Statistical & AI Techniques in Data Mining Project

MTH552A

Application of K-means Algorithm, Hierarchical Clustering and Principal Component Analysis on Iris and Breast Cancer Dataset

Rasamanjari Nandan (191111)

(1st Year, M. Sc. Statistics)

Indian Institute of Technology, Kanpur

Acknowledgement

I would like to express my heartfelt gratitude to Dr. Amit Mitra, Department of Mathematics and Statistics, IIT Kanpur, for giving me an opportunity, valuable guidance and inspiration to complete this project. It has been a great learning experience and has also provided me with a practical insight of the theoretical knowledge gathered during the course lectures.

THANK YOU

Contents :

Introduction

There are many industries where understanding how things group together is beneficial. For example, retailers want to understand the similarities among their customers to direct advertisement campaigns, botanists classify plants based on their shared similar characteristics, doctors want to group patients according to similar characteristics so that they may respond to the same treatments and many more. One way to group objects is to use clustering algorithms. Here I am going to explore the usefulness of unsupervised clustering algorithms to group plants with similar characteristics to do furthermore studies and to classify patients to understand which treatments might work well with them.

Again, nowadays dealing with high-dimensional data has been one of the most appealing and essential problem in a variety of domains. But the presence of high-dimensionality often creates trouble in classification problems as large memory and computational power is necessary while handling a large number of variables. In addition, such a large amount of information may confuse the classifier, thereby resulting in overfitting of training samples at the cost of poor generalization to new samples. So visualizing a high-dimensional data and reduction of variables thus become a natural requisite.

In this project, I have worked with two datasets, viz., "Iris" and "Breast Cancer Wisconsin (Diagnostic)". Here I have observed how K-means algorithm, Hierarchical Clustering (Complete and Single Linkage) perform on these datasets and which one is giving satisfactory result. Following this, I have performed Principal Component Analysis (PCA) to observe how effectively the dimension is getting reduced.

About the Data

Iris Dataset

1. Title: Iris Plants Database
   Updated Sept 21 by C.Blake - Added discrepency information

2. Sources:
   (a) Creator: R.A. Fisher
   (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
   (c) Date: July, 1988

3. Number of Instances: 150 (50 in each of three classes)

4. Number of Attributes: 4 numeric, predictive attributes and the class

5. Attribute Information:
   1. sepal length in cm
   2. sepal width in cm
   3. petal length in cm
   4. petal width in cm
   5. class:
      – Iris Setosa
      – Iris Versicolour
      – Iris Virginica

6. Missing Attribute Values: None

7. Class Distribution: 33.3% for each of 3 classes.

8. Link of the dataset

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/

from here I have downloaded iris.data

Breast Cancer Dataset

1. Title: Wisconsin Diagnostic Breast Cancer (WDBC)

2. Source Information

a) Creators:

Dr. William H. Wolberg, General Surgery Dept., University of Wisconsin, Clinical Sciences Center, Madison, WI 53792 wolberg@eagle.surgery.wisc.edu

W. Nick Street, Computer Sciences Dept., University of Wisconsin, 1210 West Dayton St., Madison, WI 53706 street@cs.wisc.edu 608-262-6619

Olvi L. Mangasarian, Computer Sciences Dept., University of Wisconsin, 1210 West Dayton St., Madison, WI 53706 olvi@cs.wisc.edu

b) Donor: Nick Street

c) Date: November 1995

3. Number of instances: 569

4. Number of attributes: 32 (ID, diagnosis, 30 real-valued input features)

5. Attribute information

1) ID number

2) Diagnosis (M = malignant, B = benign)

3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness ($perimeter^2/area - 1.0$)

g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)
k) feature 11 to feature 30

6. Missing attribute values: none

7. Class distribution: 357 benign, 212 malignant

8. Link of the dataset

https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/

from here I have downloaded wdbc.data

After downloading the datasets, I copied and pasted both the datasets in excel sheets and observed that all the data is being printed in one column. So I pasted the data in separate columns in excel sheets as follows: Data -> Text to Columns -> Delimited -> Next -> Comma -> Next -> Finish. Here I didn't specify the column names which I have done later on in R code.

K-means Algorithm

K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

Algorithm :

Suppose there are n data points.

Data : $\Psi = \{\vec{x_{1p}}, \vec{x_{2p}}, ..., \vec{x_{np}}\}$

Steps :

1. Specify number of clusters K.
2. i) Randomly partition the n cases into k clusters.
or
ii) Generate K random initial seed points and walk through the dataset to assign the n objects into k clusters corresponding to these seed points.
3. Reassign objects if the object is closer (Euclidean sense) to cluster center of another cluster than to its randomly assigned cluster.
4. Recalculate cluster means of clusters losing an object and also for the cluster gaining the object.
5. Continue step 2 & 3 till no further reassignment is possible.

Result :

We get K clusters of the n data points.

Hierarchical Clustering :

Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other based on their mutual distances and is visualized through a hierarchical tree called Dendogram.

A hierarchical tree is a nested set of partitions giving rise to hierarchical structure of clusters. It has certain characteristics such as sectioning the tree at a particular level partitions the data into g disjoint groups, sectioning the tree at two different levels and if we choose two groups from these two sections then the two groups are either disjoint or one is totally contained in the other.

Data : $\Psi = \{\vec{x_{1p}}, \vec{x_{2p}}, ..., \vec{x_{np}}\}$

Agglomerative Hierarchical Clustering Algorithm (AHC):

It operates with successive merger of objects. It starts with n clusters each having a single object. At each step, merge the two most similar group of objects, thus reducing the number of clusters by one. So after (n-1) steps, we merge all the cases to form a single cluster.

Divisive Hierarchical Clustering Algorithm :

It operates by successive splitting of objects. It starts with all objects in one cluster and in each step it splits clusters into two clusters such that the distance between the split clusters is the maximum. Proceeding in this way, we get n groups each having a single object. It is computationally inefficient.

Algorithm :

Steps for Agglomerative Hierarchical Clustering Algorithm:

1. Start with n objects in n clusters and an $n \times n$ symmetric distance matrix (D) where each element $((d_{ij}))$ indicates the distance between the i^{th} and j^{th} object.

2.Search the distance matrix for the most similar pair of objects. Suppose U & V is such that $d_{UV} = \underset{i,j}{min}\ d_{ij}$

3. Merge U & V to form (U,V) cluster. Update distance matrix by

i) delete row & column corresponding to U & V and

ii) add a new row & a column specifying the distance between (U,V) and remaining clusters.

4. Repeat steps 2 & 3 (n-1) times so that we get a single cluster with n objects. Record the clusters merged at each level and the merger levels.

5. Construct the dendogram tree with the information of mergers.

Complete Linkage AHC :

All the steps are same. Only in step 3 (Distance matrix updation stage) changes occur.

(U,V) -> merged cluster

W -> cluster from the previous stage

$d_{(U,V),W} = max\ (d_{U,W}, d_{V,W})$

Single Linkage AHC :

All the steps are same. Only in step 3 (Distance matrix updation stage) changes occur.

$d_{(U,V),W} = min\ (d_{U,W}, d_{V,W})$

Principal Component Analysis (PCA) :

Principal Component Analysis (PCA) is a technique that is useful for the compression and classification of data. The purpose is to reduce the dimensionality of a data set (sample) by finding a new set of variables, smaller than the original set of variables, that nonetheless retains most of the sample's information.

It starts with a set of n vectors $\Psi = (\vec{x_{1p}}, \vec{x_{2p}}, ..., \vec{x_{np}})$ from a p-variate population with mean $\vec{\mu_p}$ and covariance matrix $\Sigma_{p\times p}$. PCA aims at replacing each vector of the set Ψ with a p-dimensional vector $\vec{y} = (y_1, y_2, ..., y_p)$ such that the components of $\vec{y}$ i.e. $y_1, y_2, ..., y_p$ are uncorrelated, total variance of $\vec{x}$ = total variance of $\vec{y}$ and the total variation in $(y_1, y_2, ..., y_k) \approx$ the total variation in $(x_1, x_2, ..., x_p)$ where $k \ll p$. Though the last characteristic is desirable, still it can not be achieved always.

Now for the set Ψ, we calculate $\overline{\vec{x}}$ where

$$\overline{\vec{x}}' = (\overline{x_1}, ..., \overline{x_p}) = \frac{1}{n}\sum_{j=1}^{n} \vec{x_j} = \text{Sample mean vector} \tag{1}$$

and the observed sample covariance matrix is

$$S_{n-1} = \frac{1}{n-1}(\Psi\Psi' - n\overline{\vec{x}}\overline{\vec{x}}') = \frac{1}{n-1}\sum_{j=1}^{n}(\vec{x_j} - \overline{\vec{x}})(\vec{x_j} - \overline{\vec{x}})' \tag{2}$$

Now the sample principal components (P.C.s) are uncorrelated linear combinations $\vec{l}'\vec{x}$, where $\vec{x}$ is a vector from Ψ, maximizing the sample variance with sample variances in decreasing order i.e. i^{th} sample principle component is the linear combination $\vec{l_i'}\vec{x}$ such that the sample variance of $\vec{l}'\vec{x}$ is maximized at $\vec{l_i}$ keeping zero sample covariance between $\vec{l}'\vec{x_j}$ and $\vec{l_k'}\vec{x_j}$ $\forall\ k < i$.

Now let $(\hat{\lambda}_1, \hat{\vec{e_1}}), (\hat{\lambda}_2, \hat{\vec{e_2}}), ..., (\hat{\lambda}_p, \hat{\vec{e_p}})$ are the eigenvalue-orthonormal eigenvector pairs of S_{n-1} where $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq ... \geq \hat{\lambda}_p$. Then the i^{th} sample principle component is

$$\hat{y}_i = \hat{\vec{e}_i'}\vec{x}, \qquad\qquad i = 1(1)p \tag{3}$$

It can be shown that the sample variance of $\hat{y}_i = \hat{\vec{e}_i'} S_{n-1} \hat{\vec{e}_i}$ is $\hat{\lambda}_i$ and the sample covariance between $\hat{y}_i$ and $\hat{y}_j$ $(\forall i \neq j)$ is $\hat{\vec{e}_i'} S_{n-1} \hat{\vec{e}_j} = 0$. So, sample variance of $\hat{y}_1$ $(\hat{\lambda}_1) \geq$ sample variance of $\hat{y}_2$ $(\hat{\lambda}_2) \geq ... \geq$ sample variance of $\hat{y}_p$ $(\hat{\lambda}_p)$. Again total sample variation $\sum_{i=1}^{p} \hat{\lambda}_i = \sum_{i=1}^{p} S_{ii} = \mathrm{tr}S_{n-1}$. So, the proportion of total variation explained by the first k PCs is

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \tag{4}$$

If $\sum_{i=1}^{k} \lambda_i \approx \sum_{i=1}^{p} \lambda_i$ for $k \ll p$, then the data dimension reduction is most meaningful.

If units of variables are different or if the variables have widely varying ranges, then we should work with covariance matrix of standardized variables, i.e. the correlation matrix of the original variables and we can proceed according to the algorithm mentioned below.

Algorithm :

Data : $\Psi = \{\vec{x_{1p}}, \vec{x_{2p}}, ..., \vec{x_{np}}\}$ (in high-dimensional space)

Steps :

1) Calculate the mean vector $\overline{\vec{x}}$ using equation (1)
2) Calculate the covariance matrix S_{n-1} using equation (2) applying (1)
3) Calculate the eigenvalues and sort them in decreasing order.
4) Calculate the corresponding orthonormal eigenvectors.
5) Calculate the principal components using equation (3)
6) Determine the k using equation (4) or Scree Plot (described below) for which it seems that most of the variation is explained by the

resulting set with principal components.

Result :

$\nu = \{\vec{y_{1k}}, \vec{y_{2k}}, ..., \vec{y_{nk}}\}$

So we get k-dimensional set of vectors instead of p-dimensional set of vectors.

Other usages of PCA :

There are several other usages of PCA instead of only reducing the dimension of the dataset such as data projection and visualization, rough clustering detection, multidimensional outlier detection, checking for multivariate normality, ranking of $\vec{x_{1p}}, \vec{x_{2p}}, ..., \vec{x_{np}}$ based on the first principal components $\hat{y_1}^{(1)}, \hat{y_1}^{(2)}, ..., \hat{y_1}^{(n)}$, variable clustering detection as for x_i, we plot $(r_{x_i,\hat{y_1}}, r_{x_i,\hat{y_2}}, ..., r_{x_i,\hat{y_k}}) = r_i,\ \ i = 1(1)p$ and detect clustering.

Scree Plot :

A Scree Plot is a line plot of the eigenvalues of Principal Component Analysis. The scree plot is used to determine the number of principal components to keep in PCA. It displays the eigenvalues in a downward curve, ordering the eigenvalues from largest to smallest. According to the scree plot, the "elbow" of the graph where the eigenvalues seem to level off is found should be retained as significant.

Results :

Iris Dataset

Performing Exploratory Data Analysis (EDA)

EDA will help us learn more about the variables and make an informed decision about whether we should scale the data. Because k-means and hierarchical clustering measure similarity between points using a distance formula, it can place extra emphasis on certain variables that have a larger scale and thus larger differences between points.

First I need to check whether the data should be scaled or not. For that I have observed the summary of data.

```
> # Collecting evidence for the question 'should the data be scaled?'
> summary(data_1)
  sepal_length    sepal_width     petal_length    petal_width                  class
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
 Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Clearly the data don't vary too much. So I don't need to scale the data. Now here I have worked with the first four numeric columns namely sepal length, sepal width, petal length, petal width.

K-Means Algorithm

Here I have selected the number of clusters as 5 i.e. K = 5. Now it is also important to make sure that my results are reproducible when conducting a statistical analysis i.e. we should get same result repeatedly when we run on same data. Because I am doing an analysis that has a random aspect, it is necessary to set a seed to ensure reproducibility.

So at first I set the seed as 10. Now after running K-means algorithm for the first time I get the cluster sizes as

```
> first_clust_iris$size
[1] 12 39 24 50 25
```

Now different iterations of k-means can result in different clusters.If the algorithm is genuinely grouping similar observations (as opposed to clustering noise), then cluster assignments will be somewhat robust between various iterations of the algorithm.

With regards to Iris data, this would mean that the same flowers would be grouped even when the algorithm is initialized at different random points. If flowers are not in similar clusters with various algorithm runs, then the clustering method is not picking up on meaningful relationships between flowers.

I'm going to explore how the flowers are grouped with another iteration of the k-means algorithm. I will then be able to compare the resulting groups of flowers.

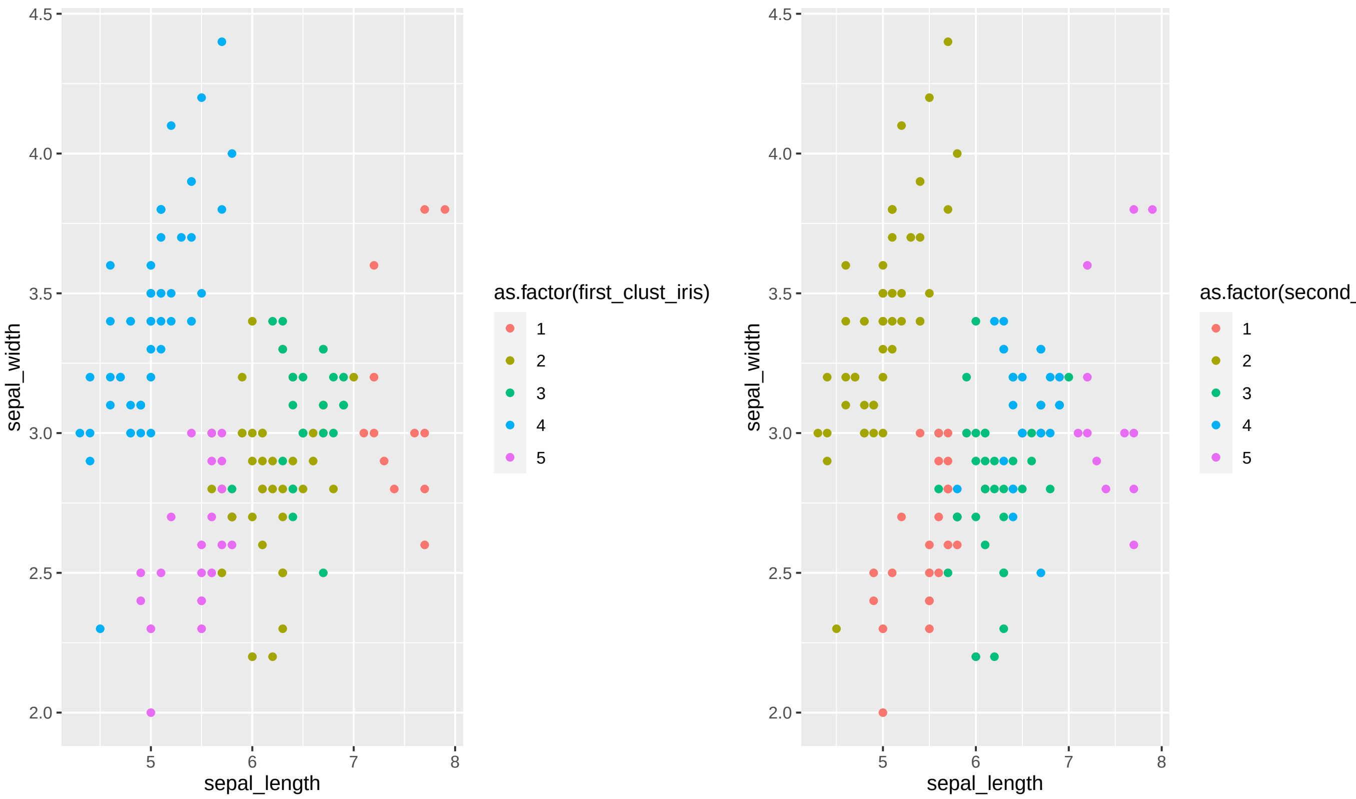
Now I set the seed as 38 and after running K-means algorithm I got the cluster sizes as

```
> second_clust_iris$size
[1] 27 50 37 24 12
```

It is important that the clusters are stable. Even though the algorithm begins by randomly initializing the cluster centers, if the k-means algorithm is the right choice for the data, then different initializations of the algorithm will result in similar clusters.

But here we can see that the cluster sizes vary a lot from first iteration to second iteration. The clusters from different iterations may not be the same, but the clusters should be roughly the same size and have similar distributions of variables. If there is a lot of change in clusters between different iterations of the algorithm, then k-means clustering is not the right choice for the data. This can be validated by visualizing the clusters' change between different iterations of the algorithm to get an idea of the cluster stabilities.

For checking effectiveness of clustering algorithm, I have selected a characteristic at random and considered another characteristic accordingly (I selected one characteristic at random because if k-means algorithm fails to give consistent clustering for the characteristic, then it is of no use to check whether k-means algorithm is working fine or not for other characteristics) and here I have chosen sepal length. Now after checking correlation I got sepal width has almost zero correlation with sepal length. So I opted these two characteristics for visualization as it will be easier to observe the clustering algorithm as in that case all the clusters will not fall on a straight line. Thereafter I have tried to visualize and compare the plots of that particular characteristics for k-means algorithm w.r.t. different iterations.



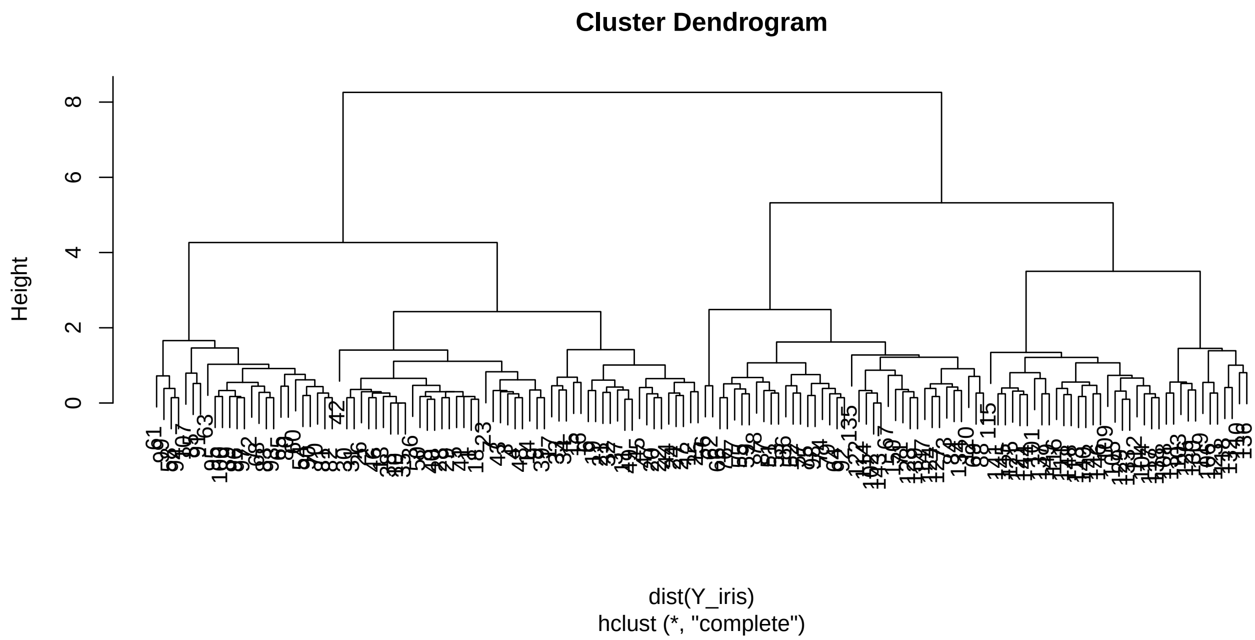
Clearly we can see there are different clustering assignments which is not desirable if K-means algorithm works well. So, I concluded that K-means algorithm is not a right choice for clustering in Iris data.

Hierarchical Clustering

An alternative to k-means clustering is hierarchical clustering. This method works well when data have a nested structure. Iris data might follow this type of structure. Hierarchical clustering also does not require the number of clusters to be selected before running the algorithm.

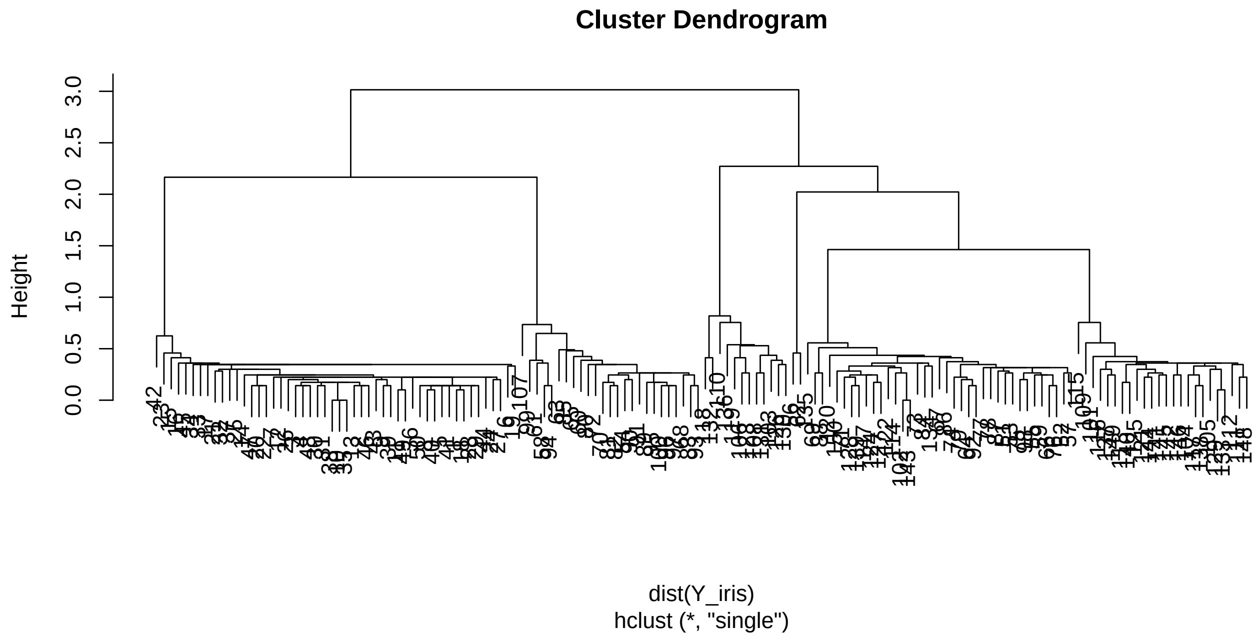
Clusters can be selected by using the dendrogram. The dendrogram allows us to see how similar observations are to one another, and they are useful in helping us choose the number of clusters to group the data.

So first I have visualized the plot of hierarchical clustering based on complete linkage method.



So from the picture it is clear that Complete Linkage HC is working fine in grouping the data and it is not responding to noise. So, I concluded that Complete Linkage Hierarchical Clustering can be a good choice for clustering Iris data.

Then I have visualized the plot of hierarchical clustering based on single linkage method.



So in this case Single Linkage Hierarchical Clustering is also working well. So, this can also be an optimal choice for clustering Iris data.

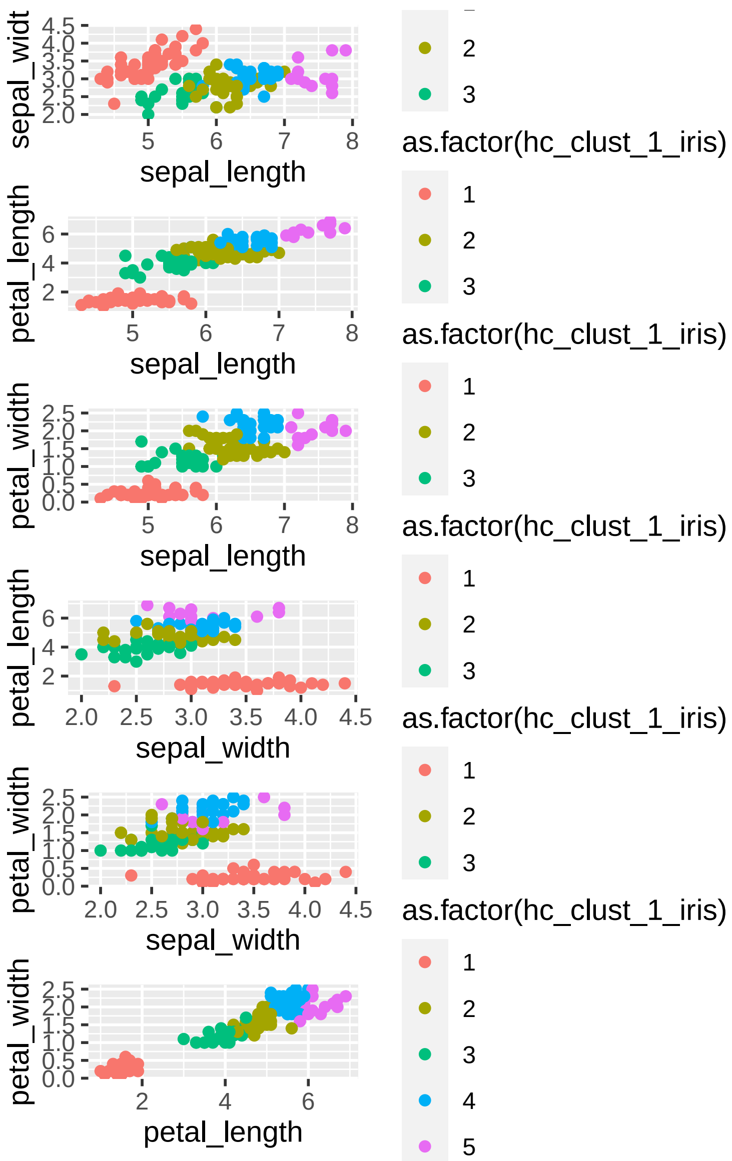
Now I get a summary for Complete Linkage HC for clustering assignment.

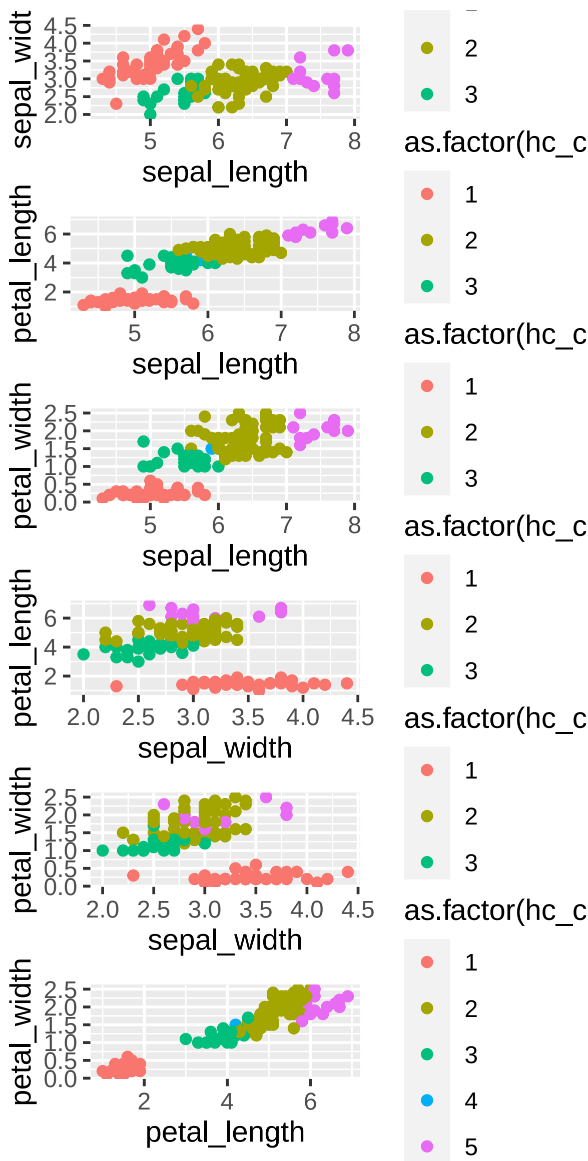
```
> clust_summary_iris
  hc_clust_1_iris sepal_length.avg sepal_length.sd sepal_width.avg sepal_width.sd
1               1         5.006000       0.3524897        3.418000      0.3810244
2               2         6.207692       0.3571706        2.853846      0.2780062
3               3         5.508000       0.3264966        2.600000      0.2614065
4               4         6.529167       0.2628922        3.058333      0.2263446
5               5         7.475000       0.2701010        3.125000      0.3980064
  petal_length.avg petal_length.sd petal_width.avg petal_width.sd
1         1.464000       0.1735112        0.244000      0.1072095
2         4.746154       0.2918599        1.564103      0.2170219
3         3.908000       0.3762978        1.204000      0.1790717
4         5.508333       0.2569329        2.162500      0.2261444
5         6.300000       0.3567530        2.050000      0.2540580
```

And here is the summary for Single Linkage HC.

```
> clust_summary_iris
  hc_clust_2_iris sepal_length.avg sepal_length.sd sepal_width.avg sepal_width.sd
1               1         5.006000       0.3524897        3.418000      0.3810244
2               2         6.347541       0.3505265        2.932787      0.2803101
3               3         5.508000       0.3264966        2.600000      0.2614065
4               4         5.800000       0.1414214        2.900000      0.1414214
5               5         7.475000       0.2701010        3.125000      0.3980064
  petal_length.avg petal_length.sd petal_width.avg petal_width.sd
1         1.464000       0.1735112        0.244000      0.1072095
2         5.059016       0.4540107        1.804918      0.3639717
3         3.908000       0.3762978        1.204000      0.1790717
4         4.350000       0.2121320        1.400000      0.1414214
5         6.300000       0.3567530        2.050000      0.2540580
```

Now I will look for several plots w.r.t. two HC algorithms.





From the plots I concluded that both Complete and Single linkage HC give almost same clustering assignment. Only in some cases Complete HC detects the assignment as cluster 2 whereas Single HC detects it as cluster 4.

So, ultimately I concluded

```
# Adding TRUE if the algorithm shows promise, adding FALSE if it does not
explore_kmeans_iris <- FALSE
explore_hierarch_complete_iris <- TRUE
explore_hierarch_single_iris <- TRUE
```

PCA

Now to conduct PCA, first I computed the sample variance-covariance matrix of iris data. Then I computed eigenvalues and orthonormal eigenvectors where the vectors are in column and got

```
> # Eigen-values of Z are
> lambda = eigen(Z)$value
> lambda
[1] 4.22484077 0.24224357 0.07852391 0.02368303
```

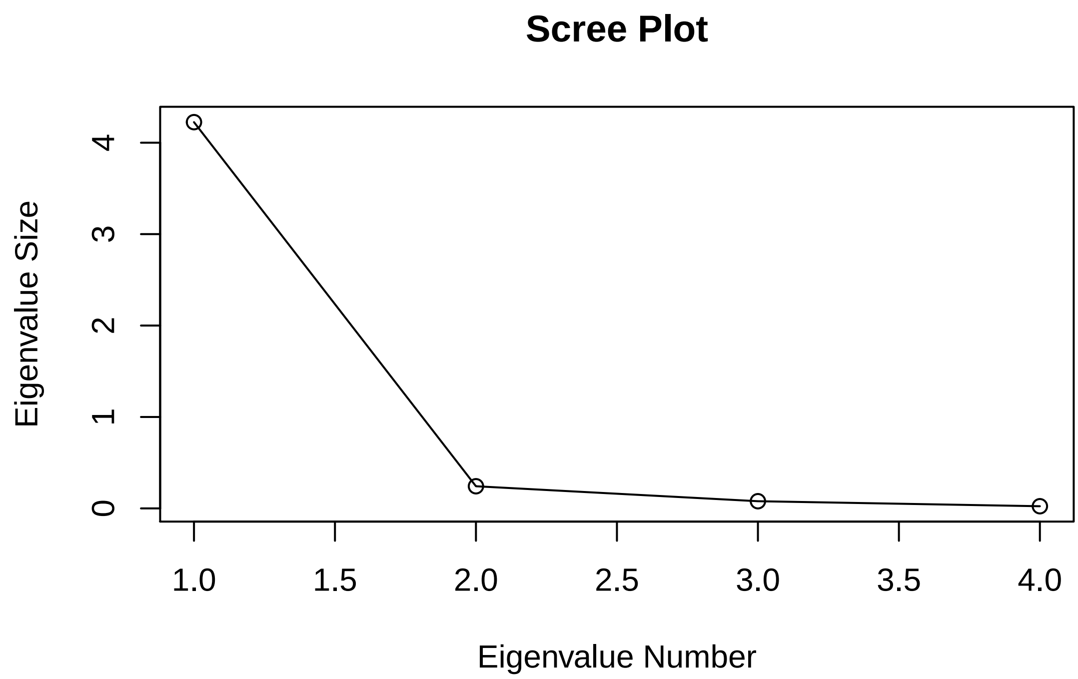
```
> # Orthonormal eigen-vectors are (in columns)
> v = eigen(Z)$vector
> v
            [,1]        [,2]        [,3]       [,4]
[1,]  0.36158968 -0.65653988  0.58099728  0.3172545
[2,] -0.08226889 -0.72971237 -0.59641809 -0.3240944
[3,]  0.85657211  0.17576740 -0.07252408 -0.4797190
[4,]  0.35884393  0.07470647 -0.54906091  0.7511206
```

Then I saw how the PCs are explaining the proportion of total variation.

```
[1] 0.9246162
[1] 0.05301557
[1] 0.01718514
[1] 0.005183085
```

So the first PC is explaining almost 92% variation of the total variation. So I could reduce the dimensionality of the 4-dimensional vector to 1-dimensional vector using the formula stated in theory.

And this is also evident from the scree plot.



So, ultimately I have computed the sample principal components.

Breast Cancer Data

Performing Exploratory Data Analysis (EDA)

First I checked summary statistic for wdbc data to understand whether the data should be scaled or not.

```
> # Collecting evidence for the question 'should the data be scaled?'
> summary(data_3)
   ID number          Diagnosis      radius           texture          perimeter
 Min.   :     8670   B:357      Min.   : 6.981   Min.   : 9.71   Min.   : 43.79
 1st Qu.:   869218   M:212      1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17
 Median :   906024              Median :13.370   Median :18.84   Median : 86.24
 Mean   : 30371831              Mean   :14.127   Mean   :19.29   Mean   : 91.97
 3rd Qu.:  8813129              3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10
 Max.   :911320502              Max.   :28.110   Max.   :39.28   Max.   :188.50
      area          smoothness        compactness        concavity
 Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000
 1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956
 Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154
 Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880
 3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070
 Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680
    concave            symmetry          fractal          feature_11
 Min.   :0.00000   Min.   :0.1060   Min.   :0.04996   Min.   :0.1115
 1st Qu.:0.02031   1st Qu.:0.1619   1st Qu.:0.05770   1st Qu.:0.2324
 Median :0.03350   Median :0.1792   Median :0.06154   Median :0.3242
 Mean   :0.04892   Mean   :0.1812   Mean   :0.06280   Mean   :0.4052
 3rd Qu.:0.07400   3rd Qu.:0.1957   3rd Qu.:0.06612   3rd Qu.:0.4789
 Max.   :0.20120   Max.   :0.3040   Max.   :0.09744   Max.   :2.8730
   feature_12        feature_13        feature_14        feature_15
 Min.   :0.3602   Min.   : 0.757   Min.   :  6.802   Min.   :0.001713
 1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169
 Median :1.1080   Median : 2.287   Median : 24.530   Median :0.006380
 Mean   :1.2169   Mean   : 2.866   Mean   : 40.337   Mean   :0.007041
 3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146
 Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130
```

```
  feature_16          feature_17          feature_18          feature_19      
 Min.   :0.002252   Min.   :0.00000   Min.   :0.000000   Min.   :0.007882  
 1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638   1st Qu.:0.015160  
 Median :0.020450   Median :0.02589   Median :0.010930   Median :0.018730  
 Mean   :0.025478   Mean   :0.03189   Mean   :0.011796   Mean   :0.020542  
 3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710   3rd Qu.:0.023480  
 Max.   :0.135400   Max.   :0.39600   Max.   :0.052790   Max.   :0.078950  
  feature_20           feature_21      feature_22      feature_23    
 Min.   :0.0008948   Min.   : 7.93   Min.   :12.02   Min.   : 50.41  
 1st Qu.:0.0022480   1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11  
 Median :0.0031870   Median :14.97   Median :25.41   Median : 97.66  
 Mean   :0.0037949   Mean   :16.27   Mean   :25.68   Mean   :107.26  
 3rd Qu.:0.0045580   3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40  
 Max.   :0.0298400   Max.   :36.04   Max.   :49.54   Max.   :251.20  
  feature_24       feature_25         feature_26         feature_27    
 Min.   : 185.2   Min.   :0.07117   Min.   :0.02729   Min.   :0.0000  
 1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145  
 Median : 686.5   Median :0.13130   Median :0.21190   Median :0.2267  
 Mean   : 880.6   Mean   :0.13237   Mean   :0.25427   Mean   :0.2722  
 3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829  
 Max.   :4254.0   Max.   :0.22260   Max.   :1.05800   Max.   :1.2520  
  feature_28         feature_29        feature_30     
 Min.   :0.00000   Min.   :0.1565   Min.   :0.05504  
 1st Qu.:0.06493   1st Qu.:0.2504   1st Qu.:0.07146  
 Median :0.09993   Median :0.2822   Median :0.08004  
 Mean   :0.11461   Mean   :0.2901   Mean   :0.08395  
 3rd Qu.:0.16140   3rd Qu.:0.3179   3rd Qu.:0.09208  
 Max.   :0.29100   Max.   :0.6638   Max.   :0.20750  
```

From the summary statistic it is evident that wdbc data should be scaled as it varies in different ranges. Now I have worked with the last 30 numeric columns except the first two labeling columns. So after scaling I have got the summary statistic as

```
> # Printing summary after scaling
> summary(Y_wdbc)
     radius             texture            perimeter              area        
 Min.   :-2.0279   Min.   :-2.2273   Min.   :-1.9828   Min.   :-1.4532  
 1st Qu.:-0.6888   1st Qu.:-0.7253   1st Qu.:-0.6913   1st Qu.:-0.6666  
 Median :-0.2149   Median :-0.1045   Median :-0.2358   Median :-0.2949  
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.4690   3rd Qu.: 0.5837   3rd Qu.: 0.4992   3rd Qu.: 0.3632  
 Max.   : 3.9678   Max.   : 4.6478   Max.   : 3.9726   Max.   : 5.2459  
   smoothness         compactness         concavity           concave       
 Min.   :-3.10935   Min.   :-1.6087   Min.   :-1.1139   Min.   :-1.2607  
 1st Qu.:-0.71034   1st Qu.:-0.7464   1st Qu.:-0.7431   1st Qu.:-0.7373  
 Median :-0.03486   Median :-0.2217   Median :-0.3419   Median :-0.3974  
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.63564   3rd Qu.: 0.4934   3rd Qu.: 0.5256   3rd Qu.: 0.6464  
 Max.   : 4.76672   Max.   : 4.5644   Max.   : 4.2399   Max.   : 3.9245  
    symmetry            fractal           feature_11         feature_12     
 Min.   :-2.74171   Min.   :-1.8183   Min.   :-1.0590   Min.   :-1.5529  
 1st Qu.:-0.70262   1st Qu.:-0.7220   1st Qu.:-0.6230   1st Qu.:-0.6942  
 Median :-0.07156   Median :-0.1781   Median :-0.2920   Median :-0.1973  
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.53031   3rd Qu.: 0.4706   3rd Qu.: 0.2659   3rd Qu.: 0.4661  
 Max.   : 4.48081   Max.   : 4.9066   Max.   : 8.8991   Max.   : 6.6494  
   feature_13         feature_14          feature_15         feature_16     
 Min.   :-1.0431   Min.   :-0.7372   Min.   :-1.7745   Min.   :-1.2970  
 1st Qu.:-0.6232   1st Qu.:-0.4943   1st Qu.:-0.6235   1st Qu.:-0.6923  
 Median :-0.2864   Median :-0.3475   Median :-0.2201   Median :-0.2808  
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.2428   3rd Qu.: 0.1067   3rd Qu.: 0.3680   3rd Qu.: 0.3893  
 Max.   : 9.4537   Max.   :11.0321   Max.   : 8.0229   Max.   : 6.1381  
```

```
  feature_17          feature_18          feature_19          feature_20     
 Min.   :-1.0566   Min.   :-1.9118   Min.   :-1.5315   Min.   :-1.0960  
 1st Qu.:-0.5567   1st Qu.:-0.6739   1st Qu.:-0.6511   1st Qu.:-0.5846  
 Median :-0.1989   Median :-0.1404   Median :-0.2192   Median :-0.2297  
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.3365   3rd Qu.: 0.4722   3rd Qu.: 0.3554   3rd Qu.: 0.2884  
 Max.   :12.0621   Max.   : 6.6438   Max.   : 7.0657   Max.   : 9.8429  
  feature_21          feature_22           feature_23          feature_24     
 Min.   :-1.7254   Min.   :-2.22204   Min.   :-1.6919   Min.   :-1.2213  
 1st Qu.:-0.6743   1st Qu.:-0.74797   1st Qu.:-0.6890   1st Qu.:-0.6416  
 Median :-0.2688   Median :-0.04348   Median :-0.2857   Median :-0.3409  
 Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.5216   3rd Qu.: 0.65776   3rd Qu.: 0.5398   3rd Qu.: 0.3573  
 Max.   : 4.0906   Max.   : 3.88249   Max.   : 4.2836   Max.   : 5.9250  
  feature_25          feature_26          feature_27          feature_28     
 Min.   :-2.6803   Min.   :-1.4426   Min.   :-1.3047   Min.   :-1.7435  
 1st Qu.:-0.6906   1st Qu.:-0.6805   1st Qu.:-0.7558   1st Qu.:-0.7557  
 Median :-0.0468   Median :-0.2693   Median :-0.2180   Median :-0.2233  
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.5970   3rd Qu.: 0.5392   3rd Qu.: 0.5307   3rd Qu.: 0.7119  
 Max.   : 3.9519   Max.   : 5.1084   Max.   : 4.6965   Max.   : 2.6835  
  feature_29          feature_30     
 Min.   :-2.1591   Min.   :-1.6004  
 1st Qu.:-0.6413   1st Qu.:-0.6913  
 Median :-0.1273   Median :-0.2163  
 Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.4497   3rd Qu.: 0.4504  
 Max.   : 6.0407   Max.   : 6.8408  
```

Now I have worked with this scaled data.

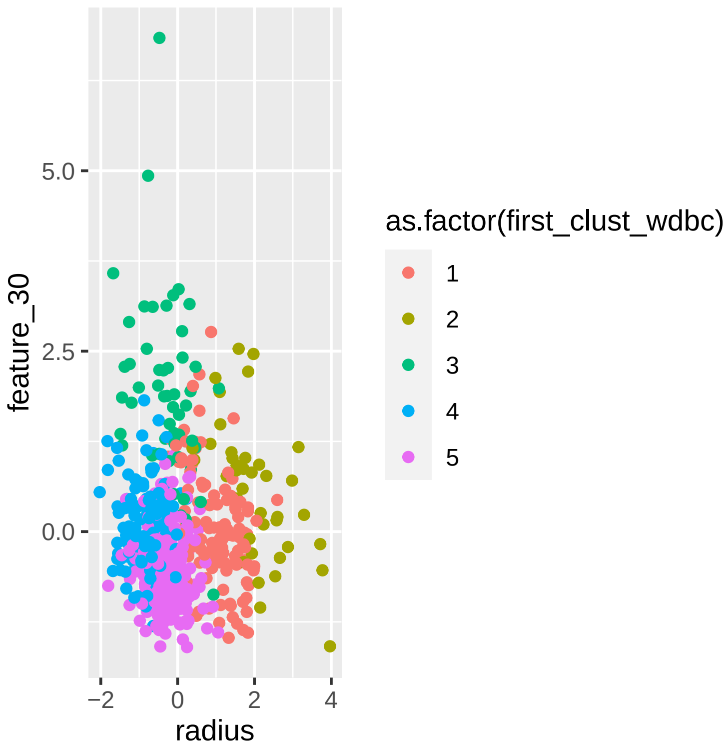
K-Means Algorithm

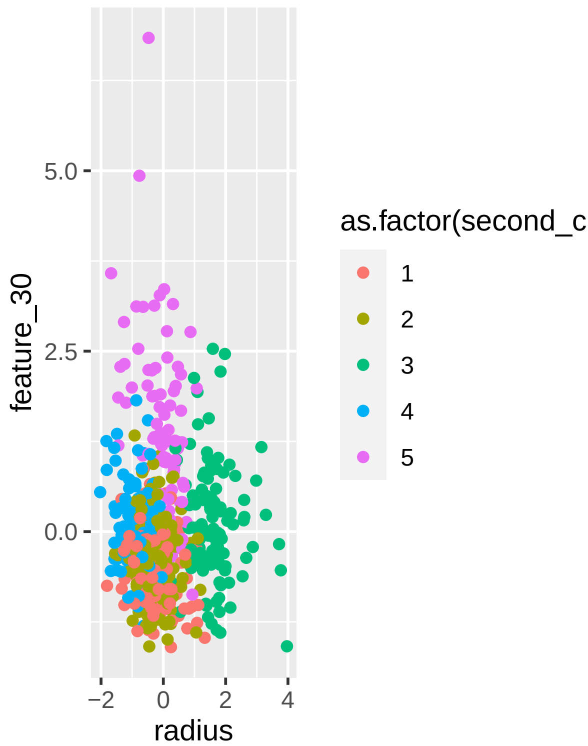
Now, due to similar arguments as of Iris data, here also I have conducted two iterations of K-means algorithm setting the seed at 10 and 38 respectively and got the results as follows :

```
> first_clust_wdbc$size
[1] 119  39  51 134 226
```

```
> second_clust_wdbc$size
[1] 107 172 109  99  82
```

Here also the cluster sizes differ widely from iteration to iteration. So, K-Means can not be a good choice. Still I plot some graphs to see the clustering assignment. Here I plotted radius and feature 30 for two different iterations.

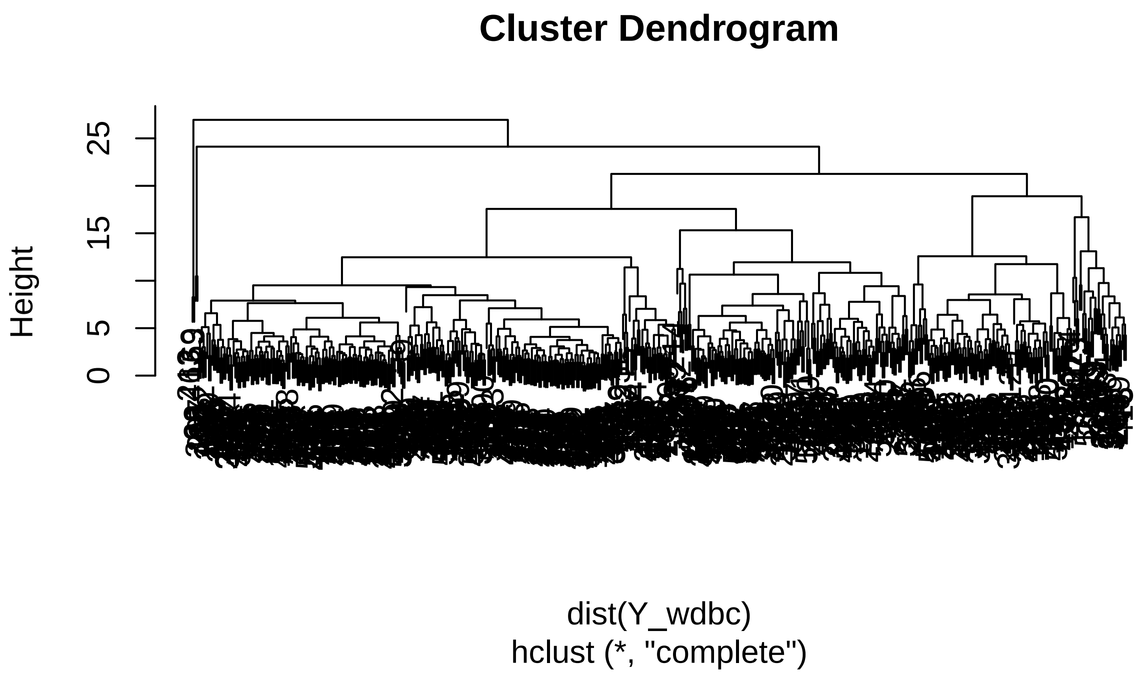




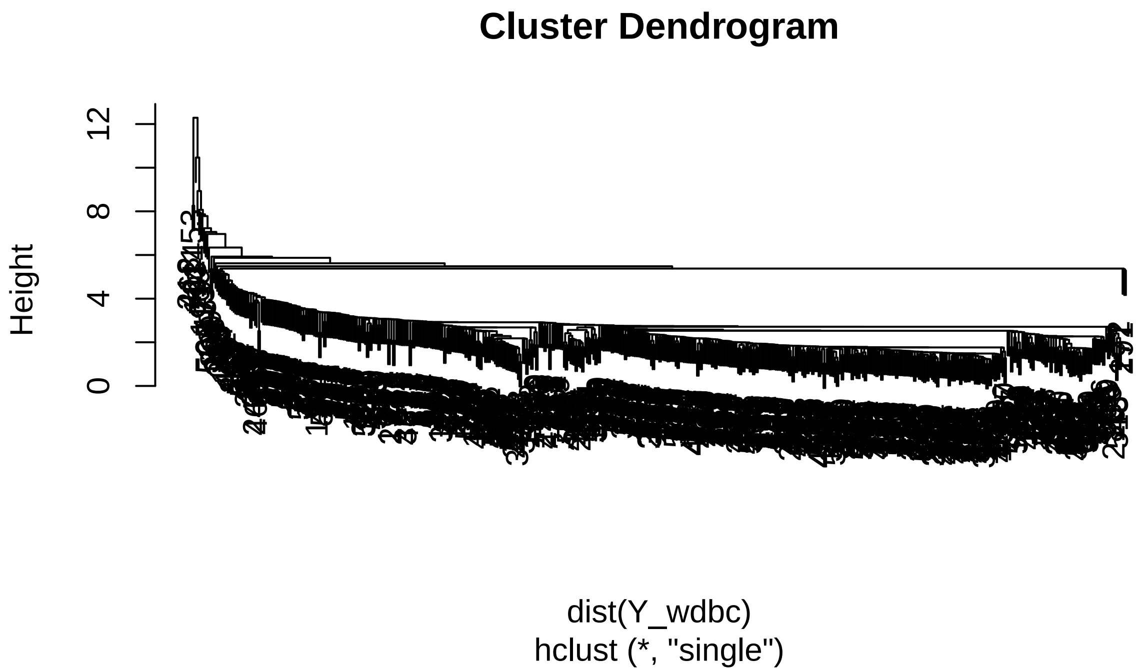
So we can see clustering assignment is also changing widely from iteration to iteration. So, I concluded that we can't use here K-means algorithm for further studies.

Hierarchical Clustering

First I have checked the dendogram for Complete Linkage HC Algorithm. Now, from the dendogram it is clear that Complete Linkage HC is working well in grouping the data and it is not responding to noise. So, I concluded that Complete Linkage Hierarchical Clustering can be a good choice for clustering wdbc data.



Then I have visualized the plot of hierarchical clustering based on single linkage method.



Clearly Single Linkage HC Algorithm is responding too much to noise. So it cannot be a good choice.

Now I get the summary for desired clustering algorithm, i.e. Complete Linkage HC Algorithm.

```
> clust_summary_wdbc
  hc_clust_wdbc radius.avg radius.sd texture.avg texture.sd perimeter.avg
1             1  1.3659338 0.8885130   0.8324394  0.9528660     1.4767946
2             2  1.3899661 0.6386809   0.5642947  0.9264262     1.3591955
3             3 -0.4377731 0.5516721  -0.1943482  0.9380195    -0.4398325
4             4 -1.3471129 0.1408576  -0.6869621  0.3271636    -1.2627704
5             5  3.8698976 0.1384498   0.7161883  1.2823499     3.9397112
  perimeter.sd   area.avg     area.sd smoothness.avg smoothness.sd compactness.avg
1   0.89332095  1.3569908 0.976267124      1.0093177     0.9687482       2.1156203
2   0.63382419  1.3866334 0.784542606      0.1525597     0.6968596       0.4423991
3   0.54528861 -0.4439194 0.433717316     -0.1208027     1.0151722      -0.2742838
4   0.14521032 -1.0784992 0.099863018      0.7494043     0.0301664       0.8759061
5   0.04656042  5.2430714 0.004018633      1.0622570     0.2916085       1.3417026
  compactness.sd concavity.avg concavity.sd concave.avg concave.sd symmetry.avg
1      1.0718720     1.9966987    0.9586272   1.9650529  0.8288375    1.3602272
2      0.6942472     0.7896003    0.7247870   1.0643821  0.7280532    0.1178633
3      0.8033400    -0.3658561    0.6645406  -0.4117078  0.6117816   -0.1406594
4      0.2490343     3.4257569    0.8674763   0.3154628  0.6345271    1.8890934
5      0.6319579     3.1736238    0.3849537   2.9709382  0.1712968    0.1564198
  symmetry.sd fractal.avg fractal.sd feature_11.avg feature_11.sd feature_12.avg
1   1.3258575  0.93182985 1.29360749      1.7245623     1.1449986     0.72742519
2   0.7069465 -0.61917638 0.64069205      0.8551246     0.8050065    -0.17811060
3   0.9410521  0.06109438 0.93735456     -0.3710122     0.4568275    -0.02177598
4   1.1271704  3.38685006 1.25189527      0.6158316     1.2675321     1.29094289
5   1.0652663 -0.99961008 0.09814859      8.3112950     0.8312522     0.31568400
  feature_12.sd feature_13.avg feature_13.sd feature_14.avg feature_14.sd
1     1.2833191     1.94867666     1.1772468      1.4670742     1.0111836
2     0.6677996     0.72719413     0.7611270      0.8734654     0.8193136
3     1.0140314    -0.35740230     0.4362241     -0.3638199     0.2572942
4     1.8842563     0.05585999     0.7652131     -0.1445798     0.5002021
5     0.2179072     8.63016617     1.1646068     10.8496815     0.2580284
  feature_15.avg feature_15.sd feature_16.avg feature_16.sd feature_17.avg
1     0.24102695     1.1449296     1.74377982     1.3179514      1.1968874
2    -0.25042279     0.6230148     0.02891925     0.5836667      0.1304504
3     0.02918315     1.0465522    -0.16077099     0.8851686     -0.1765831
4     1.07194067     0.3346520     3.65653370     0.3869543     10.5348721
5     1.16869280     1.3659267     0.85167012     1.0274031      1.3359240
```

	feature_17.sd	feature_18.avg	feature_18.sd	feature_19.avg	feature_19.sd
1	0.8495953	1.5661448	1.0109168	1.18627454	1.8368319
2	0.3877382	0.3623403	0.5917000	-0.33923333	0.7012322
3	0.7418840	-0.2331017	0.8636467	-0.02891066	0.8776483
4	2.1597799	5.0579288	2.2426969	2.19838912	0.5568665
5	0.3902596	1.3336277	1.3648708	1.43445056	2.6397695

	feature_20.avg	feature_20.sd	feature_21.avg	feature_21.sd	feature_22.avg
1	1.02423067	0.97866050	1.3944261	0.7625679	0.7555799
2	-0.18662155	0.43970027	1.4460089	0.7272366	0.5564812
3	-0.06413087	0.93470692	-0.4510148	0.5062569	-0.1824923
4	6.01064610	5.41967045	-1.1595096	0.1038735	-0.7495981
5	0.27289371	0.02191277	3.2702297	1.1601648	-0.1232007

	feature_22.sd	feature_23.avg	feature_23.sd	feature_24.avg	feature_24.sd
1	0.9436695	1.5476823	0.8180071	1.3215683	0.84813069
2	0.9356594	1.3777168	0.7175619	1.4329742	0.93661024
3	0.9453150	-0.4472817	0.5130103	-0.4486620	0.38319430
4	0.3635476	-1.1603650	0.1165796	-0.9273323	0.06930021
5	1.4841027	3.3506033	1.3194119	4.3837468	2.17960334

	feature_25.avg	feature_25.sd	feature_26.avg	feature_26.sd	feature_27.avg
1	0.5801642	0.9449090	1.7038860	1.3692764	1.5971412
2	0.3108564	0.8156184	0.3915936	0.7640060	0.6598442
3	-0.1160551	1.0170552	-0.2248335	0.8539015	-0.2968064
4	0.2772988	0.5884187	0.6520099	0.7159313	3.6650169
5	-0.3249148	0.6658422	0.2182263	1.2314197	1.1001189

	feature_27.sd	feature_28.avg	feature_28.sd	feature_29.avg	feature_29.sd
1	1.0770632	1.6811957	0.5745443	1.1777895	1.6887316
2	0.6912754	0.9959919	0.5782132	0.1624833	0.7540651
3	0.8129841	-0.3693908	0.7857922	-0.1276641	0.9088378
4	1.4587888	0.7826250	0.1925568	1.2401417	1.2800905
5	1.2310225	1.4664589	1.1080086	-1.2223803	1.1349374

	feature_30.avg	feature_30.sd
1	1.17223566	1.6241972
2	-0.08847758	0.7942497
3	-0.07382447	0.9151836
4	2.09033962	0.3288638
5	-1.06226307	0.7446416

Now I want to plot some of the characteristics of wdbc data w.r.t. clustering assignment. But there are many features. So I decided to plot some specific ones. But which characteristics should I plot?

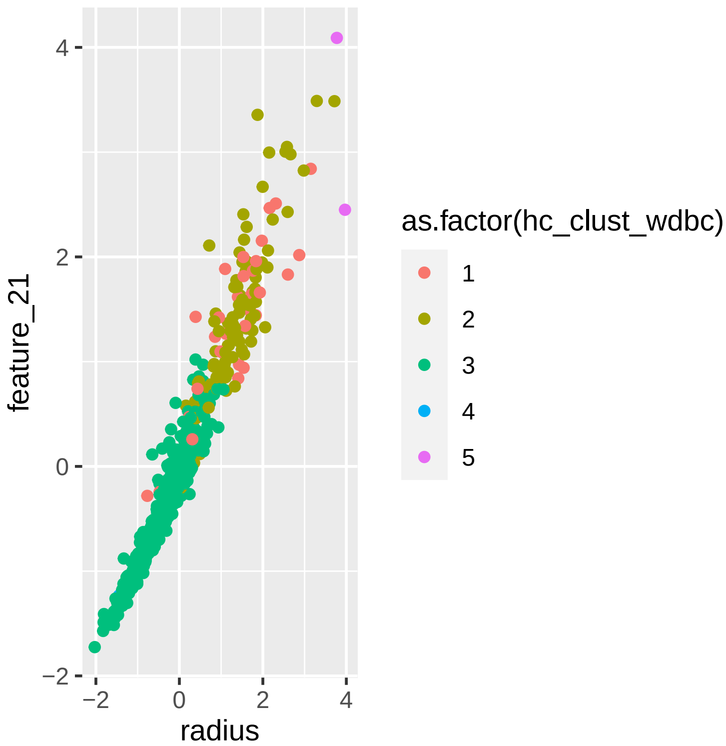
For that I have fitted Logistic Regression to know the importance of the features. Here the response variable i.e. Diagnosis is categorical in nature. So I have fitted logistic regression with a binomial family.

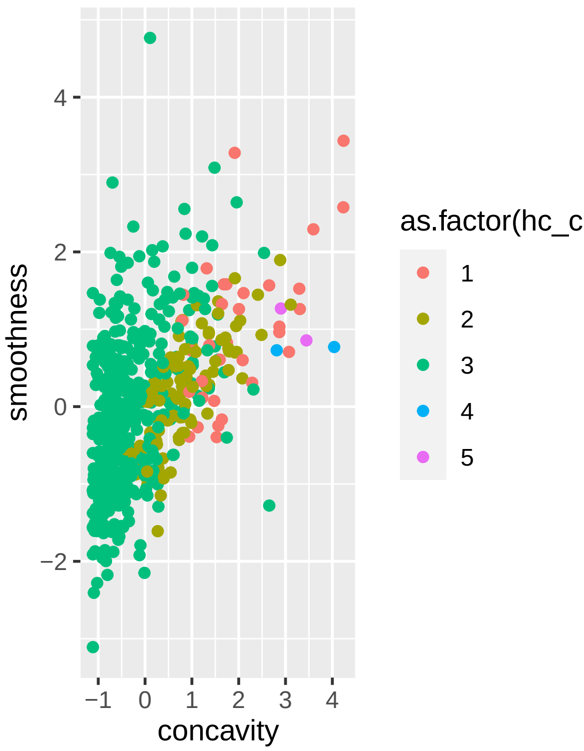
But to do so, I need to transform the non-numeric Diagnosis column to numeric column. For that I have opted Label Encoding Method. It simply converts each category of a variable into a number. This method is most useful when there is some notion of ordering in the categories of the variable.

So after denoting benign as 0 and malignant as 1 and fitting logistic regression, I got

```
Coefficients:
                 Estimate Std. Error   z value Pr(>|z|)    
(Intercept)     9.706e+14  2.279e+07  42587608   <2e-16 ***
radius          3.587e+15  1.739e+08  20634950   <2e-16 ***
texture        -6.731e+13  9.707e+06  -6934101   <2e-16 ***
perimeter       3.711e+14  1.733e+08   2141315   <2e-16 ***
area           -4.484e+15  5.260e+07 -85239335   <2e-16 ***
smoothness      7.199e+14  8.114e+06  88724686   <2e-16 ***
compactness    -9.228e+14  2.003e+07 -46079491   <2e-16 ***
concavity       1.102e+15  2.369e+07  46528800   <2e-16 ***
concave        -2.372e+14  2.197e+07 -10794988   <2e-16 ***
symmetry       -4.950e+14  5.786e+06 -85543045   <2e-16 ***
fractal        -8.434e+13  1.119e+07  -7534839   <2e-16 ***
feature_11      2.657e+13  2.470e+07   1075452   <2e-16 ***
feature_12     -1.804e+14  5.776e+06 -31235833   <2e-16 ***
feature_13      5.076e+14  2.370e+07  21416677   <2e-16 ***
feature_14      6.279e+14  1.938e+07  32395187   <2e-16 ***
feature_15      1.923e+14  5.652e+06  34021343   <2e-16 ***
feature_16     -5.917e+14  1.127e+07 -52508434   <2e-16 ***
feature_17     -4.939e+14  1.120e+07 -44080297   <2e-16 ***
feature_18      4.186e+14  9.559e+06  43790752   <2e-16 ***
feature_19      2.671e+13  6.428e+06   4155400   <2e-16 ***
feature_20      1.582e+14  8.895e+06  17780287   <2e-16 ***
feature_21      6.430e+15  7.964e+07  80738514   <2e-16 ***
feature_22      6.507e+14  1.213e+07  53623850   <2e-16 ***
feature_23     -1.736e+15  5.667e+07 -30635369   <2e-16 ***
feature_24     -3.668e+15  5.202e+07 -70513120   <2e-16 ***
feature_25     -2.304e+14  9.322e+06 -24720897   <2e-16 ***
feature_26     -1.533e+14  1.734e+07  -8843739   <2e-16 ***
feature_27      8.050e+14  1.594e+07  50503027   <2e-16 ***
feature_28      2.692e+14  1.712e+07  15727067   <2e-16 ***
feature_29      5.422e+14  8.689e+06  62407882   <2e-16 ***
feature_30      3.183e+14  1.243e+07  25600722   <2e-16 ***
hc_clust_wdbc  -5.841e+14  8.319e+06 -70212626   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

So I have plotted radius with feature 21 and concavity with smoothness.





So, ultimately I have concluded

```
# Adding TRUE if the algorithm shows promise, adding FALSE if it does not
explore_kmeans_wdbc <- FALSE
explore_hierarch_complete_wdbc <- TRUE
explore_hierarch_single_wdbc <- FALSE
```

PCA

As before, here also I have computed the sample variance-covariance matrix of wdbc data and got the eigenvalues and orthonormal eigenvectors as

```
> lambda = eigen(Z)$value
> lambda
 [1] 1.328161e+01 5.691355e+00 2.817949e+00 1.980640e+00 1.648731e+00 1.207357e+00
 [7] 6.752201e-01 4.766171e-01 4.168948e-01 3.506935e-01 2.939157e-01 2.611614e-01
[13] 2.413575e-01 1.570097e-01 9.413497e-02 7.986280e-02 5.939904e-02 5.261878e-02
[19] 4.947759e-02 3.115940e-02 2.997289e-02 2.743940e-02 2.434084e-02 1.805501e-02
[25] 1.548127e-02 8.177640e-03 6.900464e-03 1.589338e-03 7.488031e-04 1.330448e-04
```

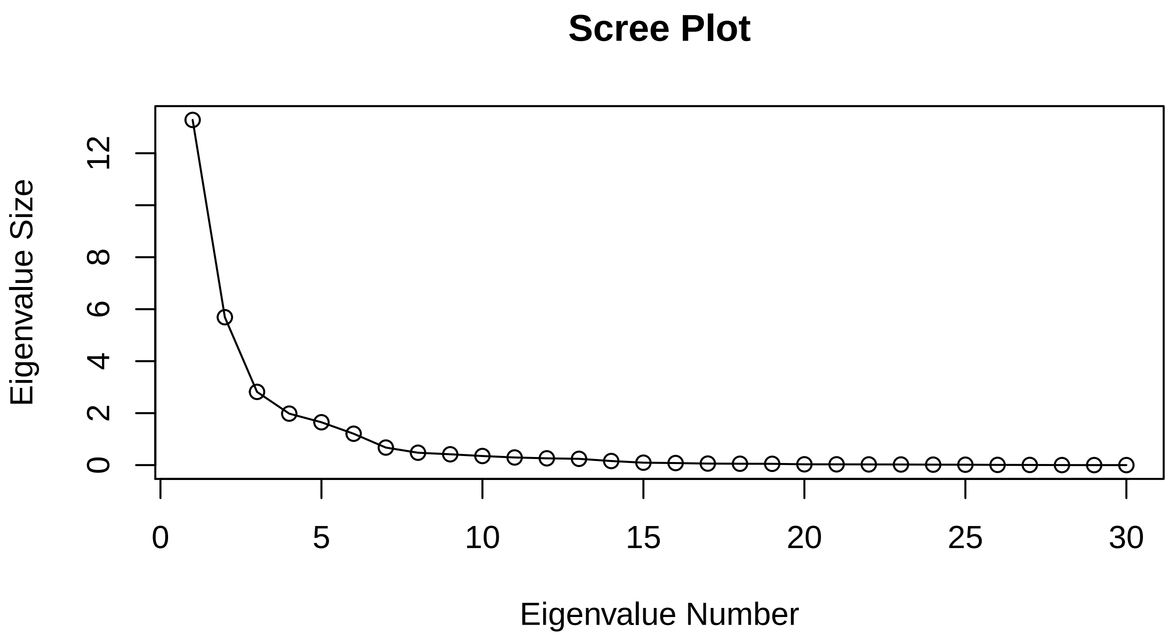
Here I haven't printed the eigenvectors because it is a 30×30 matrix.

Then I have seen how the PCs are explaining the proportion of total variation.

```
[1] 0.4427203
[1] 0.1897118
[1] 0.09393163
[1] 0.06602135
[1] 0.05495768
[1] 0.04024522
[1] 0.02250734
[1] 0.01588724
[1] 0.01389649
[1] 0.01168978
[1] 0.00979719
[1] 0.008705379
[1] 0.00804525
[1] 0.005233657
[1] 0.003137832
[1] 0.002662093
[1] 0.001979968
[1] 0.001753959
[1] 0.001649253
[1] 0.001038647
[1] 0.0009990965
[1] 0.0009146468
[1] 0.0008113613
[1] 0.0006018336
[1] 0.0005160424
[1] 0.000272588
[1] 0.0002300155
[1] 5.297793e-05
[1] 2.49601e-05
[1] 4.434827e-06
```

So the first five PCs are explaining almost 85% variation of the total variation. So I could reduce the dimensionality of the 30-dimensional vector to 5-dimensional vector using the formula stated in theory.

And this is also evident from the scree plot.



Thereafter I have computed the sample principal components.

With this my analyses of two datasets end.

R Code :

```
getwd()

setwd("C:/Users/user/Desktop/Projects/AI Project Final")

# Working with Iris Dataset

# Importing the data

data_1 = read.csv("iris_1.csv" , header = FALSE)

attach(data_1)

# Changing the names of the variables

names(data_1) <- c('sepal_length', 'sepal_width', 'petal_length', 'petal_width','class')

# Printing the first ten rows

head(data_1,n = 10)

# Collecting evidence for the question 'should the data be scaled?'

summary(data_1)

# Seperating columns from the original data to work with

data_2 = data_1[,1:4]

Y_iris = data_2

# Printing the first ten rows

head(Y_iris,n = 10)

# Setting the seed so that the results are reproducible

seed_val  <- 10

set.seed(seed_val)

# Selecting a number of clusters

k=5

# Running the k-means algorithm

first_clust_iris = kmeans(Y_iris, centers = 5, nstart = 1)
```

```
# How many flowers are in each cluster?

first_clust_iris$size

# Setting the seed

seed_val <- 38

set.seed(seed_val)

# Selecting a number of clusters and run the k-means algorithm

second_clust_iris = kmeans(Y_iris, centers = 5, nstart = 1)

# How many flowers are in each cluster?

second_clust_iris$size

# Adding cluster assignments to the data

Y_iris["first_clust_iris"] <- first_clust_iris$cluster

Y_iris["second_clust_iris"] <- second_clust_iris$cluster

# Printing the first ten rows

head(Y_iris,n = 10)

# Checking correlation

cor(Y_iris)

# Observing first characteristic of flowers

p_iris = cor(Y_iris)[,'sepal_length']

p_iris[order(-p_iris),drop = FALSE]

# Loading ggplot2

library(ggplot2)

# Creating the plot of sepal length and sepal width for the first clustering algorithm

plot_one_iris  <- ggplot(Y_iris, aes(x = sepal_length, y = sepal_width, color = as.factor(first_clust_iris))) +
geom_point()

# Creating the plot of sepal length and sepal width for the second clustering algorithm
```

```
plot_two_iris  <- ggplot(Y_iris, aes(x = sepal_length, y = sepal_width, color =
as.factor(second_clust_iris))) + geom_point()

# Installing and loading gridExtra package

install.packages("gridExtra")

library(gridExtra)

grid.arrange(plot_one_iris, plot_two_iris, ncol = 2)

# Executing hierarchical clustering with complete linkage

hier_clust_1_iris <- hclust(dist(Y_iris), method = "complete")

# Printing the dendrogram

plot(hier_clust_1_iris)

# Getting cluster assignments based on number of selected clusters

hc_1_assign_iris <- cutree(hier_clust_1_iris, k = 5)

# Executing hierarchical clustering with single linkage

hier_clust_2_iris <- hclust(dist(Y_iris), method = "single")

# Printing the dendrogram

plot(hier_clust_2_iris)

# Getting cluster assignments based on number of selected clusters

hc_2_assign_iris <- cutree(hier_clust_2_iris, k = 5)

# Adding assignment of chosen hierarchical linkage

Y_iris["hc_clust_1_iris"] <- hc_1_assign_iris

# Checking how complete and single hierarchical clustering differs

hc_1_assign_iris == hc_2_assign_iris

# Removing the first_clust_iris, and second_clust_iris variables

hd_simple_iris <- Y_iris[,!(names(Y_iris) %in% c("first_clust_iris","second_clust_iris"))]

# Printing first 10 rows

head(hd_simple_iris, n = 10)
```

```
# Getting the mean and standard deviation summary statistics

clust_summary_iris <- do.call(data.frame, aggregate(. ~ hc_clust_1_iris, data = hd_simple_iris,
function(x) c(avg = mean(x), sd = sd(x))))

clust_summary_iris

# Adding assignment of chosen hierarchical linkage

Y_iris["hc_clust_2_iris"] <- hc_2_assign_iris

# Removing the first_clust_iris, second_clust_iris and hc_clust_1_iris variables

hd_simple_iris <- Y_iris[,!(names(Y_iris) %in% c("first_clust_iris","second_clust_iris","hc_clust_1_iris"))]

# Printing first 10 rows

head(hd_simple_iris, n = 10)

# Getting the mean and standard deviation summary statistics

clust_summary_iris <- do.call(data.frame, aggregate(. ~ hc_clust_2_iris, data = hd_simple_iris,
function(x) c(avg = mean(x), sd = sd(x))))

clust_summary_iris

hd_simple_iris["hc_clust_1_iris"] <- hc_1_assign_iris

# Here we will look at several plots

# Plotting sepal_length and sepal_width

plot_one_iris_sp.le_sp.wi <- ggplot(hd_simple_iris,aes(x = sepal_length, y = sepal_width, color =
as.factor(hc_clust_1_iris))) + geom_point()

# Plotting sepal_length and sepal_width

plot_two_iris_sp.le_sp.wi <- ggplot(hd_simple_iris,aes(x = sepal_length, y = sepal_width, color =
as.factor(hc_clust_2_iris))) + geom_point()

# Plotting sepal_length and petal_length

plot_one_iris_sp.le_pe.le <- ggplot(hd_simple_iris,aes(x = sepal_length, y = petal_length, color =
as.factor(hc_clust_1_iris))) + geom_point()

# Plotting sepal_length and sepal_width
```

plot_two_iris_sp.le_pe.le <- ggplot(hd_simple_iris,aes(x = sepal_length, y = petal_length, color = as.factor(hc_clust_2_iris))) + geom_point()

Plotting sepal_length and petal_width

plot_one_iris_sp.le_pe.wi <- ggplot(hd_simple_iris,aes(x = sepal_length, y = petal_width, color = as.factor(hc_clust_1_iris))) + geom_point()

Plotting sepal_length and sepal_width

plot_two_iris_sp.le_pe.wi <- ggplot(hd_simple_iris,aes(x = sepal_length, y = petal_width, color = as.factor(hc_clust_2_iris))) + geom_point()

Plotting sepal_width and petal_length

plot_one_iris_sp.wi_pe.le <- ggplot(hd_simple_iris,aes(x = sepal_width, y = petal_length, color = as.factor(hc_clust_1_iris))) + geom_point()

Plotting sepal_width and petal_length

plot_two_iris_sp.wi_pe.le <- ggplot(hd_simple_iris,aes(x = sepal_width, y = petal_length, color = as.factor(hc_clust_2_iris))) + geom_point()

Plotting sepal_width and petal_width

plot_one_iris_sp.wi_pe.wi <- ggplot(hd_simple_iris,aes(x = sepal_width, y = petal_width, color = as.factor(hc_clust_1_iris))) + geom_point()

Plotting sepal_width and petal_width

plot_two_iris_sp.wi_pe.wi <- ggplot(hd_simple_iris,aes(x = sepal_width, y = petal_width, color = as.factor(hc_clust_2_iris))) + geom_point()

Plotting petal_length and petal_width

plot_one_iris_pe.le_pe.wi <- ggplot(hd_simple_iris,aes(x = petal_length, y = petal_width, color = as.factor(hc_clust_1_iris))) + geom_point()

Plotting petal_length and petal_width

plot_two_iris_pe.le_pe.wi <- ggplot(hd_simple_iris,aes(x = petal_length, y = petal_width, color = as.factor(hc_clust_2_iris))) + geom_point()

Printing the plots

grid.arrange(plot_one_iris_sp.le_sp.wi, plot_two_iris_sp.le_sp.wi, plot_one_iris_sp.le_pe.le, plot_two_iris_sp.le_pe.le, plot_one_iris_sp.le_pe.wi, plot_two_iris_sp.le_pe.wi,

```
plot_one_iris_sp.wi_pe.le, plot_two_iris_sp.wi_pe.le, plot_one_iris_sp.wi_pe.wi,
plot_two_iris_sp.wi_pe.wi, plot_one_iris_pe.le_pe.wi, plot_two_iris_pe.le_pe.wi, ncol=2)

# Adding TRUE if the algorithm shows promise, adding FALSE if it does not

explore_kmeans_iris <- FALSE

explore_hierarch_complete_iris <- TRUE

explore_hierarch_single_iris <- TRUE

# PCA for iris dataset

Y_iris = as.matrix(data_2)

# Sample variance covariance matrix of iris dataset with working columns

Z = cov(Y_iris)

# Computing the eigenvalues and corresponding eigenvectors of Z

eigen(Z)

# Eigen-values of Z are

lambda = eigen(Z)$value

lambda

# Orthonormal eigen-vectors are (in columns)

v = eigen(Z)$vector

v

# Ordering of eigenvalues in descending order

order(lambda)

# Proportion of total variation explained by the principal components

for (i in lambda) {

  print(i / sum(lambda))

}

# Scree Plot

plot(lambda, xlab = 'Eigenvalue Number', ylab = 'Eigenvalue Size', main = 'Scree Plot')
```

```
lines(lambda)

# Calculating sample PCAs from observed data matrix

X_pca_iris = rep(0,nrow(Y_iris))

for (i in 1:nrow(Y_iris)) {

  X_pca_iris[i] = crossprod(v[,1],Y_iris[i,])

}

X_pca_iris

# Working with Breast Cancer Dataset

# Importing the data

data_3 = read.csv("wdbc_1.csv" , header = FALSE)

attach(data_3)

# Changing the names of the variables

names(data_3) <- c('ID number', 'Diagnosis', 'radius',      'texture',       'perimeter',     'area',
        'smoothness',   'compactness',  'concavity',    'concave',       'symmetry',      'fractal',
        'feature_11',   'feature_12',   'feature_13',   'feature_14',    'feature_15',    'feature_16',
        'feature_17',   'feature_18',   'feature_19',   'feature_20',    'feature_21',    'feature_22',
        'feature_23',   'feature_24',   'feature_25',   'feature_26',    'feature_27',    'feature_28',
        'feature_29',   'feature_30')

# Print the first ten rows

head(data_3,n = 10)

# Collecting evidence for the question 'should the data be scaled?'

summary(data_3)

# Seperating columns from the original data to work with

data_4 = data_3[,3:32]

Y_wdbc = data_4

# Standardizing Breast Cancer Dataset

for (i in 1:30) {
```

```
  Y_wdbc[,i] = (Y_wdbc[,i]-mean(Y_wdbc[,i]))/sd(Y_wdbc[,i])
}
# Printing the first ten rows
head(Y_wdbc,n = 10)
# Printing summary after scaling
summary(Y_wdbc)
# Setting the seed so that results are reproducible
seed_val  <- 10
set.seed(seed_val)
# Selecting a number of clusters
k=5

# Running the k-means algorithm
first_clust_wdbc = kmeans(Y_wdbc, centers = 5, nstart = 1)
# How many patients are in each cluster?
first_clust_wdbc$size
# Setting the seed
seed_val <- 38
set.seed(seed_val)
# Selecting a number of clusters and run the k-means algorithm
second_clust_wdbc = kmeans(Y_wdbc, centers = 5, nstart = 1)
# How many patients are in each cluster?
second_clust_wdbc$size
# Adding cluster assignments to the data
Y_wdbc["first_clust_wdbc"] <- first_clust_wdbc$cluster
```

```
Y_wdbc["second_clust_wdbc"] <- second_clust_wdbc$cluster

# Printing the first ten rows

head(Y_wdbc,n = 10)

# Checking correlation

cor(Y_wdbc)

# Observing first characteristic of flowers

p_wdbc = cor(Y_wdbc)[,'radius']

p_wdbc[order(-p_wdbc),drop = FALSE]

# Creating the plot of radius and feature_30 for the first clustering algorithm

plot_one_wdbc  <- ggplot(Y_wdbc, aes(x = radius, y = feature_30, color = as.factor(first_clust_wdbc))) +
geom_point()

# Creating the plot of radius and feature_30 for the second clustering algorithm

plot_two_wdbc  <- ggplot(Y_wdbc, aes(x = radius, y = feature_30, color = as.factor(second_clust_wdbc)))
+ geom_point()

grid.arrange(plot_one_wdbc, plot_two_wdbc, ncol = 2)

# Executing hierarchical clustering with complete linkage

hier_clust_1_wdbc <- hclust(dist(Y_wdbc), method = "complete")

# Printing the dendrogram

plot(hier_clust_1_wdbc)

# Getting cluster assignments based on number of selected clusters

hc_1_assign_wdbc <- cutree(hier_clust_1_wdbc, k = 5)

# Executing hierarchical clustering with single linkage

hier_clust_2_wdbc <- hclust(dist(Y_wdbc), method = "single")

# Printing the dendrogram

plot(hier_clust_2_wdbc)

# Getting cluster assignments based on number of selected clusters
```

```
hc_2_assign_wdbc <- cutree(hier_clust_2_wdbc, k = 5)

# Adding assignment of chosen hierarchical linkage

Y_wdbc["hc_clust_wdbc"] <- hc_1_assign_wdbc

# Removing the first_clust_wdbc and second_clust_wdbc variables

hd_simple_wdbc <- Y_wdbc[,!(names(Y_wdbc) %in% c("first_clust_wdbc","second_clust_wdbc"))]

# Printing first 10 rows

head(hd_simple_wdbc, n = 10)

# Get the mean and standard deviation summary statistics

clust_summary_wdbc <- do.call(data.frame, aggregate(. ~ hc_clust_wdbc, data = hd_simple_wdbc,
function(x) c(avg = mean(x), sd = sd(x))))

clust_summary_wdbc

# Label Encoding

combined <- hd_simple_wdbc

encoded <- rep(0,length(data_3$Diagnosis))

for (i in 1:length(data_3$Diagnosis)) {

  if(data_3$Diagnosis[i] == 'B'){

    encoded[i] = 0

  }else{

    encoded[i] = 1

  }

}

encoded

combined["encoded"] <- encoded

# Printing first 10 rows

head(combined,n = 10)

# Fitting logistic regression
```

```
logistic_model <- glm(encoded ~.,family = binomial(link = 'logit'),data = combined)

summary(logistic_model)

# Ploting radius and feature_21

plot_one_wdbc_re_fe.21 <- ggplot(hd_simple_wdbc,aes(x = radius, y = feature_21, color =
as.factor(hc_clust_wdbc))) + geom_point()

# Ploting concavity and smoothness

plot_two_wdbc_con_sm<- ggplot(hd_simple_wdbc,aes(x = concavity, y = smoothness, color =
as.factor(hc_clust_wdbc))) + geom_point()

grid.arrange(plot_one_wdbc_re_fe.21, plot_two_wdbc_con_sm, ncol=2)

# Adding TRUE if the algorithm shows promise, adding FALSE if it does not

explore_kmeans_wdbc <- FALSE

explore_hierarch_complete_wdbc <- TRUE

explore_hierarch_single_wdbc <- FALSE

# PCA for Breast Cancer Dataset

# Removing the first_clust_wdbc, second_clust_wdbc and hc_clust_wdbc variables from Y_wdbc

Y_wdbc <- Y_wdbc[,!(names(Y_wdbc) %in%
c("first_clust_wdbc","second_clust_wdbc","hc_clust_wdbc"))]

# Printing first 10 rows

head(Y_wdbc,n = 10)

Y_wdbc = as.matrix(Y_wdbc)

# Executing this line to print all the rows of the dataset and printing sample PCAs

options(max.print = 1000000)

# Sample variance covariance matrix of iris dataset with working columns

Z = cov(Y_wdbc)

# Computing the eigenvalues and corresponding eigenvectors of Z

eigen(Z)

# Eigen-values of Z are
```

```
lambda = eigen(Z)$value

lambda

# Orthonormal eigen-vectors are (in columns)

v = eigen(Z)$vector

v

# Ordering of eigenvalues in descending order

order(lambda)

# Proportion of total variation explained by the principal components

for (i in lambda) {

  print(i / sum(lambda))

}# Scree Plot

plot(lambda, xlab = 'Eigenvalue Number', ylab = 'Eigenvalue Size', main = 'Scree Plot')

lines(lambda)

# Calculating sample PCAs from observed data matrix

for(i in 1:nrow(Y_wdbc)){

  X_pca_wdbc = crossprod(v[,1],Y_wdbc[i,])

  for (j in 2:5) {                              # Since 5 PCs explain a significant amount of variation

    X_pca_wdbc = cbind(X_pca_wdbc,crossprod(v[,j],Y_wdbc[i,]))

  }

  print(X_pca_wdbc)

}
```

Bibliography

1. Lecture notes of Dr. Amit Mitra, Professor, Department of Mathematics and Statistics, IIT Kanpur.

2. An Introduction to Statistical Learning with Applications in R- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

3. Wikipedia