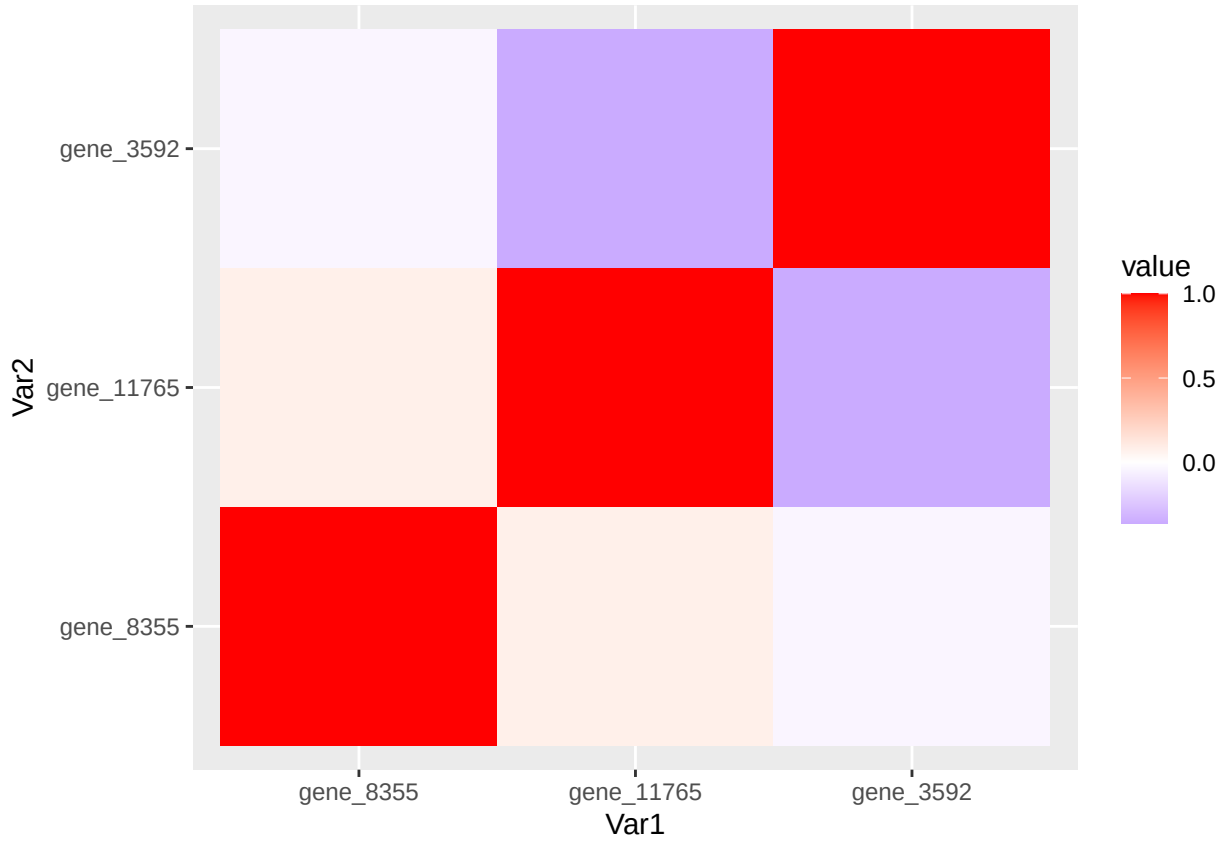
- Randomly pick 3 genes and their expressions from "stdgexp2", and save them as object "stdgexp2a".
- Randomly pick 60% of samples from "stdgexp2a", use them as the training set, and use the rest as the test set. You can round down the number of samples in the training set by the command `floor` if it is not an integer.

Build a quadratic discriminant analysis model using the training set, and apply the obtained model to the test set to classify each of its observations. You should code "BRCA" as 0 and "LUAD" as 1. If for an observation the posterior probability of being "BRCA" is predicted by the model to be greater than 0.5, the observation is classified as "BRCA". Report via a 2-by-2 table on the classification errors. Note that the predicted posterior probability given by `qda` is for an observation to belong to class "BRCA".

Before building a quadratic discriminant analysis model, you need to check for highly correlated gene expressions, i.e., you need to check the sample correlations between each pair of columns of the training set. If there are highly correlated gene expressions, the estimated covariance matrix can be close to to being singular, leading to unstable inference. You can remove a column from two columns when their contained expressions have sample correlation greater than 0.9 in absolute value.

```
# Code from Professor Chen
set.seed(123)

stdgexp2a = stdgexp2[, sample(1:ncol(stdgexp2), 3)]
n = nrow(stdgexp2a)

# obtain subsets of data
trainid1 = sample(1:n, floor(0.6 * n))
```

```
testid1 = (1:n)[-trainid1]

trainOB1 = stdgexp2a[trainid1, ]
testOB1 = stdgexp2a[testid1, ]

# relabel: note 1=BRCA, 4=LUAD for
labels2$Code[labels2$Class == "LUAD"] <- 1
labels2$Code[labels2$Class == "BRCA"] <- 4
labels2a = as.numeric(labels2$Code)
labels2a[labels2a == 1] = 0
labels2a[labels2a == 4] = 1
trainLb1 = labels2a[trainid1]
testLb1 = labels2a[testid1]

# check correlations and visualize them via a heatmap
corMat1 = cor(trainOB1)
melted_cormat1 <- melt(corMat1)
ggplot(data = melted_cormat1, aes(x = Var1, y = Var2, fill = value)) +
    geom_tile() + scale_fill_gradient2(low = "blue", high = "red")
```



```
# check on highly correlated variables
corMat11 = corMat1

# set diagonal entries of corMat1 to be 0 since they are 1
diag(corMat11) = 0
hcIdx1 = which(abs(corMat11) > 0.9, arr.ind = TRUE)
hcIdx1
```

```
##      row col
```

```
# remove these highly correlated variables

# extract X and Y from test and training data
testX1 = as.matrix(testOB1)
testY1 = as.numeric(testLb1)
trainX1 = as.matrix(trainOB1)
trainY1 = as.numeric(trainLb1)

# combine labels and expressions
train1 = as.data.frame(cbind(trainLb1, trainOB1))
```

```
set.seed(123)

# Build a quadratic discriminant analysis model using the
# training set
qda.fit1 <- qda(trainLb1 ~ ., data = train1)
pred1 <- predict(qda.fit1)$class

mtx1 <- as.matrix(pred1)

# apply the obtained model to the test set to classify each
# of its observations

# Report via a 2-by-2 table on the classification errors.
table(mtx1 > 0.5, mtx1 < 0.5, dnn = c("QDAEstimtedClassLabel",
    "TrueClassLabel"))
```

```
##                       TrueClassLabel
## QDAEstimtedClassLabel FALSE TRUE
##                 FALSE     0    6
##                 TRUE    258    0
```

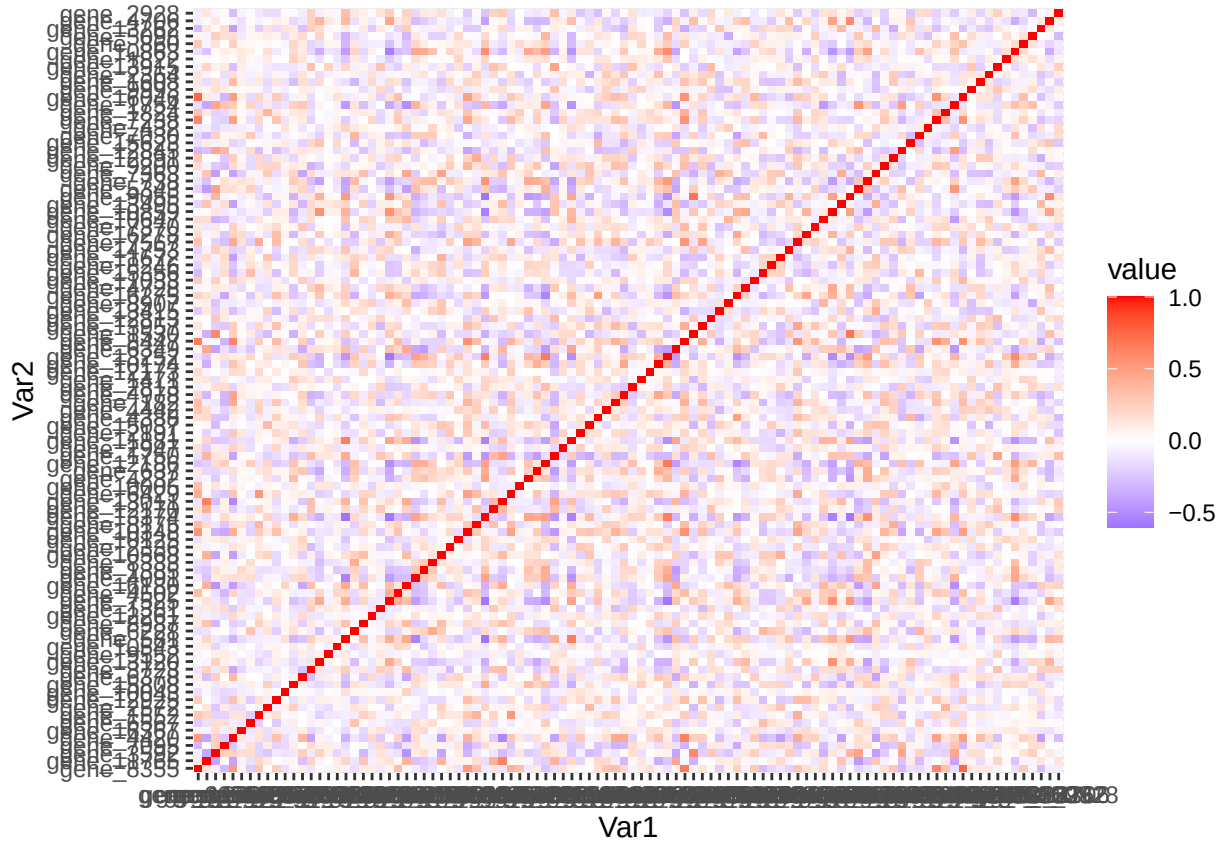
(**Taks B3**) Do the following:

- Randomly pick 100 genes and their expressions from "stdgexp2", and save them as object "stdgexp2b".

- Randomly pick 75% of samples from "stdgexp2b", use them as the training set, and use the rest as the test set. You can round down the number of samples in the training set by the command `floor` if it is not an integer.

Then apply quadratic discriminant analysis by following the requirements given in **Taks B2**. Compare classification results you find here with those found in **Taks B2**, and explain on any difference you find between the classification results.

```
# Code from Professor Chen

set.seed(123)

stdgexp2b = stdgexp2[, sample(1:ncol(stdgexp2), 100)]
n1 = nrow(stdgexp2b)

# obtain subsets of data
trainid = sample(1:n1, floor(0.75 * n1))
testid = (1:n1)[-trainid]

trainOB = stdgexp2b[trainid, ]
testOB = stdgexp2b[testid, ]

# relabel: note 1=BRCA, 4=LUAD for
labels2$Code[labels2$Class == "LUAD"] <- 1
labels2$Code[labels2$Class == "BRCA"] <- 4
labels2a = as.numeric(labels2$Code)
labels2a[labels2a == 1] = 0
labels2a[labels2a == 4] = 1
trainLb = labels2a[trainid]
testLb = labels2a[testid]

# check correlations and visualize them via a heatmap
corMat = cor(trainOB)
melted_cormat <- melt(corMat)
ggplot(data = melted_cormat, aes(x = Var1, y = Var2, fill = value)) +
    geom_tile() + scale_fill_gradient2(low = "blue", high = "red")
```




```
# check on highly correlated variables
corMat1 = corMat

# set diagonal entries of corMat1 to be 0 since they are 1
diag(corMat1) = 0
hcIdx = which(abs(corMat1) > 0.9, arr.ind = TRUE)
hcIdx
```

```
##      row col
```

```
# remove these highly correlated variables

# extract X and Y from test and training data
testX = as.matrix(testOB)
testY = as.numeric(testLb)
trainX = as.matrix(trainOB)
trainY = as.numeric(trainLb)

# combine labels and expressions
train2 = as.data.frame(cbind(trainLb, trainOB))
test2 = as.data.frame(cbind(testLb, testOB))
```

```r
set.seed(123)

# apply quadratic discriminant analysis by following the
# requirements given in **Taks B2**.

# Build a quadratic discriminant analysis model using the
# training set
qda.fit2 <- qda(trainLb ~ ., data = train2)
pred2 <- predict(qda.fit2)$class

mtx2 <- as.matrix(pred2)

# apply the obtained model to the test set to classify each
# of its observations

# Report via a 2-by-2 table on the classification errors.
table(mtx2 < 0.5, mtx2 > 0.5, dnn = c("QDAEstimtedClassLabel",
    "TrueClassLabel"))
```

```
##                       TrueClassLabel
## QDAEstimtedClassLabel FALSE TRUE
##                 FALSE     0  228
##                 TRUE    102    0
```

```r
# Compare classification results you find here with those
# found in **Taks B2**
table(mtx1 < 0.5, mtx1 > 0.5, dnn = c("QDAEstimtedClassLabel",
    "TrueClassLabel"))
```

```
##                       TrueClassLabel
## QDAEstimtedClassLabel FALSE TRUE
##                 FALSE     0  258
##                 TRUE      6    0
```

With a larger sample size of 100 gene expressions, there are less false negatives than with a sample size of 3 gene expressions. False positives are relatively the same with the sample size of 100 gene expressions having more than the sample size of 3 gene expressions.

(**Taks B4**) Do the following:

- Randomly pick 100 genes and their expressions from "stdgexp2", and save them as object "stdgexp2b".
- Randomly pick 75% of samples from "stdgexp2b", use them as the training set, and use the rest as the test set. You can round down the number of samples in the training set by the command `floor` if it is not an integer.

Then apply k-nearest-neighbor (k-NN) method with neighborhood size k=3 to the test data to classify each observation in the test set into one of the cancer types. Here, for an observation, if the average of being cancer type "BRCA" is predicted by k-NN to be greater than 0.5, then the observation is classified as being "BRCA". Report via a 2-by-2 table on the classification errors. Compare and comment on the classification results obtained here to those obtain in **Taks B3**. If there is any difference between the classification results, explain why.

```
set.seed(123)

# apply k-nearest-neighbor (k-NN) method with neighborhood
# size k=3
k.nn <- knn(train2, test2, cl = trainLb, k = 3)
kN <- as.matrix(k.nn)

# Here, for an observation, if the average of being cancer
# type 'BRCA' is predicted by k-NN to be greater than 0.5,
# then the observation is classified as being 'BRCA'.

# Report via a 2-by-2 table on the classification errors.
table(kN < 0.5, kN > 0.5, dnn = c("kNNEstimtedClassLabel", "TrueClassLabel"))
```

```
##                       TrueClassLabel
## kNNEstimtedClassLabel FALSE TRUE
##                 FALSE     0   79
##                 TRUE     32    0
```

```
# B3
table(mtx2 < 0.5, mtx2 > 0.5, dnn = c("QDAEstimtedClassLabel",
    "TrueClassLabel"))
```

```
##                       TrueClassLabel
## QDAEstimtedClassLabel FALSE TRUE
##                 FALSE     0  228
##                 TRUE    102    0
```

With the same sample size of 100 gene expressions and different classification methods, the results differ. There are both less false negatives and false positives than from the kNN method compared to the quadratic analysis by a significant amount.