PROBLEM-1

January 17, 2020

Introduction of the data:-

A gene is a stretch of DNA inside the cell that tells the cell how to make a specific protein. All cells in the body contain the same genes, but the genes have different expression levels in different cell types, and cells can regulate gene expression levels in response to their environment. Many diseases, including cancer, fundamentally involve breakdowns in the regulation of gene expression. The expression profile of cancer cells becomes abnormal, and different kinds of cancers have different expression profiles. Our data are gene expression measurements from cells drawn from 64 different tumors (from 64 different patients). In each case, a device called a microarray (or gene chip) measured the expression of each of 6830 distinct genes, essentially the logarithm of the chemical concentration of the gene's product. Thus, each record in the data set is a vector of length 6830. The cells mostly come from known cancer types, so there are classes, in addition to the measurements of the expression levels. The classes are breast, cns(central nervous system), colon, leukemia, melanoma, nsclc(non-small cell lung cancer), ovarian, prostate, renal, K562A, K562B, MCF7A, MCF7D (those three are laboratory tumor cultures) and unknown.

The dataset is present in the library ElemStatLearn as nci.

ANALYSIS:-

I first install the library ElemStatLearn from CRAN and load the dataset as follows:-

```
library(ElemStatLearn)

Warning:  package 'ElemStatLearn' was built under R version 3.5.3

data(nci)
data = t(nci)
```

The matrix needs to be transposed because it gives the genes as the rows and cells as the columns.

We use the K-means algorithm to cluster the cells.First let us state what is the algorithm.

K-Means Algorithm:-

K-means is one of the most popular partitioning methods of clustering. It performs a non-hierarchical clustering and groups the observations of a dataset into a given number of clusters with the view to minimize the within-cluster scatter/variation due to the clustering process.

Suppose we have n observations $x_1, x_2, ..., x_n \epsilon \mathbb{R}^p$ and a measure $d_{ij} = d(x_i, x_j)$ of dissimilarity between x_i and x_j, for every $i, j = 1(1)n$. We may wish to group the n observations into K clusters. A clustering of the above n observations can be thought of as a function C which assigns the cluster number for an observation, i.e, for an observation x_i, $C(x_i) = k \epsilon \{1, 2, ..., K\}$ denotes the cluster k to which x_i is assigned during the clustering process. We may, instead, simply denote by $C(i)$, the cluster to which x_i is assigned.

Now, the within cluster scatter due to the above clustering is given by

$$W = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k,} \sum_{j:C(j)=k} d_{ij}$$

where n_k is the number of observations in thek^{th} cluster, $k = 1(1)K$.

In particular, we may take $d_{ij} = \|x_i - x_j\|^2$, the squared Euclidean Distance between the observations x_i and x_j. Then, we have

$$W = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k} \sum_{,j:C(j)=k} \|x_i - x_j\|^2$$

$$= \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k} \|x_i - \bar{x_k}\|^2$$

where

$$\bar{x_k} = \frac{1}{n_k} \sum_{i:C(i)=k} x_i$$

is the mean of the observations in the k^{th} cluster.

This W is the within-cluster variation due to the above clustering. Equivalently, we may choose to minimize

$$W = \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k} \|x_i - c_k\|^2$$

with respect to the clustering method C and $c_1, c_2, ..., c_K$.

K-means attempts to find the clustering method C so as to approximately minimize this within cluster variation. It runs in the following way:

- We start with an initial guess for $c_1, c_2, ..., c_K$ (e.g., pick K points at random over the range of $x_1, ..., x_K$), then:

1. Minimize over C: for each $i = 1(1)n$, find the cluster center c_k closest to x_i , and let $C(i) = k$

2. Minimize over $c_1, ..., c_K$: for each $k = 1(1)K$, let $c_k = \bar{x_k}$.

We repeat this process and stop when W does not change any more.

Thus, one gets hold of a clustering method C to cluster the above n observations.

a)We use the K-means algorithm to cluster the cells with k=14 to match the number of true classes.

```
cluster = kmeans(data,centers=14)
cluster$cluster

         CNS          CNS          CNS        RENAL       BREAST          CNS
           4            4            4           12           12           12
         CNS       BREAST        NSCLC        NSCLC        RENAL        RENAL
          12           12            4            4            1            1
       RENAL        RENAL        RENAL        RENAL        RENAL       BREAST
           1            1            1            1            1           10
       NSCLC        RENAL      UNKNOWN      OVARIAN     MELANOMA     PROSTATE
          10           10           14           14            4            5
     OVARIAN      OVARIAN      OVARIAN      OVARIAN      OVARIAN     PROSTATE
           5            5            5            5            5            5
       NSCLC        NSCLC        NSCLC     LEUKEMIA K562B-repro K562A-repro
           5            5            5           11            7            7
    LEUKEMIA     LEUKEMIA     LEUKEMIA     LEUKEMIA     LEUKEMIA        COLON
           7           11            2            2           11            5
       COLON        COLON        COLON        COLON        COLON        COLON
           6            9            9            9            6            9
 MCF7A-repro       BREAST  MCF7D-repro       BREAST        NSCLC        NSCLC
          13           13           13           13            9            5
       NSCLC     MELANOMA       BREAST       BREAST     MELANOMA     MELANOMA
           5            3            3            3            8            8
    MELANOMA     MELANOMA     MELANOMA     MELANOMA
           8            8            8            8
```

i)The K-means algorithm makes a lumping error whenever it assigns two cells of different classes to the same cluster, and a splitting error when it puts two cells of the same class in different clusters. Each pair of cells can give an error.

A)So here I will write a function which takes as inputs a vector of classes and a vector of clusters, and gives as output the number of lumping errors.Then I have to test your function by verifying that when the classes are (1, 2, 2), the three clusterings (1, 2, 2), (2,1, 1) and (4, 5, 6) all give zero lumping errors, but the clustering (1, 1, 1) gives two lumping errors.

```
# Count number of errors made by lumping different classes into
# one cluster
```

```r
# Uses explicit, nested iteration
# Input: vector of true classes, vector of guessed clusters
# Output: number of distinct pairs belonging to different classes
# lumped into one cluster
numlumperror <- function(true.class,guessed.cluster) {
n = length(true.class)
# Checking if the two vectors have the same length!
stopifnot(n == length(guessed.cluster))
# Initialize the count variable
num.lumping.errors = 0
# checking all distinct pairs
for (i in 1:(n-1)) {
# think of above-diagonal entries in a matrix
for (j in (i+1):n) {
if ((true.class[i] != true.class[j])
& (guessed.cluster[i] == guessed.cluster[j])) {
# lumping error: really different but put in same cluster
num.lumping.errors = num.lumping.errors +1
}
}
}
return(num.lumping.errors)
}
numlumperror(c(1,2,2),c(1,2,2))

[1] 0

numlumperror(c(1,2,2),c(2,1,1))

[1] 0

numlumperror(c(1,2,2),c(4,5,6))

[1] 0

numlumperror(c(1,2,2),c(1,1,1))

[1] 2
```

So we see that the function gives 0 number of lumping errors when the true class is (1,2,2) and the clusterings are (1,2,2),(2,1,1) and (4,5,6) but two lumping errors when the clustering is (1,1,1).So this is a valid function for counting the number of lumping errors.

B)Here I will write a function for finding the number of splitting errors and verify for the inputs given above.

```r
# Count number of errors made by splitting same classes into different cluster
# Uses explicit, nested iteration
# Input: vector of true classes, vector of guessed clusters
# Output: number of distinct pairs belonging to same class
# splitted into different clusters
numspliterror <- function(true.class,guessed.cluster) {
n = length(true.class)
# Checking if the two vectors have the same length!
stopifnot(n == length(guessed.cluster))
# Initialize the count variable
num.splitting.errors = 0
# checking all distinct pairs
for (i in 1:(n-1)) {
# think of above-diagonal entries in a matrix
for (j in (i+1):n) {
if ((true.class[i] == true.class[j])
& (guessed.cluster[i] != guessed.cluster[j])) {
# splitting error: really same but put in different clusters
num.splitting.errors = num.splitting.errors +1
}
}
}
return(num.splitting.errors)
}
numspliterror(c(1,2,2),c(1,2,2))

[1] 0

numspliterror(c(1,2,2),c(2,1,1))

[1] 0

numspliterror(c(1,2,2),c(4,5,6))

[1] 1

numspliterror(c(1,2,2),c(1,1,1))

[1] 0
```

ii)We repeat the process three times to get different clusterings.

```r
c1 = kmeans(data,centers=14)
c1$cluster

       CNS         CNS         CNS       RENAL      BREAST         CNS
         8           8           8           8           8           8
```

```
        CNS      BREAST       NSCLC       NSCLC       RENAL       RENAL 
          8           8          13          13          14          14 
      RENAL       RENAL       RENAL       RENAL       RENAL      BREAST 
         14          14          14          14          14           6 
      NSCLC       RENAL     UNKNOWN     OVARIAN    MELANOMA    PROSTATE 
          6          13          11          11          13           7 
    OVARIAN     OVARIAN     OVARIAN     OVARIAN     OVARIAN    PROSTATE 
          7           7           7           7           7           7 
      NSCLC       NSCLC       NSCLC    LEUKEMIA K562B-repro K562A-repro 
          4           4           4          12          12          12 
   LEUKEMIA    LEUKEMIA    LEUKEMIA    LEUKEMIA    LEUKEMIA       COLON 
         12          12          12          12          12           7 
      COLON       COLON       COLON       COLON       COLON       COLON 
          5           5           5           5           5           5 
MCF7A-repro      BREAST MCF7D-repro      BREAST       NSCLC       NSCLC 
          9           9           9           9           7           4 
      NSCLC    MELANOMA      BREAST      BREAST    MELANOMA    MELANOMA 
          4          10           2           2           3           1 
   MELANOMA    MELANOMA    MELANOMA    MELANOMA 
          1           3           1           3 
```

```
c2 = kmeans(data,centers=14)
c2$cluster
```

```
        CNS         CNS         CNS       RENAL      BREAST         CNS 
          5           5           5           5           5           5 
        CNS      BREAST       NSCLC       NSCLC       RENAL       RENAL 
         10           5          10          12           6           6 
      RENAL       RENAL       RENAL       RENAL       RENAL      BREAST 
          6           6           6           6           6           2 
      NSCLC       RENAL     UNKNOWN     OVARIAN    MELANOMA    PROSTATE 
          2          10           1           1          10           9 
    OVARIAN     OVARIAN     OVARIAN     OVARIAN     OVARIAN    PROSTATE 
          9           9           9           9           9           9 
      NSCLC       NSCLC       NSCLC    LEUKEMIA K562B-repro K562A-repro 
          9           9           9          13           3           3 
   LEUKEMIA    LEUKEMIA    LEUKEMIA    LEUKEMIA    LEUKEMIA       COLON 
          3          13          13          13          13           9 
      COLON       COLON       COLON       COLON       COLON       COLON 
          4           4           4           4           4           4 
MCF7A-repro      BREAST MCF7D-repro      BREAST       NSCLC       NSCLC 
         11          11          11          11           9          14 
      NSCLC    MELANOMA      BREAST      BREAST    MELANOMA    MELANOMA 
         14           8           8           8           7           7 
   MELANOMA    MELANOMA    MELANOMA    MELANOMA 
          7           7           7           7 
```

```
c3 = kmeans(data,centers=14)
c3$cluster

        CNS         CNS         CNS       RENAL      BREAST         CNS 
          6           6           4          11          11           4 
        CNS      BREAST       NSCLC       NSCLC       RENAL       RENAL 
          4           4           4           4           1           1 
      RENAL       RENAL       RENAL       RENAL       RENAL      BREAST 
          1           1           1           1           1           2 
      NSCLC       RENAL     UNKNOWN     OVARIAN    MELANOMA    PROSTATE 
          2           2           3           3          14          14 
    OVARIAN     OVARIAN     OVARIAN     OVARIAN     OVARIAN    PROSTATE 
          9           9           9           9          14          14 
      NSCLC       NSCLC       NSCLC    LEUKEMIA K562B-repro K562A-repro 
         13          13          13           8           8           8 
   LEUKEMIA    LEUKEMIA    LEUKEMIA    LEUKEMIA    LEUKEMIA       COLON 
          8           5           5           5           5          14 
      COLON       COLON       COLON       COLON       COLON       COLON 
         12          12          12          12          12          12 
MCF7A-repro      BREAST MCF7D-repro      BREAST       NSCLC       NSCLC 
          8           8           8           8          14          13 
      NSCLC    MELANOMA      BREAST      BREAST    MELANOMA    MELANOMA 
         13          10          10          10           7           7 
   MELANOMA    MELANOMA    MELANOMA    MELANOMA 
          7           7           7           7 
```

A)We find the number of lumping and splitting errors on each of these three runs.

```
data_class=rownames(data)
#Counting Lumping Errors
numlumperror(data_class,c1$cluster)

[1] 67

numlumperror(data_class,c2$cluster)

[1] 81

numlumperror(data_class,c3$cluster)

[1] 58
```

We find the number of splitting errors on each of these three runs.

```
#Counting Splitting Errors
numspliterror(data_class,c1$cluster)

[1] 91

numspliterror(data_class,c2$cluster)

[1] 95

numspliterror(data_class,c3$cluster)

[1] 101
```

Comments:-

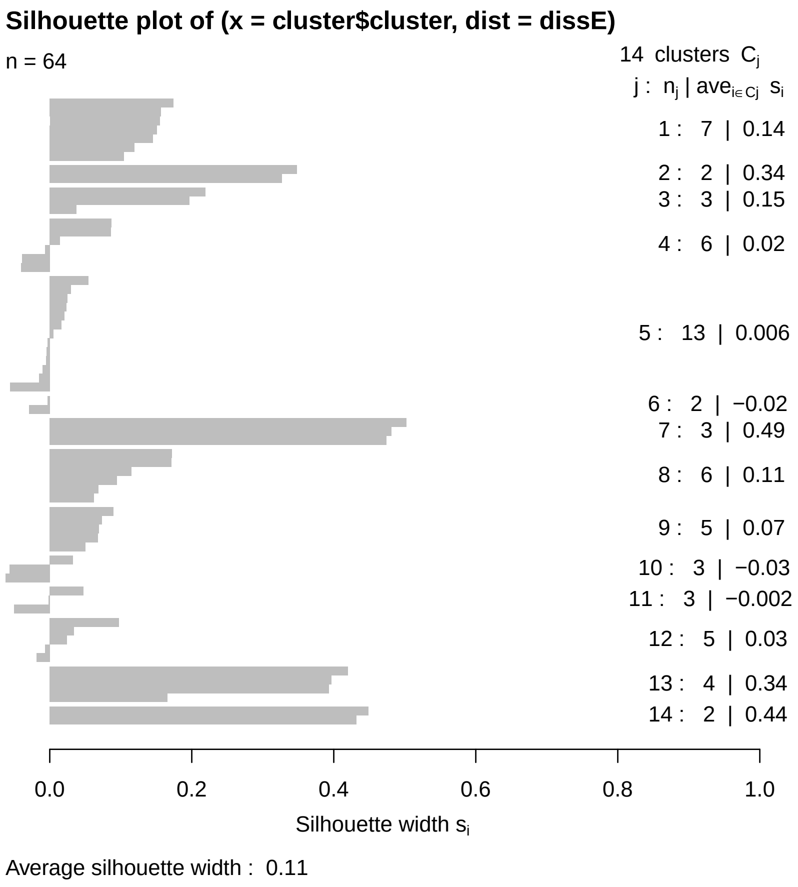
For comparison, there are 64* 63/2 = 2016 pairs, so the error rate here is quite good.

B)We find if there are any classes which seem particularly hard for K-means to pick up.For doing this first we see how good was the clustering process.For doing this we use the Silhouette Plot and Silhouette Coefficient which are defined as follows:-

- Suppose we are given a particular clustering, C_k, of the data into K clusters.
- c(i) is the cluster containing ith item.
- a_i = average dissimilarity of that ith item to all other members of the same cluster c(i).
- Let d(i,c) be the average dissimilarity of the ith item to all members of another cluster c.
- Suppose b_i = min d(i,c). Then b_i gives the least average dissimilarity of the ith item to other clusters where c is not c(i).
- If b_i = d(i,C), then C is the second-best cluster for the ith item.
- C will be called the neighbor of data point i .
- Comparing b_i with a_i gives an idea about the clustering.
- The ith silhouette value (or width) is given by si (CK) = siK = (bi - ai)/ max{ a_i,b_i}
- Clearly -1$\leq$ siK$\leq$ 1.
- Large positive value of siK : i.e. ai $<<$ bi means ith item is well-clustered.
- Large negative value of siK : i.e. ai $>>$ bi means poor clustering.
- siK $\simeq$0 means ith item lies between two clusters.

- max $\{$siK$\}<$ 0:25 indicates that either there are no definable clusters in the data or even there are, the clustering process could not identify it.
- Negative silhouette widths tend to attract attention: the items corresponding to these negative values are considered to be borderline allocations; they are neither well-clustered nor are they assigned by the clustering process to an alternative cluster.
- A silhouette plot is a bar plot of all the $\{$siK$\}$ after they are ranked in decreasing order, where the length of the ith bar is siK :
- The average silhouette width $\overline{s_K}$, is the average of all the $\{$siK$\}$
- The statistic $\overline{s_K}$ is a very useful indicator of the clustering C_K
- An alternative way to choose K is to minimize $\overline{s_K}$.
- As a tool of clustering diagonostic we use Silhouette Coefficient defined as SC $= max_K \overline{sK}$.

```
library(cluster)
dissE=daisy(data)
sk=silhouette(cluster$cluster,dissE)
plot(sk)
```

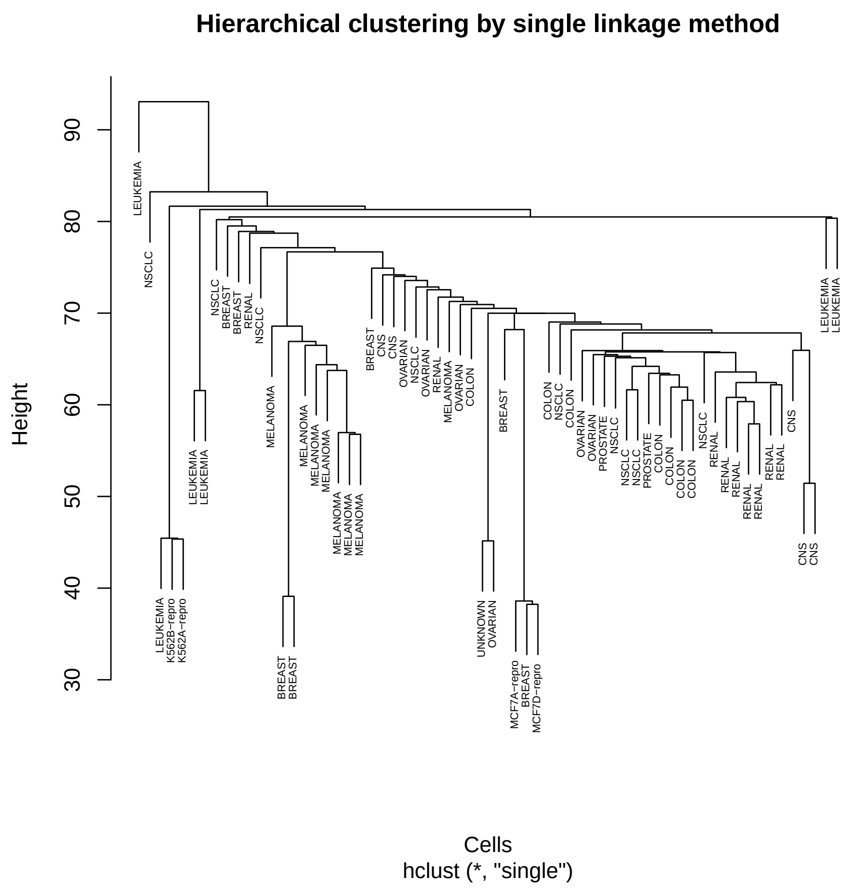


It is evident the clustering is moderately perfect.So there may be classes which was hard to pick up.So we use entropy for the Cluster Assignments.We start with the confusion matrix and then check for one run of K-means and then for more.BREAST and NSCLC tend not to be clustered together.

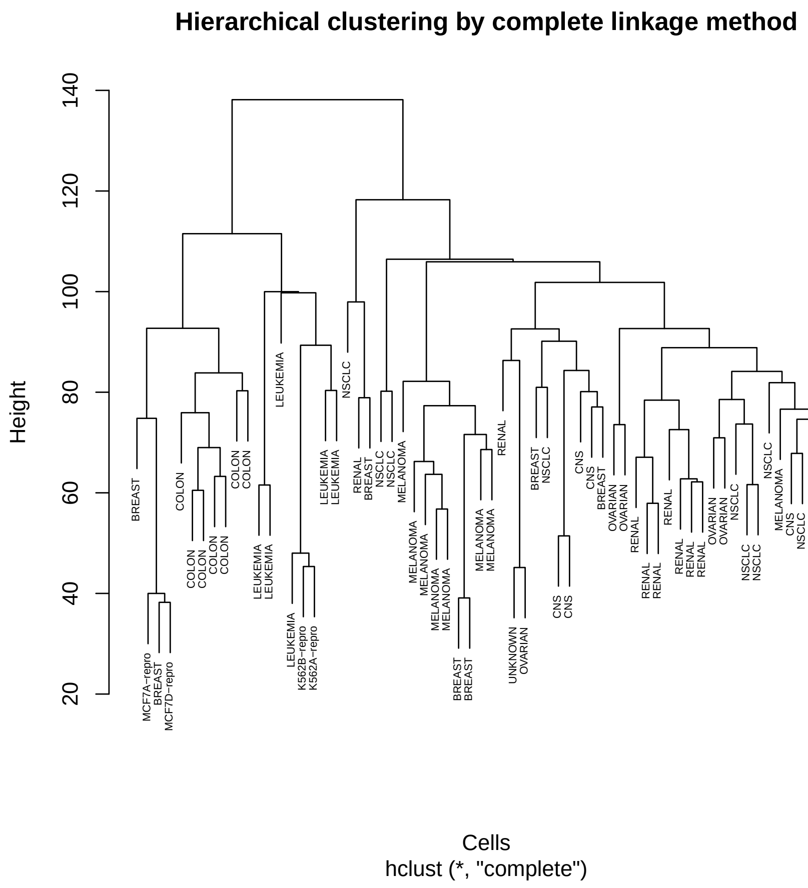
C)Now we find if there are any pair of cells which are always clustered together.We see cells number 1 and 2 are always clustered together in my three runs, and are CNS tumors. However, cells 3 and 4 are always clustered together, too, and one is CNS and one is RENAL.

b)Now we perform a hierarchial clustering of the dataset using single linkage and complete linkage,produce a dendogram and check which linkage performs better.

```
d = dist(data)
h1=hclust(d,method="single")
plot(h1,cex=0.5,xlab="Cells",main="Hierarchical clustering by single linkage method")
```



```
h2=hclust(d,method="complete")
plot(h2,cex=0.5,xlab="Cells",main="Hierarchical clustering by complete linkage method")
```



Comments:-

Single linkage is good with RENAL and decent with MELANOMA (though confused with BREAST). There are little sub-trees of cells of the same time, like COLON or CNS, but mixed together with others. The complete linkage method has sub-trees for COLON, LEUKEMIA, MELANOMA, and RENAL.So according to me the method of Single Linkage performs better.

c)The next objective is to reduce the dimension of the dataset.To achieve this we perform Principal Component Analysis on the dataset.

First let us state PCA.

Principal Component Analysis

Suppose we have data on d variables for n observations. Typically, in modern day applications, most of the data we deal with are such that $n << d$. This gives rise to many problems, which we collectively refer to as the **curse of dimensionality.** However, there are many ways to overcome these problems. The most common of them is to reduce the dimension of the data.

We can reduce the dimension in two ways:

- Either by discarding some of the d variables from our study,
- Or, by considering $k(\leq d)$ variables which are some linear combinations of the d variables under study.

Principal Component Analysis (PCA) deals with determining those linear combinations of these variables, which are appropriate in some sense

Let x_{ij}be the value of the j^{th} variable for the i^{th} observation, and $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$ denote the vector of values of the d variables for the i^{th} observation, $i = 1(1)n$, $j = 1(1)d$

Dimension reduction can be performed by making a transformation of the form

$$y_i^{k\times 1} = W^{k\times d} x_i \quad i = 1(1)n$$

where W is a suitable matrix of transformation. In PCA, W is such that its rows are mutually orthonormal.

If $X^{n\times d} = ((x_{ij})) = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix}$ be the original data matrix, and $Y^{n\times k} = ((y_{ij})) = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{pmatrix}$ be the data matrix of reduced dimension, then we have

$$Y = XW^T,$$

where the columns of W^T,i.e, $w_1, w_2, ..., w_k$ are mutually orthonormal.

Generally, the columns $w_1, w_2, ..., w_k$ are so chosen that w_i is the eigen-vector corresponding to the i^{th} largest eigen-value of S, where

$$S = \frac{1}{n}(X - 1\bar{x}^T)^T(X - 1\bar{x}^T)$$

is the sample dispersion matrix for the given data.

The vectors $w_1, w_2, ..., w_k$ are called the principal component directions, w_i being the i^{th} principal component direction, and, corresponding to w_i, the quantity $Xw_i = \begin{pmatrix} x_1^T w_i \\ x_2^T w_i \\ \vdots \\ x_n^T w_i \end{pmatrix}$, which is the vector of projections of the rows of X onto w_i, is called the vector of principal component projections or simply the principal component along the direction of w_i. The number k of principal components is determined such that these k components together account for most of the variability in X values.

The amount of variation in X explained by the i^{th} principal component Xw_i is $w_i^T S w_i$, which can be shown to be the i^{th} largest eigen-value of S. Thus, performing an eigen analysis of the sample dispersion matrix S helps a lot in doing PCA.

For the sake of ease in computation, instead of working with X, we work with the centred matrix $X - 1\bar{x}^T$.

i)We perform PCA and find the variances associated with the principal components.

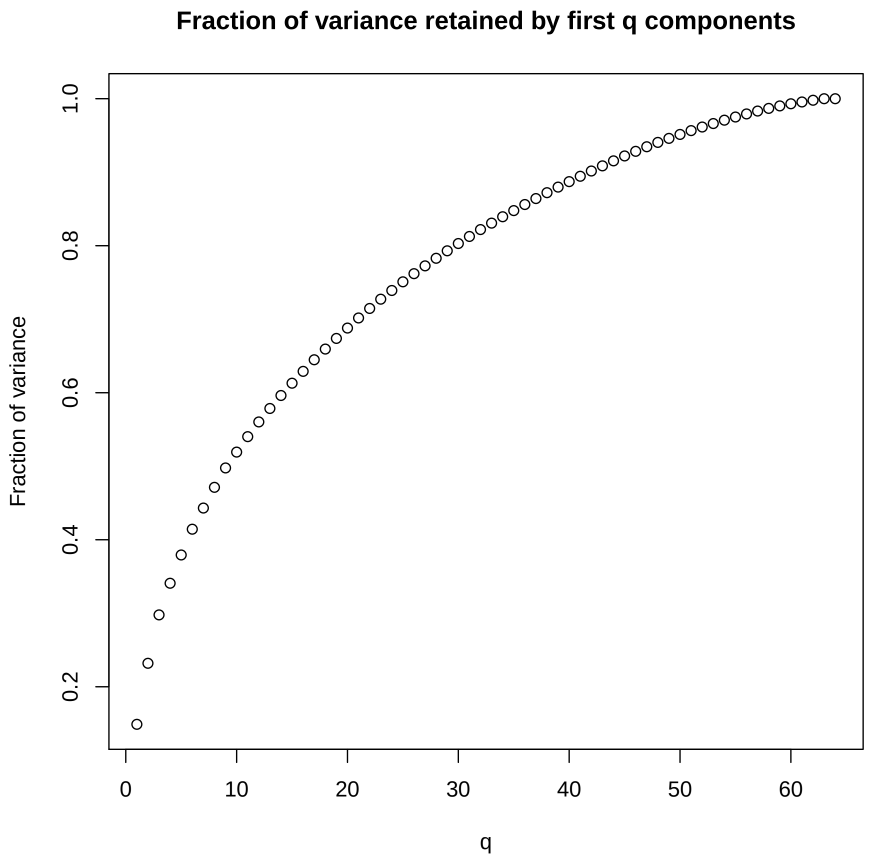
```
data.pca=prcomp(data)
variances = (data.pca$sdev)^2
signif(variances,2)

 [1] 6.3e+02 3.5e+02 2.8e+02 1.8e+02 1.6e+02 1.5e+02 1.2e+02 1.2e+02
 [9] 1.1e+02 9.2e+01 8.9e+01 8.5e+01 7.8e+01 7.5e+01 7.1e+01 6.8e+01
[17] 6.7e+01 6.2e+01 6.2e+01 6.0e+01 5.8e+01 5.5e+01 5.4e+01 5.0e+01
[25] 5.0e+01 4.7e+01 4.5e+01 4.4e+01 4.3e+01 4.2e+01 4.1e+01 4.0e+01
[33] 3.7e+01 3.7e+01 3.6e+01 3.5e+01 3.4e+01 3.4e+01 3.2e+01 3.2e+01
[41] 3.1e+01 3.0e+01 3.0e+01 2.9e+01 2.8e+01 2.7e+01 2.6e+01 2.5e+01
[49] 2.3e+01 2.3e+01 2.2e+01 2.1e+01 2.0e+01 1.9e+01 1.8e+01 1.8e+01
[57] 1.7e+01 1.5e+01 1.4e+01 1.2e+01 1.0e+01 9.9e+00 8.9e+00 1.2e-28
```

ii)There are 64 principal components in total.

This is because, when n $<$ p, it is necessarily true that the data lie on an n-dimensional subspace of the p-dimensional feature space, so there are only n orthogonal directions along which the data can vary at all. (Actually, one of the variances is very small because any 64 points lie on a 63-dimensional surface, so we really only need 63 directions, and the last variance is within numerical error of zero.) Alternately, we can imagine that there are (6830 - 64) = 6766 other principal components, each with zero variance.

iii)Now we plot the fraction of the total variance retained by the q principal components.

```
plot(cumsum(variances)/sum(variances),xlab="q",ylab="Fraction of variance",main = "Fraction
```

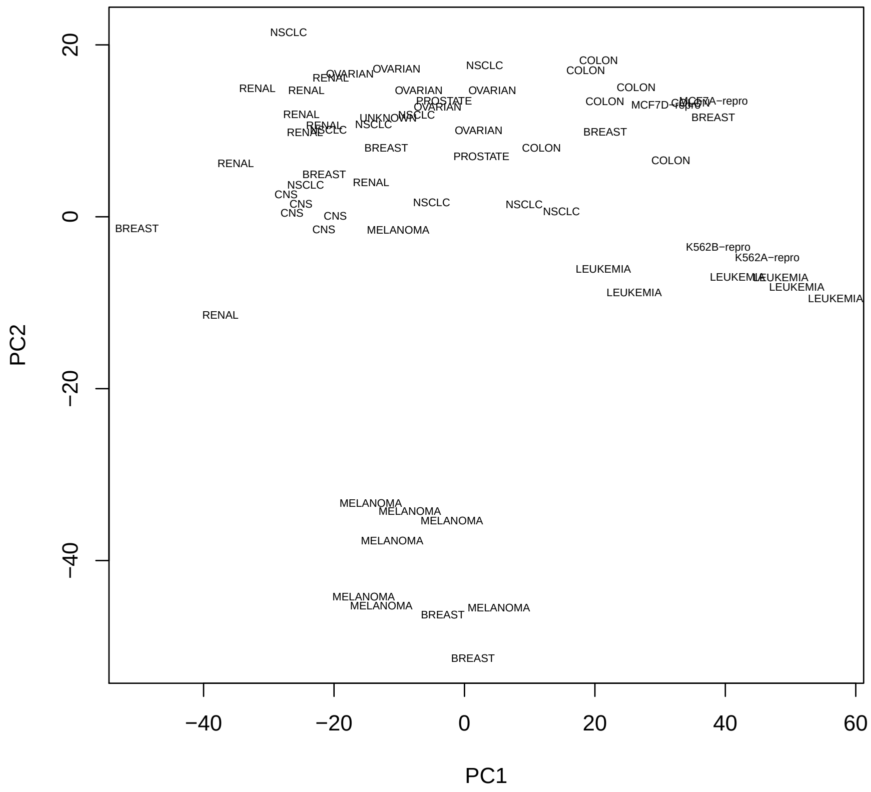


<u>Comments:-</u>

From the figure it is evident to see that we can use a lesser number of principal components than 63.

iv)We plot the projection of each cell on to the first principal component with its labels.

```
plot(data.pca$x[,1:2],type="n")
text(data.pca$x[,1:2],data_class, cex=0.5)
```



A)We see in MELANOMA most of the cells cluster in the bottom- center of the gure (PC1 between about 0 and -20, PC2 around -40), with only a few non-MELANOMA cells nearby, and a big gap to the rest of the data. Then we can also see that for LEUKEMIA in the center left, CNS in the upper left, RENAL above it, and COLON in the top right.So these are the tumor classes which form a cluster in these projections.

B)The cells in the class BREAST are scattered and they do not seem to form a cluster.

v)We find the amount of variation explained by the first two principal components.

```
sum(data.variances[1:2])/sum(data.variances)

Error in eval(expr, envir, enclos):  object 'data.variances' not found
```

Comments:-

So we see the first two principal components explain about 23% of the total variation.