CS-422 Assignment 4
Clustering, PCA

Name **:Dhayalini Nagaraj**

A ID **:A20359686**

I. Clustering
I. a. Use Iris data set

1. Install packages:

cluster
factoextra for visualizing clusters using ggplot2 plotting system
NbClust for finding the optimal number of clusters

```
if(!require(devtools)) install.packages("devtools")

devtools::install_github("kassambara/factoextra")

pkgs <- c("cluster", "NbClust")

install.packages(pkgs)

library(factoextra)

library(cluster)

library(NbClust)
```

The cluster, factoextra and NbClust packages are installed.

Cluster package is has methods for cluster analysis.
Factoextra:Extract and Visualize the Results of Multivariate Data Analyses
NbClust:It is used to determine the best number of clusters in the data set.

2. Prepare the data. Explain very briefly the output of these 3 commands. Don't repeat the comment, explain what it means and write in your own words.

```
> data(iris)

> head(iris)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species

1         5.1         3.5          1.4         0.2  setosa
```

2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Data preparation is the first step. **data ()** function is used to load specified data sets, or list the available data sets. To load a data frame in a package we use data() function.

data(iris) – loads iris dataset

head() function is used to obtain first several rows of a matrix or data frame.

head(iris) displays first several rows of iris data set.

Remove species column (5) and scale the data

iris.scaled <- scale(iris[, -5])

scale () function is generic function whose default method centers and/or scales the columns of a numeric matrix. The value of scale determines how column scaling is performed (after centering). If scale is a numeric vector with length equal to the number of columns of x, then each column of x is divided by the corresponding value from scale. If scale is TRUE then scaling is done by dividing the (centered) columns of x by their standard deviations if center is TRUE, and the root mean square otherwise. If scale isFALSE, no scaling is done.

Here the last column species is removed and the other columns of iris dataset is scaled.

3. Run the kmeans clustering. What number of clusters k will you use and why?

K means clustering is a Partitional clustering approach. Each cluster is associated with a centroid (center point) and each point is assigned to the cluster with the closest centroid. Number of clusters, K, must be specified.

The iris data set gives measurements of the variables sepal length, sepal width, petal length and petal width in centimeters for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica. Since there are 3 species in the Iris dataset we can set k as 3.

3 number of clusters can be used.

So each species can be be grouped as a cluster.

K-means clustering

> set.seed(123)

#We have set k as 3

> km.res <- kmeans(iris.scaled, 3, nstart = 25)

k-means group number of each observation

> km.res$cluster

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2

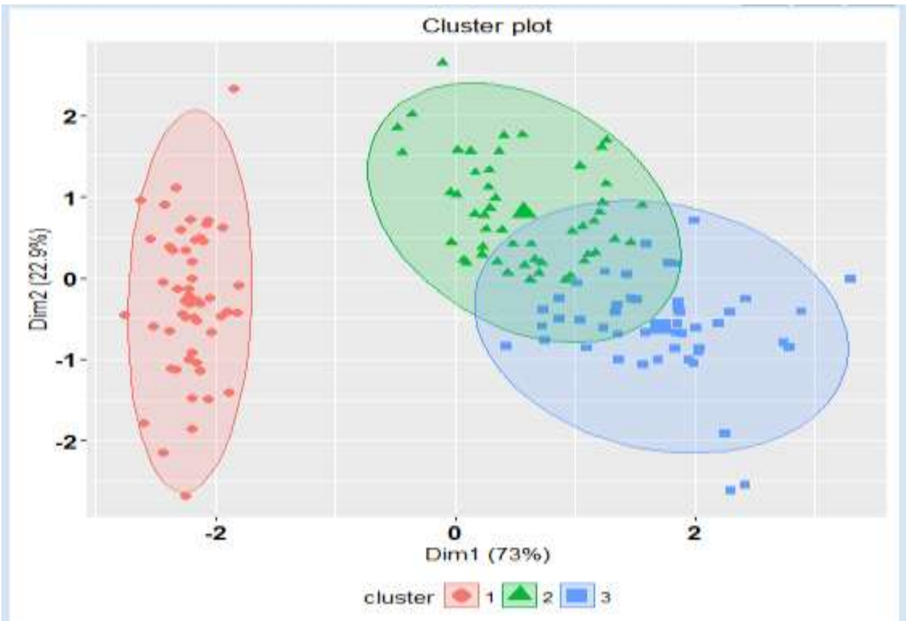
[75] 2 3 3 3 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3

[112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 3 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3

[149] 3 2

Visualize k-means clusters

fviz_cluster(km.res, data = iris.scaled, geom = "point", stand = FALSE, frame.type = "norm")



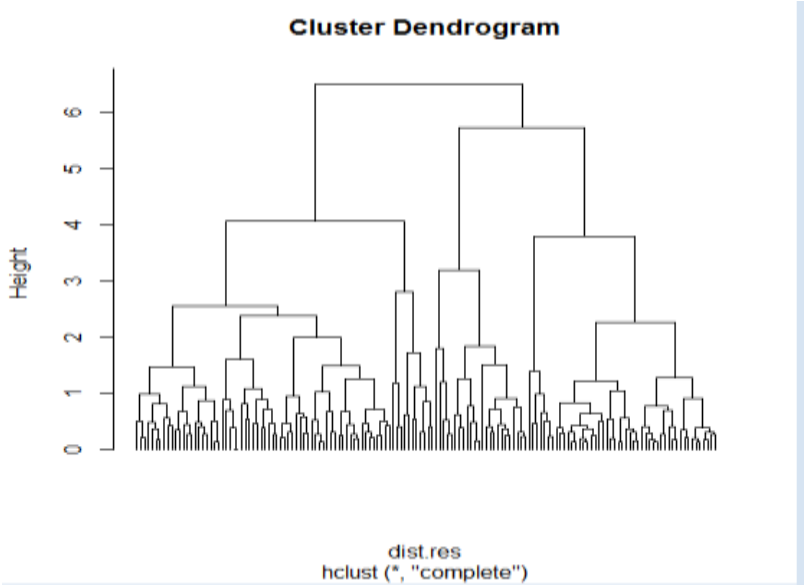
It visualizes the clustering result. fviz_cluster() It provides ggplot2-based elegant visualization of partitioning methods including kmeans. An ellipse is drawn around each cluster. The red green and blue represents each cluster of data.

4. Use the hierarchical clustering. Explain the results - how many levels of clusters do you see, what is the top cluster.

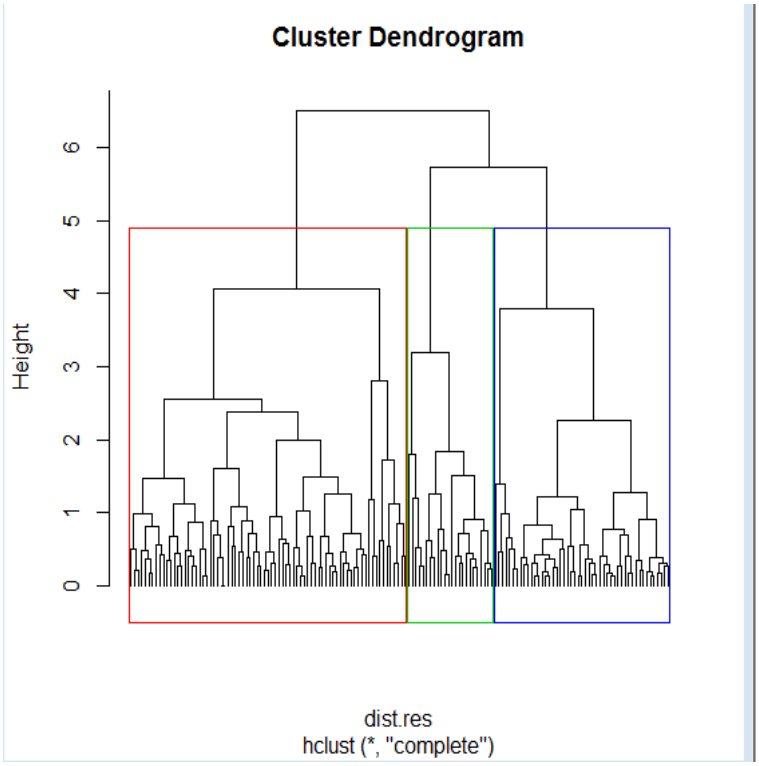
# Compute pairewise distance matrices
dist.res <- dist(iris.scaled, method = "euclidean")

# Hierarchical clustering results
hc <- hclust(dist.res, method = "complete")

# Visualization of hclust
plot(hc, labels = FALSE, hang = -1)



# Add rectangle around 3 groups
rect.hclust(hc, k = 3, border = 2:4)



The Hierarchical clustering produces a set of nested clusters organized as a hierarchical tree. It can be visualized as a dendrogram. A tree like diagram that records the sequences of merges or splits. By 'cutting' the dendogram at the proper level any desired number of clusters can be obtained.

The algorithm works as follows:
Put each data point in its own cluster.
Identify the closest two clusters and combine them into one cluster.
Repeat the above step till all the data points are in a single cluster.

Hierarchical clustering can be divided into two main types: **agglomerative** and **divisive**.

Agglomerative clustering: It's also known as AGNES (Agglomerative Nesting). It works in a bottom-up manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root) (see figure below). The result is a tree which can be plotted as a dendrogram.

Divisive hierarchical clustering: It's also known as DIANA (Divise Analysis) and it works in a top-down manner. The algorithm is an inverse order of AGNES. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster

According to the cluster dendogram there are 6 levels of cluster. Based on Agglomerative clustering the two most similar clusters are combined and form a big cluster. This is repeated till a single root cluster is formed. This root cluster is formed at the 6 level of cluster.

The top cluster is at level 5. Let us cut the dendogram at level 5. The level of cutting the dendogram at which the best level of similarity can be achieved is known as the top level. By cutting the dendogram at level 5, there are two clusters. By plotting the values of two clusters it can be found all the three species can be labelled clearly.

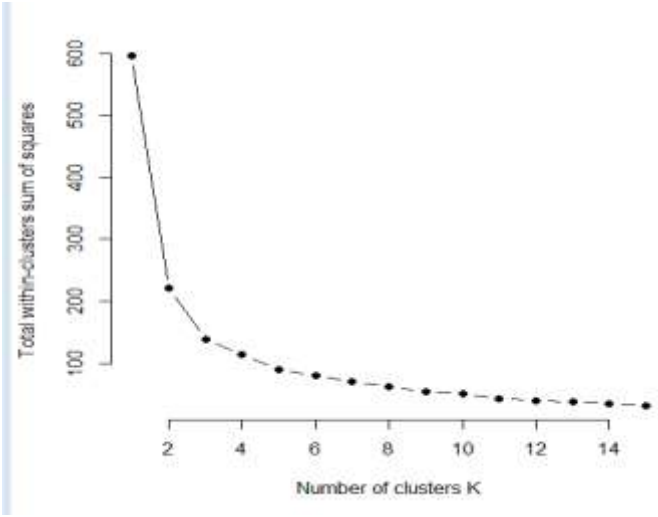
The k is set as 3. So the 3 clusters are highlighted at level 5 with three different colors. This indicates at level 5 the three species can be identified clearly. The 3 boxes represent the three species of the iris dataset.

5. Determine the best number of clusters.
5. b. Use the Elbow method as follows:

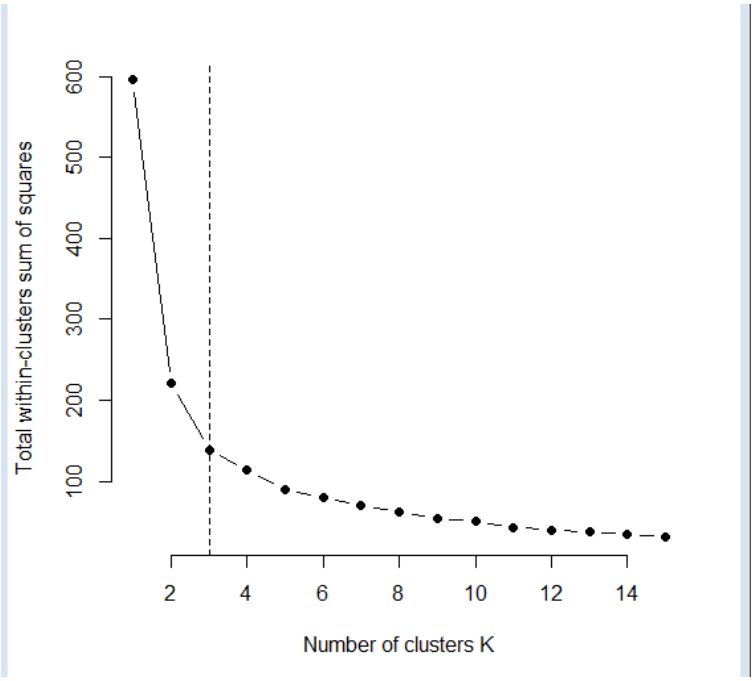
1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters
2. Give the definition of the wss measure
3. For each k, calculate the total within-cluster sum of square (wss)
4. Plot the curve of wss according to the number of clusters k.
5. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.
Explain how you will use the result to select the best number of clusters.

```
set.seed(123)
# Compute and plot wss for k = 2 to k = 15
k.max <- 15
# Maximal number of clusters
data <- iris.scaled
wss <- sapply(1:k.max, function(k){kmeans(data, k, nstart=10 )$tot.withinss})

plot(1:k.max, wss, type="b", pch = 19, frame = FALSE, xlab="Number of clusters K", ylab="Total
within-clusters sum of squares")
```



abline(v = 3, lty =2)



The oldest method for determining the true number of clusters in a data set is inelegantly called the elbow method.

The Elbow method looks at the percentage of variance explained as a function of the number of clusters. We should choose the number of cluster in such a way that adding a new cluster does not give much modelling of the data. If one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion".

The idea is that Start with K=2, and keep increasing it in each step by 1, calculating your clusters and the cost that comes with the training. At some value for K the cost drops dramatically, and after that it reaches a plateau when you increase it further. This is the K value you want.

This "elbow" cannot always be unambiguously identified. Percentage of variance explained is the ratio of the between-group variance to the total variance, also known as an F-test. A slight variation of this method plots the curvature of the within group variance.

Run k-means clustering on the dataset for a range of values of *k* (say, *k* from 1 to 10 in the examples above), and for each value of *k* calculate the sum of squared errors (SSE). Then, plot a line chart of the SSE for each value of *k*. If the line chart looks like an arm, then the "elbow" on the arm is the value of *k* that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase *k*. So our goal is to choose a small value of *k* that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing *k*.

The basic idea behind partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation (known as total within-cluster variation or total within-cluster sum of square) is minimized:

Minimize $(\sum_{k=1}^{k} W(C_k))$ Where Ck is the kth cluster and W(Ck) is the within-cluster variation.

The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible

The within-cluster sum of squares is: $\sum_{k=1}^{k} \sum_{i \epsilon S_k} \sum_{j=1}^{p} (x_{ij} - \sim x_{kj})^2$ where S_k is the set of observatioins in the kth cluster and $\sim x_{kj}$ is the j th variable of the cluster center for the k th cluster.

Each observation is allocated to the closest cluster, and the distance between an observation and a cluster is calculated from the Euclidean distance between the observation and the cluster center.Each cluster center will then be updated as the mean for observations in each cluster.

We perform this exercise in a loop to find updated cluster centers and allocation of each observation. The iteration will stop when the maximum number of iterations is reached or the change of within-cluster sum of squares in two successive iterations is less than the threshold value. The updated cluster centers for the last iteration are called Final Cluster Centers

Graph : In this graph, the k- means clustering algorithm is done for various k values and wss is calculated for each k values. The graph is plotted according to the wss and k value. When the k value is increased the wss value is decreased. When k value is set as 2 the wss value is high and when k value is increased the wss drops from high level. By plotting the graph for various k values we can find a bend in the graph where the marginal gain is dropped. The location of a bend(knee) represents or indicates the appropriate number of clusters. According to the graph the elbow method suggests 3 clusters. Therefore the best number of clusters is 3. The R function abline() is be used to add vertical line at k = 3 to show the appropriate number of clusters.

5.c. Use the nbclust to determine the number of clusters. Explain briefly how this approach determines the best number of clusters.

Clustering is the partitioning of a set of objects into groups (clusters) so that objects within a group are more similar to each others than objects in different groups. A wide variety of indices have been proposed to find the optimal number of clusters in a partitioning of a data set during the clustering process.

The R package NbClust is developed for that purpose. It provides 30 indices which is used to determine the number of clusters in a data set and it offers also the best clustering scheme from different results to the user. In addition, it provides a function to perform k-means and hierarchical clustering with different distance measures and aggregation methods. Any

combination of validation indices and clustering methods can be requested in a single function call. This enables the user to simultaneously evaluate several clustering schemes while varying the number of clusters, to help determining the most appropriate number of clusters for the data set of interest

In the NbClust package, the knee is detected by a local peak in the plot of second differences between levels of the index. Thus, the suitable number of clusters is chosen by visual inspection of the second differences plot. The absence of such a knee might be an indication that the data set possesses no clustering structure.

In the NbClust package, validity indices can be applied to outputs of two clustering algorithms: k-means and hierarchical agglomerative clustering (HAC), by varying all combinations of number of clusters, distance measures and clustering methods.

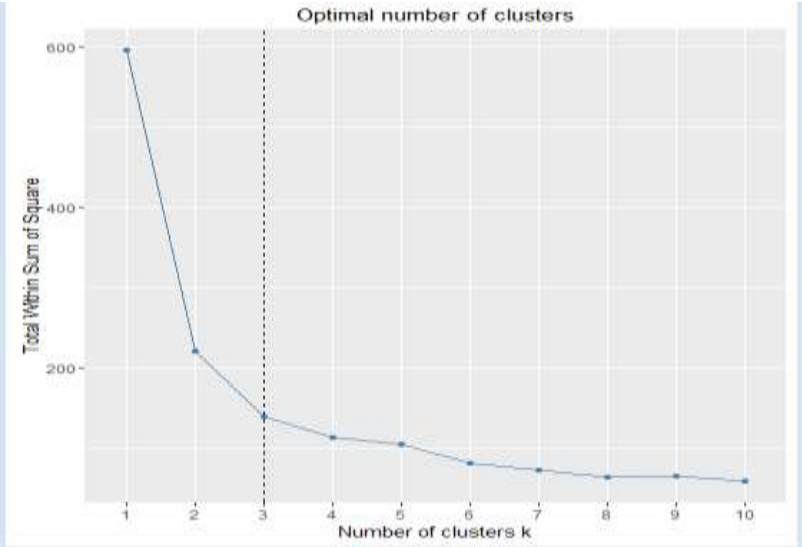
One important benefit of NbClust is that the user can simultaneously select multiple indices and number of clusters in a single function call. Moreover, it offers the user the best clustering scheme from different results. The simplified format of the function NbClust() is:

NbClust(data = NULL, diss = NULL, distance = "euclidean", min.nc = 2, max.nc = 15, method = NULL, index = "all")

- data: matrix
- diss: dissimilarity matrix to be used. By default, diss=NULL, but if it is replaced by a dissimilarity matrix, distance should be “NULL”
- distance: the distance measure to be used to compute the dissimilarity matrix. Possible values include “euclidean”, “manhattan” or “NULL”.
- min.nc, max.nc: minimal and maximal number of clusters, respectively
- method: The cluster analysis method to be used including “ward.D”, “ward.D2”, “single”, “complete”, “average” and more
- index: the index to be calculated including “silhouette”, “gap” and more.

For kmeans: Explain what you see on the plot and how you will use it for determining the best number of clusters

fviz_nbclust(iris.scaled, kmeans, method = "wss") + geom_vline(xintercept = 3, linetype = 2).

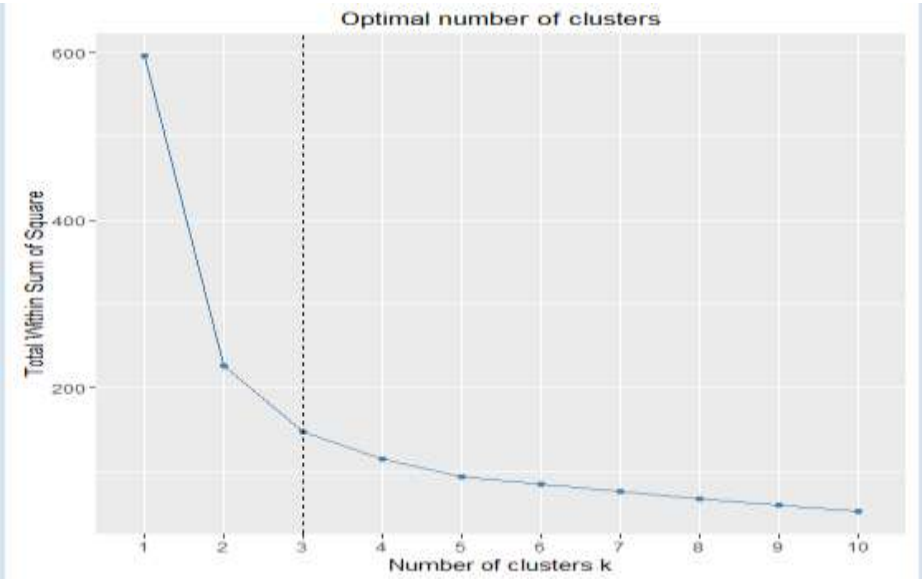


k-means is an iterative method which minimizes the within-class sum of squares for a given number of clusters. The algorithm starts with an initial guess for cluster centers, and each observation is placed in the cluster to which it is closest. The cluster centers are then updated, and the entire process is repeated until the cluster centers no longer move. k-means is said to be a reallocation method.
The general principle:
1. Select as many points as the number of desired clusters to create initial centers.
2. Each observation is then associated with the nearest center to create temporary clusters.
3. The gravity centers of each temporary cluster are calculated and these become the new cluster centers.
4. Each observation is reallocated to the cluster which has the closest center.
5. This procedure is iterated until convergence.

In this graph the iris scaled data frame is used. The elbow method for kmeans() function is used. It is similar to elbow method where kmeans partition function is used. The k values are varied from 1 to 10 and wss values are calculated for each k values. The values are plotted in the graph according to k means clustering. The wss value is decreased for every iteration of k value. From observation, there is a marginal gain drop in the graph at a particular cluster value. This k value is considered as the appropriate cluster value. The location of the bend(knee) indicates the number of clusters.
This method suggest 3 cluster solution.

For hierarchical clustering: Explain what you see on the plot and how you will use it for determining the best number of clusters.

fviz_nbclust(iris.scaled, hcut, method = "wss") + geom_vline(xintercept = 3, linetype = 2)



Hierarchical clustering seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- Agglomerative or “bottom up” approach where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive or “top down” approach where all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram. Hierarchical clustering requires to define a dissimilarity measure (or distance) and an agglomeration criterion. Many distances are available as well as several agglomeration methods.

In this, nbclust uses iris scaled data frame. The elbow method is used for hierarchical clustering. The helper function hcut() is used to compute hierarchical clustering (HC) algorithm and cut the dendrogram in k clusters. The cluster values are varied from 1 to 10 and hierarchical clustering is done. For each k value wss is also calculated. According to the k values and wss calculated in hierarchical clustering the values are plotted in the graph. Since it is an elbow method, we can find the elbow curve from k =3 to 10. The wss values decrease for increase in k. But after certain k value there is no big difference in wss. We can also find a bend in the graph where the marginal gain is dropped. This particular point, the bend(knee) denotes the appropriate number of clusters. This method also gives the optimal number of clusters as 3.

I.b. Use Wine data set
1. The wine data set is from the UCI ML repository, the rattle package will install it for you.
2. Use the rattle package in R to load the data
install.packages("rattle")

library("rattle")

Run the following commands, very briefly explain the output. Analyze the data: number of instances, dimensionality etc.

> data(wine, package="rattle")

> head(wine)

	Type	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids
1	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28
2	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26
3	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30
4	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24
5	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39
6	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34

	Proanthocyanins	Color	Hue	Dilution	Proline
1	2.29	5.64	1.04	3.92	1065

2 1.28 4.38 1.05 3.40 1050

3 2.81 5.68 1.03 3.17 1185

4 2.18 7.80 0.86 3.45 1480

5 1.82 4.32 1.04 2.93 735

6 1.97 6.75 1.05 2.85 1450

> df <- scale(wine[-1])

Wine Data Set: Using chemical analysis determine the origin of wines

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. Three types of wine are represented in the178 instances. The type variable has been transformed into a categoric variable. The dataset characteristic is multivariate and the attribute characteristics are Integer and Real.

Data Set Characteristics	Multivariate
Attribute Characteristics	Integer, Real
Number of Instances	178
Number of attributes	13
Missing Values	No
Associated Tasks	Classification
Area	Physical
Date Donated	1991-07-01
Number of Web Hits	594085

The attributes are
1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

Data preparation is the first step. **data()** is used to load the data.The wine data frame is loaded from the rattle package.

data(wine, package="rattle") – loads wine data frame.

head() function is used to obtain first several rows of a matrix or data frame.

head(wine) displays first several rows of wine data set.

df <- scale(wine[-1])

Wine data set contains 1 categorical variables (label) and 13 numerical variables. But these numerical variables is not scaled, so we use scale function for scaling and centering data.

3. Use NbClust, explain how you will use the result.

```
> library(NbClust)
> set.seed(1234)
> nc <- NbClust(df, min.nc=2, max.nc=15, method="kmeans")
```

*** : The Hubert index is a graphical method of determining the number of clusters.

In the plot of Hubert index, we seek a significant knee that corresponds to a significant increase of the value of the measure i.e the significant peak in Hubert index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.

In the plot of D index, we seek a significant knee (the significant peak in Dindex second differences plot) that corresponds to a significant increase of the value of the measure.

```
*******************************************************************
* Among all indices:
* 4 proposed 2 as the best number of clusters
* 15 proposed 3 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 1 proposed 12 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters
```

***** Conclusion *****

* According to the majority rule, the best number of clusters is 3

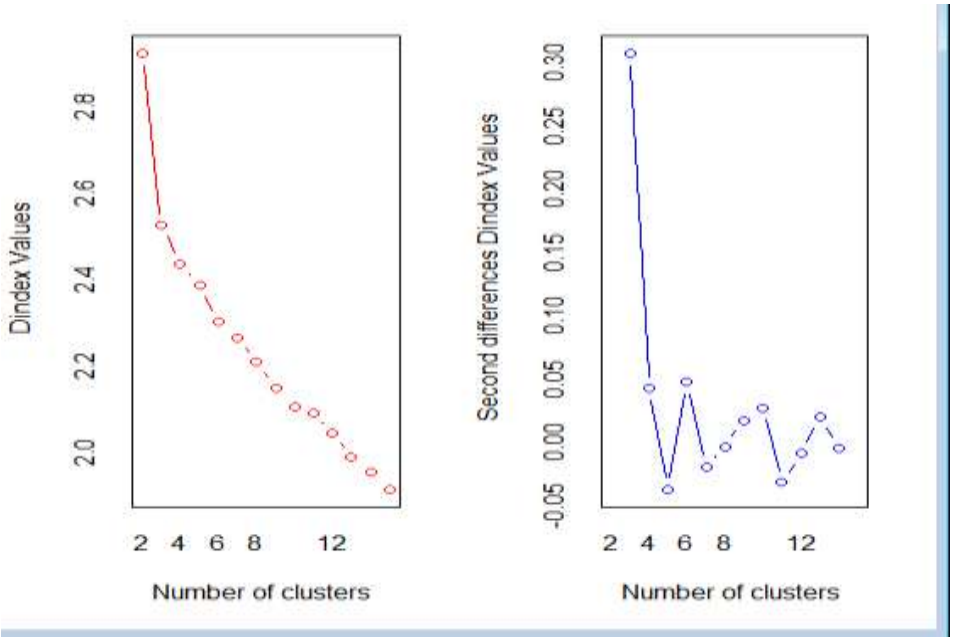
****

>> table(nc$Best.n[1,])

0 1 2 3 10 12 14 15

2 1 4 15 1 1 1 1

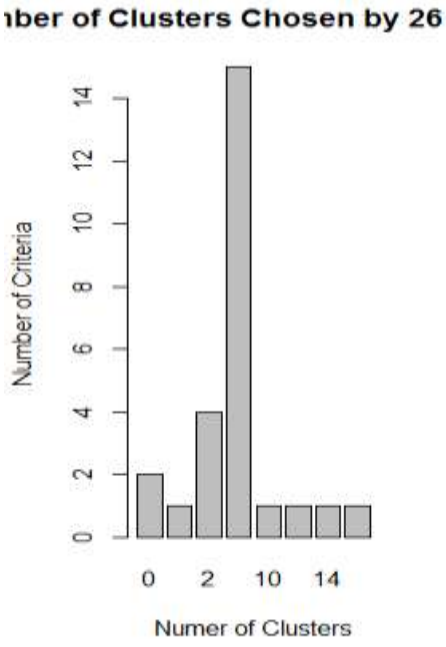
In this Hubert index and D index graphical method is used to determine the number of clusters using the kmeans clustering function. Using both indices, the majority rule is the best number of cluster is 3. Using both the index it is found that the majority has proposed the best number as 3 where as the other proposed number of cluster is also displayed. When the Hubert and D index values are plotted a significant knee can be identified that corresponds to a significant increase of the value of the measure.



The table function is used to display the nBClust result.

The first row in the table denotes the number of clusters and the second row in the table denotes the number of criteria. From the table it is seen that 14 of 24 criteria provided by the NbClust package suggest a 3-cluster solution. This is also seen from the following bar graph.

> barplot(table(nc$Best.n[1,]), xlab="Numer of Clusters", ylab="Number of Criteria", main="Number of Clusters Chosen by 26 Criteria")



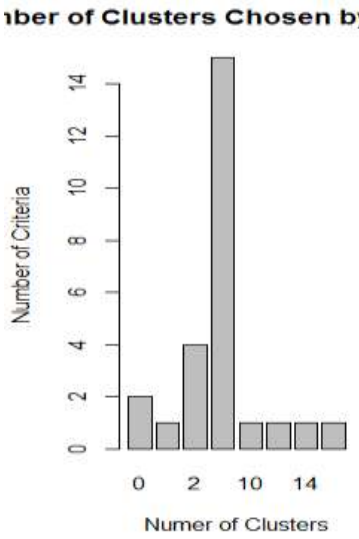
The graph denotes Recommended number of clusters using 26 criteria provided by the NbClust package.

Note that not all 30 criteria can be calculated for every dataset.

Thus using the NbClust function the best number of cluster is found as 3 and this cluster value can be used for further analysis.

4. Run the kmeans clustering using the number of clusters you get after NbClust.

> barplot(table(nc$Best.n[1,]), xlab="Numer of Clusters", ylab="Number of Criteria", main="Number of Clusters Chosen by 26 Criteria")



The graph denotes Recommended number of clusters using 26 criteria provided by the NbClust package. It is seen that 14 of 24 criteria provided by the NbClust package suggest a 3-cluster solution.

```
> set.seed(1234)

> fit.km <- kmeans(df, 3, nstart=25)

> fit.km$size

[1] 51 65 62

> fit.km$centers

    Alcohol      Malic       Ash Alcalinity   Magnesium     Phenols

1  0.1644436  0.8690954  0.1863726  0.5228924 -0.07526047 -0.97657548

2 -0.9234669 -0.3929331 -0.4931257  0.1701220 -0.49032869 -0.07576891

3  0.8328826 -0.3029551  0.3636801 -0.6084749  0.57596208  0.88274724

  Flavanoids Nonflavanoids Proanthocyanins      Color       Hue   Dilution

1 -1.21182921    0.72402116    -0.77751312  0.9388902 -1.1615122 -1.2887761

2  0.02075402   -0.03343924     0.05810161 -0.8993770  0.4605046  0.2700025

3  0.97506900   -0.56050853     0.57865427  0.1705823  0.4726504  0.7770551

    Proline

1 -0.4059428

2 -0.7517257

3  1.1220202

> aggregate(wine[-1], by=list(cluster=fit.km$cluster), mean)

  cluster  Alcohol    Malic      Ash Alcalinity Magnesium  Phenols Flavanoids

1       1 13.13412 3.307255 2.417647   21.24118  98.66667 1.683922  0.8188235

2       2 12.25092 1.897385 2.231231   20.06308  92.73846 2.247692  2.0500000

3       3 13.67677 1.997903 2.466290   17.46290 107.96774 2.847581  3.0032258

  Nonflavanoids Proanthocyanins    Color       Hue Dilution   Proline

1     0.4519608       1.145882 7.234706 0.6919608 1.696667  619.0588

2     0.3576923       1.624154 2.973077 1.0627077 2.803385  510.1692

3     0.2920968       1.922097 5.453548 1.0654839 3.163387 1100.2258
```

The best cluster found from the NbClust function is 3. So we use k=3 for kmeans() function. Once we have got the number of clusters, run k-means using this number of clusters. Output the result of calling kmeans() into a variable fit.km. The kmeans() function has an nstart option that attempts multiple initial configurations and reports on the best one. For example, adding nstart=25 will generate 25 initial configurations. This approach is often recommended.

fit.km$size displays the size of the 3 clusters 51, 65, 62.

fit.km$centers The cluster centroids are displayed.

A final cluster solution is obtained with kmeans() function and the cluster centroids are printed. Since the centroids provided by the function are based on standardized data, the aggregate() function is used along with the cluster memberships to determine variable means for each cluster in the original metric.

aggregate(wine[-1], by=list(cluster=fit.km$cluster), mean) is used to determine the variable means for each cluster in the original metric.

Analyze the confusion matrix, explain what are the entries and what do they mean for the analysis of the quality of clusters.

> ct.km <- table(wine$Type, fit.km$cluster)

> ct.km

	1	2	3
1	0	0	59
2	3	65	3
3	48	0	0

So to know how the K-means clustering uncover the actual structure of the data contained in the Type variable, a cross-tabulation of Type (wine varietal) and cluster membership is given by
> ct.km <- table(wine$Type, fit.km$cluster)

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand.

Here the confusion matrix is displayed based on the wine type and the clusters. According to the kmeans() function the confusion matrix is displayed.

The entries in the confusion matrix are instances. In wine data set there are 178 instances and they are classified as follows.

It is seen that 3 instances of 2nd type wine is classified as 1st type and 3 instances of 2nd type wine is classified as 3rd type wine.

Based on this 6 samples are missed.

Analyzing the confusion matrix we can say something about the quality of the clusters also. There are three clusters. The first cluster has 3 samples which are not classified correctly and the third cluster also has 3 samples which are misclassified. Whereas the second cluster has all the samples which belong to the same wine type. Though 6 samples are missing, the quality of the second cluster is good when compared to the other two clusters.

The index values can also be calculated inorder to find the quality of the clusters.

Based on the confustion matrix the clustering can be considered as good.

Read about Rank Index, explain how you can use to evaluate the clustering solution. Explain the attribute wine.type and how it is used for rank index.

Install flexclust package:

```
> library(flexclust)

Loading required package: grid

Loading required package: lattice

Loading required package: modeltools

Loading required package: stats4

> randIndex(ct.km)

    ARI

0.897495
```

The Rand index or Rand measure is a measure of the similarity between two data clusterings. Adjusted Rand Index is a form of the Rand index may be defined that is adjusted for the chance grouping of elements.Rand index is related to the accuracy, but is applicable even when class labels are not used.

Given a set of n elements S={O1,....,On} and two partitions of S to compare X={X1, ...,Xn}, a partition of S into r subsets and Y={Y1,....,Yn} a partition of S into s subsets define the following.

a, the number of pairs of elements in S that are in the same set in X and in the same set in Y.

b, the number of pairs of elements in S that are in different sets in X and in different sets in Y

c, the number of pairs of elements in S that are in the same set in X and in different sets in Y

d, the number of pairs of elements in S that are in different sets in X and in same set in Y

The Rand Index is a+b/a+b+c+d = a+b/ nC2.

Intuitively, a+b can be considered as the number of agreements between X and Y and c+d as the number of disagreements between X and Y.

The adjusted Rand index is the corrected for chance version of the Rand index. Though the Rand Index may only yield a value between 0 and +1, the adjusted Rand index can yield negative values if the index is less than the expected index.

Evaluation of a cluster:

Typical objective functions in clustering formalize the goal of attaining high intra-cluster similarity and low inter-cluster similarity. This is an internal critlse neerion for the quality of a clustering.

Rand index penalizes both false positive and false negative decisions during clustering.
The F measure in addition supports differential weighting of these two types of errors.

A true positive (TP) decision assigns two similar documents to the same cluster, a true negative (TN) decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit. A (FP) decision assigns two dissimilar documents to the same cluster. A (FN) decision assigns two similar documents to different clusters. The Rand index ( ) measures the percentage of decisions that are correct. That is, it is simply accuracy

RI = TP+ TN/ TP+TN+FP+FN

The Rand index gives equal weight to false positives and false negatives. Separating similar documents is sometimes worse than putting pairs of dissimilar documents in the same cluster. We can use the F measure to penalize false negatives more strongly than false positives by selecting a value $\beta>1$, thus giving more weight to recall.

Since the Rand Index gives equal weight to false positives and false negatives, evaluation of clusters is done by this.

The adjusted Rand index provides a measure of the agreement between two partitions, adjusted for chance. It ranges from -1 (no agreement) to 1 (perfect agreement). The wine data set has three classes. Each wine type is formed as a cluster group. Based on the classification of wine type the evaluation of clustering should be done. The measure of agreement between the wine.type and the cluster ssolution decides the quality of cluster. Agreement between the wine varietal type and the cluster solution is 0.9 . 0.9 is nearly equal to 1. So this can be considered as a good agreement and the quality of cluster is also good.

II PCA

II.1 Delta dataset

1. Get the data delta.csv. Explain the data: number of data points, number of attributes, types of attributes.

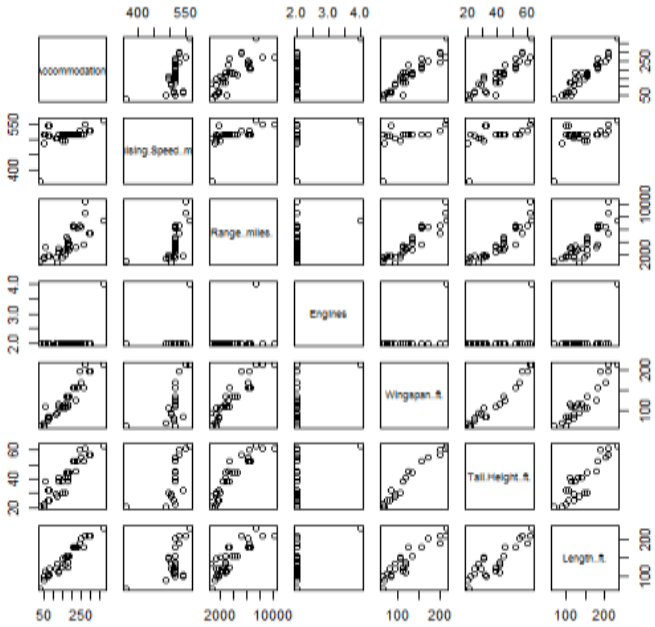
The delta data set comprises 33 variables on 44 aircraft taken from Delta.com website. It includes both quantitative measures on attributes like cruising speed, accommodation and range in miles, as well as categorical data such as whether a particular aircraft has Wi-Fi or video. These binary categorical variables were transformed into quantitative variables by assigning them values of either 1 or 0, for yes or no respectively.
So the delta dataset has 33 datapoints, 44 attributes and both quantitative and categorical types.

2. Analyze the variance in the attributes. Explain what you see in the plots.

> data <- read.csv(file="delta.csv", header=T, sep=",", row.names=1)

scatterplot matrix of intermediary (size/non-categorical) variables

> plot(data[,16:22])



When we look at the intermediary quantitative variables which are related to the aircraft physical characteristics like cruising speed, total accommodation, and other quantities like length and wingspan,these variables are about in the middle of the data frame.

So all of them can be visualized at once using a scatterplot matrix, which is the default for R's output if plot is called on a dataframe.

There are strong positive correlations between all these variables, as all of them are related to the aircraft's overall size. There is an almost perfectly linear relationship between wingspan and tail height.

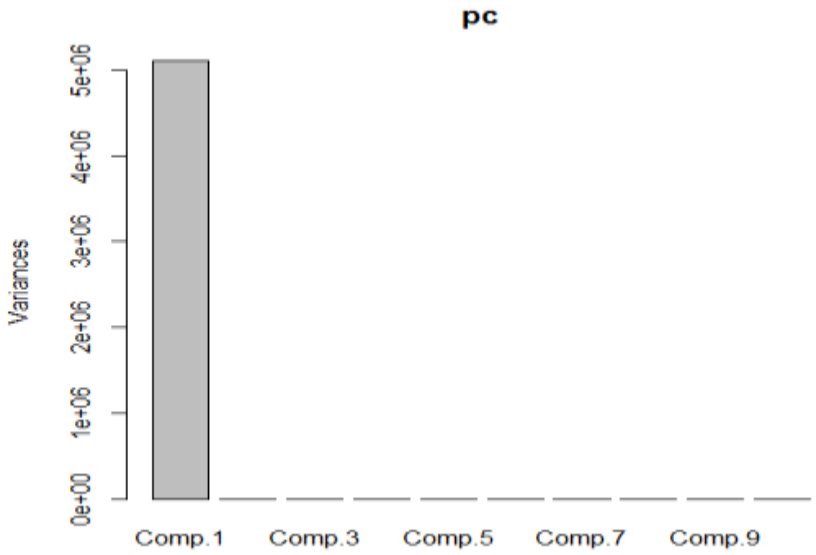
The exception here is the variable right in the middle which is the number of engines. There is one lone outlier [Boeing 747-400 (74S)] which has four, while all the other aircraft have two. In this way the engines variable is really more like a categorical variable.

3. Use PCA on raw data. Explain what you see in the chars. How many important principle components do you see?

apply principal components analysis to raw data and plot

> pc <- princomp(data)

> plot(pc)



From the graph we can see that the first principal component has a standard deviation of around 2200 and accounts for over 99.8% of the variance in the data. Looking at the first column of loadings shows that the first principle component is just the range in miles. The first principle component variance is more whereas the other components are just nil. The first component is dominant.

4. Analyze the data, explain what the comments mean in your own words.

> summary(pc) # 1 component has > 99% variance

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4
Standard deviation	2259.2372556	6.907940e+01	2.871764e+01	2.259929e+01
Proportion of Variance	0.9987016	9.337038e-04	1.613651e-04	9.993131e-05
Cumulative Proportion	0.9987016	9.996353e-01	9.997966e-01	9.998966e-01

	Comp.5	Comp.6	Comp.7	Comp.8
Standard deviation	1.482962e+01	1.049014e+01	9.1522288972	7.937495e+00
Proportion of Variance	4.303007e-05	2.153154e-05	0.0000163895	1.232761e-05
Cumulative Proportion	9.999396e-01	9.999611e-01	0.9999775129	9.999898e-01

	Comp.9	Comp.10	Comp.11	Comp.12
Standard deviation	4.523039e+00	3.623724e+00	2.606872e+00	1.929074e+00
Proportion of Variance	4.002882e-06	2.569344e-06	1.329693e-06	7.281316e-07
Cumulative Proportion	9.999938e-01	9.999964e-01	9.999977e-01	9.999985e-01

```
                       Comp.13     Comp.14     Comp.15     Comp.16
Standard deviation     1.760506e+00 1.563002e+00 1.245856e+00 4.772154e-01
Proportion of Variance 6.064390e-07 4.780036e-07 3.037023e-07 4.455956e-08
Cumulative Proportion  9.999991e-01 9.999996e-01 9.999999e-01 9.999999e-01
                       Comp.17     Comp.18     Comp.19     Comp.20
Standard deviation     3.806455e-01 3.493458e-01 2.724929e-01 2.153123e-01
Proportion of Variance 2.835002e-08 2.387939e-08 1.452856e-08 9.070884e-09
Cumulative Proportion  9.999999e-01 1.000000e+00 1.000000e+00 1.000000e+00
                       Comp.21     Comp.22     Comp.23     Comp.24
Standard deviation     1.991243e-01 1.669167e-01 1.340994e-01 1.209009e-01
Proportion of Variance 7.758194e-09 5.451444e-09 3.518566e-09 2.860034e-09
Cumulative Proportion  1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
                       Comp.25     Comp.26     Comp.27     Comp.28
Standard deviation     6.524198e-02 4.241346e-02 2.373915e-02 2.016179e-03
Proportion of Variance 8.328491e-10 3.519811e-10 1.102662e-10 7.953717e-13
Cumulative Proportion  1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
                       Comp.29 Comp.30 Comp.31 Comp.32 Comp.33
Standard deviation     2.452124e-05       0       0       0       0
Proportion of Variance 1.176513e-16       0       0       0       0
Cumulative Proportion  1.000000e+00       1       1       1       1

> loadings(pc)  #Can see all variance is in the range in miles


Loadings:
                      Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
Seat.Width..Club.                            -0.144 -0.110
Seat.Pitch..Club.                            -0.327 -0.248
Seat..Club.
```

Seat.Width..First.Class.	0.250	-0.160	-0.156			
Seat.Pitch..First.Class.	0.515	-0.110	-0.386	0.112	-0.130	
Seats..First.Class.	0.258	-0.124	-0.307	-0.109	0.160	
Seat.Width..Business.	-0.154	0.142	-0.108			
Seat.Pitch..Business.	-0.514	0.446	-0.298	0.154	-0.172	
Seats..Business.	-0.225	0.187				
Seat.Width..Eco.Comfort.	0.285	-0.224				
Seat.Pitch..Eco.Comfort.	0.159	0.544	-0.442			
Seats..Eco.Comfort.	0.200	-0.160				
Seat.Width..Economy.	0.125	0.110				
Seat.Pitch..Economy.	0.227	0.190				
Seats..Economy.	0.597	-0.136	0.345	-0.165		
Accommodation	0.697	-0.104				
Cruising.Speed..mph.	0.463	0.809	0.289	-0.144	0.115	
Range..miles.	0.999					
Engines						
Wingspan..ft.	0.215	0.103	-0.316	-0.357	-0.466	
Tail.Height..ft.	-0.100	-0.187				
Length..ft.	0.275	0.118	-0.318	0.467	0.582	
Wifi						
Video						
Power						
Satellite						
Flat.bed						
Sleeper						
Club						
First.Class						
Business						

Eco.Comfort

Economy

	Comp.8	Comp.9	Comp.10	Comp.11	Comp.12	Comp.13	Comp.14
Seat.Width..Club.			-0.165				
Seat.Pitch..Club.	0.189		-0.374	0.121	0.174		
Seat..Club.			-0.102				
Seat.Width..First.Class.	0.136		-0.246	0.341	-0.128	0.429	
Seat.Pitch..First.Class.	0.183	0.161	-0.307	0.211	-0.389		
Seats..First.Class.	0.149	0.313	0.172	-0.242	-0.659		
Seat.Width..Business.			0.244	-0.480	0.255	-0.232	
Seat.Pitch..Business.	0.379	0.285	0.401				
Seats..Business.	-0.287	-0.608	-0.294	-0.133	-0.503	-0.294	
Seat.Width..Eco.Comfort.			-0.113	0.111			
Seat.Pitch..Eco.Comfort.			-0.268	0.260	0.120		
Seats..Eco.Comfort.	-0.208	0.318	-0.733	0.156			
Seat.Width..Economy.			0.186	-0.110			
Seat.Pitch..Economy.	-0.130		0.262	-0.104	-0.132		
Seats..Economy.	0.168	0.597	-0.205	-0.127			
Accommodation	0.233	-0.592	0.183	0.153	0.152		
Cruising.Speed..mph.							
Range..miles.							
Engines							
Wingspan..ft.	-0.665		0.106	0.137			
Tail.Height..ft.		0.441	0.100	-0.698	0.299		
Length..ft.	-0.418	0.203			0.132		
Wifi							
Video							
Power							

Satellite

Flat.bed

Sleeper

Club

First.Class

Business

Eco.Comfort

Economy

	Comp.15	Comp.16	Comp.17	Comp.18	Comp.19	Comp.20
Seat.Width..Club.			0.105		0.114	
Seat.Pitch..Club.	0.119	0.239	0.153	0.259	0.161	0.197
Seat..Club.						
Seat.Width..First.Class.	0.371	-0.210	0.204	-0.152		-0.310
Seat.Pitch..First.Class.	-0.424					
Seats..First.Class.	0.361					
Seat.Width..Business.	-0.393	-0.229	0.317			-0.287
Seat.Pitch..Business.						
Seats..Business.						
Seat.Width..Eco.Comfort.		0.546	0.148	0.238		
Seat.Pitch..Eco.Comfort.		-0.124	-0.266			0.223
Seats..Eco.Comfort.	0.437	-0.107				
Seat.Width..Economy.		0.427		0.268	-0.205	-0.257
Seat.Pitch..Economy.		-0.165	0.577		0.512	0.119
Seats..Economy.						
Accommodation						
Cruising.Speed..mph.						
Range..miles.						
Engines		-0.118	-0.160		-0.144	0.522

Wingspan..ft.	0.107				
Tail.Height..ft.	-0.380				
Length..ft.					
Wifi	0.196	-0.112	0.133	0.182	
Video	-0.156	-0.436	-0.362		
Power	0.450	-0.512	0.117		
Satellite	0.244	-0.473	0.200	0.386	
Flat.bed	0.295	-0.574	0.103		
Sleeper	-0.449	0.201	0.480	-0.154	
Club					
First.Class					
Business					
Eco.Comfort					
Economy					

	Comp.21	Comp.22	Comp.23	Comp.24	Comp.25	Comp.26
Seat.Width..Club.	-0.107					
Seat.Pitch..Club.	-0.243	-0.123	0.156	-0.198		
Seat..Club.						
Seat.Width..First.Class.	0.213	-0.219	-0.113			
Seat.Pitch..First.Class.						
Seats..First.Class.						
Seat.Width..Business.	0.196	-0.114	-0.221			
Seat.Pitch..Business.						
Seats..Business.						
Seat.Width..Eco.Comfort.	-0.109	0.603	-0.229			
Seat.Pitch..Eco.Comfort.	-0.159	0.123	0.167	-0.277	0.126	
Seats..Eco.Comfort.						
Seat.Width..Economy.	-0.309	-0.205	-0.122	-0.609		

Seat.Pitch..Economy.	-0.169	0.184	-0.225	0.107	-0.120
Seats..Economy.					
Accommodation					
Cruising.Speed..mph.					
Range..miles.					
Engines	0.309	-0.153	-0.683	-0.127	-0.244
Wingspan..ft.					
Tail.Height..ft.					
Length..ft.					
Wifi	0.401	-0.574	0.282	0.554	
Video	-0.506	-0.289	-0.368	0.297	-0.238
Power	0.408	0.506	-0.158	0.183	0.139
Satellite	-0.220	-0.238	0.285	-0.519	-0.169	-0.128
Flat.bed	0.199	0.136	0.264	-0.123	-0.469
Sleeper	0.164	-0.133	-0.426		
Club					
First.Class	-0.128	-0.163	0.172	0.402	
Business	0.134	0.167	-0.114	-0.420	
Eco.Comfort					
Economy					

	Comp.27	Comp.28	Comp.29	Comp.30	Comp.31	Comp.32
Seat.Width..Club.	0.791					
Seat.Pitch..Club.	-0.460					
Seat..Club.	0.402					
Seat.Width..First.Class.						
Seat.Pitch..First.Class.						
Seats..First.Class.						
Seat.Width..Business.	-0.131					

Seat.Pitch..Business.

Seats..Business.

Seat.Width..Eco.Comfort.

Seat.Pitch..Eco.Comfort.

Seats..Eco.Comfort.

Seat.Width..Economy.

Seat.Pitch..Economy.

Seats..Economy.

Accommodation

Cruising.Speed..mph.

Range..miles.

Engines

Wingspan..ft.

Tail.Height..ft.

Length..ft.

Wifi 0.123

Video 0.136

Power

Satellite -0.132

Flat.bed -0.435

Sleeper -0.508

Club 0.591 0.587 -0.551

First.Class -0.473 -0.417 -0.439 -0.360

Business 0.495 -0.396 -0.439 -0.360

Eco.Comfort -0.813 0.439 0.360

Economy 0.789 -0.277 0.546

Comp.33

Seat.Width..Club. -0.470

Seat.Pitch..Club.

Seat..Club. 0.878

Seat.Width..First.Class.

Seat.Pitch..First.Class.

Seats..First.Class.

Seat.Width..Business.

Seat.Pitch..Business.

Seats..Business.

Seat.Width..Eco.Comfort.

Seat.Pitch..Eco.Comfort.

Seats..Eco.Comfort.

Seat.Width..Economy.

Seat.Pitch..Economy.

Seats..Economy.

Accommodation

Cruising.Speed..mph.

Range..miles.

Engines

Wingspan..ft.

Tail.Height..ft.

Length..ft.

Wifi

Video

Power

Satellite

Flat.bed

Sleeper

Club

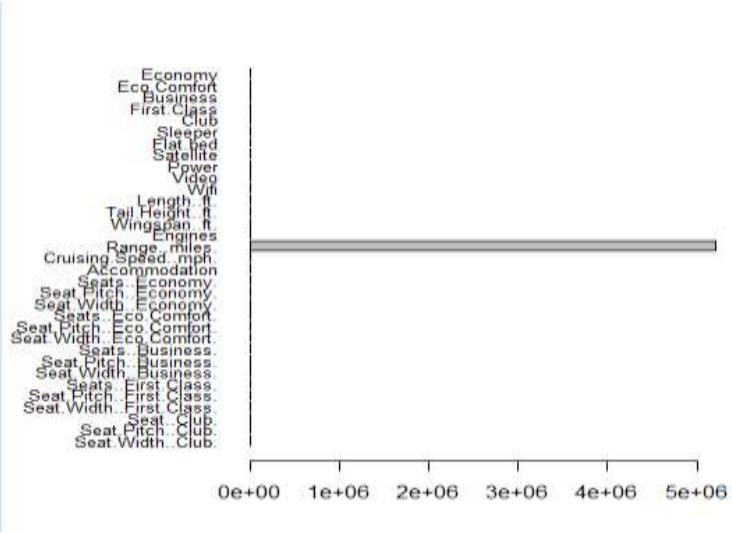
First.Class

Business

Eco.Comfort

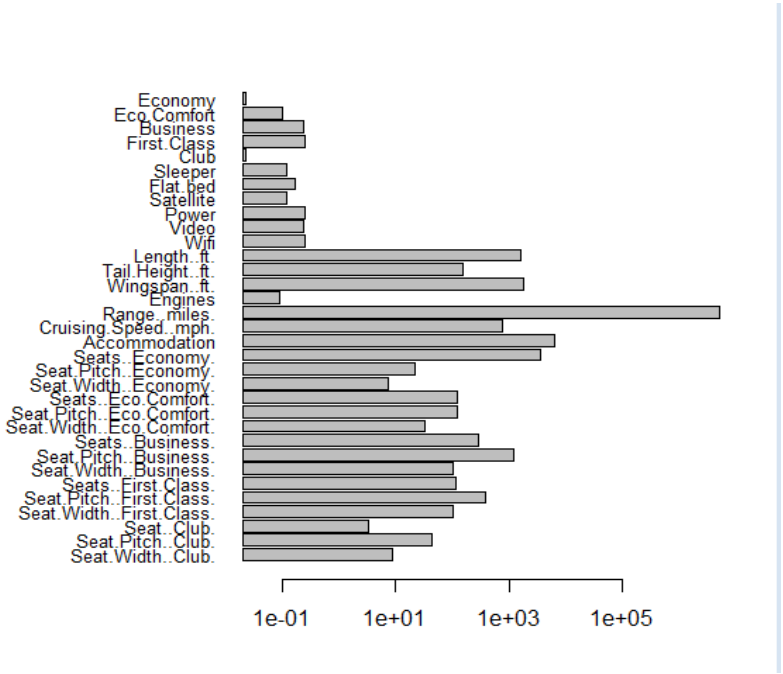
Economy

```
            Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
SS loadings     1.00  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.03  0.030  0.030  0.030  0.030  0.030  0.030  0.030  0.030
Cumulative Var  0.03  0.061  0.091  0.121  0.152  0.182  0.212  0.242  0.273
            Comp.10 Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16 Comp.17
SS loadings     1.000   1.000   1.000   1.000   1.000   1.000   1.000   1.000
Proportion Var  0.030   0.030   0.030   0.030   0.030   0.030   0.030   0.030
Cumulative Var  0.303   0.333   0.364   0.394   0.424   0.455   0.485   0.515
            Comp.18 Comp.19 Comp.20 Comp.21 Comp.22 Comp.23 Comp.24 Comp.25
SS loadings     1.000   1.000   1.000   1.000   1.000   1.000   1.000   1.000
Proportion Var  0.030   0.030   0.030   0.030   0.030   0.030   0.030   0.030
Cumulative Var  0.545   0.576   0.606   0.636   0.667   0.697   0.727   0.758
            Comp.26 Comp.27 Comp.28 Comp.29 Comp.30 Comp.31 Comp.32 Comp.33
SS loadings     1.000   1.000   1.000   1.000   1.000   1.000    1.00    1.00
Proportion Var  0.030   0.030   0.030   0.030   0.030   0.030    0.03    0.03
Cumulative Var  0.788   0.818   0.848   0.879   0.909   0.939    0.97    1.00
# verify by plotting variance of columns
> mar <- par()$mar
> par(mar=mar+c(0,5,0,0))
> barplot(sapply(data, var), horiz=T, las=1, cex.names=0.8)
```



> barplot(sapply(data, var), horiz=T, las=1, cex.names=0.8, log='x')

> par(mar=mar)



All the important components and its standard deviation, proportion of variance and cumulative proportion are displayed. Loading pc loads all the components and it shows that the variance are in the range in miles.It also displays proportion variance, cumulative variance.

The dataset variable scaling is different and all variables are not of the same type. Due to the scale of the different variables,the data set is quite variable. By plotting the variance in dataframe for all different columns we can see a major difference.

When the graph is plotted by regularly only one variable is dominant in it and the variance of it is shown. When log x is added all the variables involved are displayed and mapped according to the variance of it. The variance of the variables are not constant and there are lots of differences.

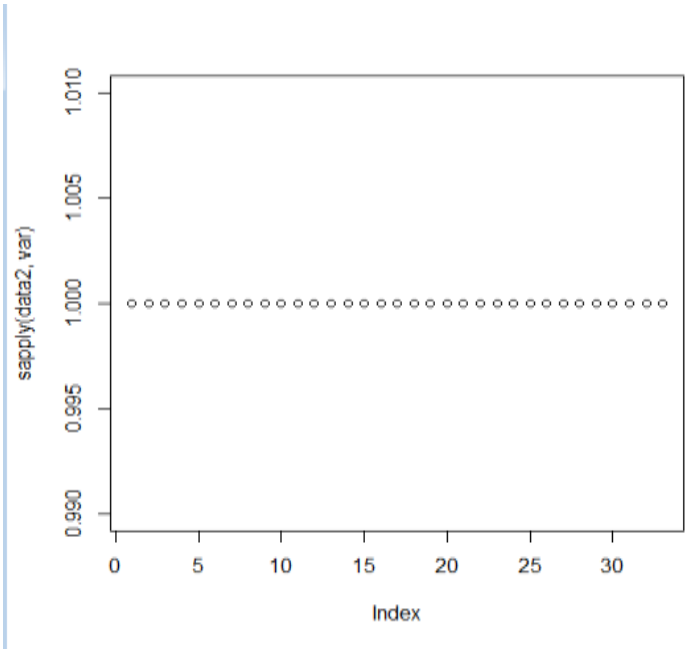
This difference in variance in both the graph is due to scalability. If the data is scaled properly then we would get a better result.

5. Analyze how scaling the data helps in this case:

```
# scale
> data2 <- data.frame(scale(data))

# Verify variance is uniform

> plot(sapply(data2, var))
```



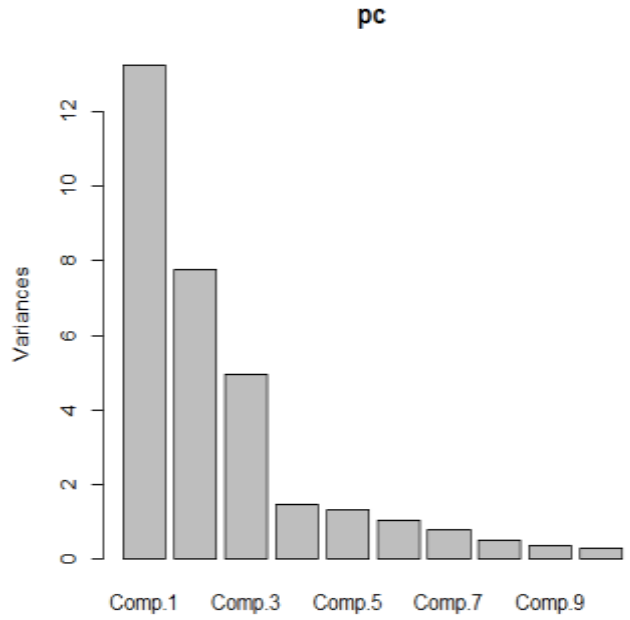
When the data was not scaled one variable was dominating. In order to avoid this we should scale the data. The delta dataset is scaled using the scale() function. The scaled data and variance are plotted in the graph across the index. This shows us the variance across different variables are equal . Using this scaled data will give a better PCA result where one variable will not dominate. From the graph it is seen that after applying the scale() function the variance is now constant across variables.

6. Apply PCA to the scaled data. How many components explain > 85% variance?

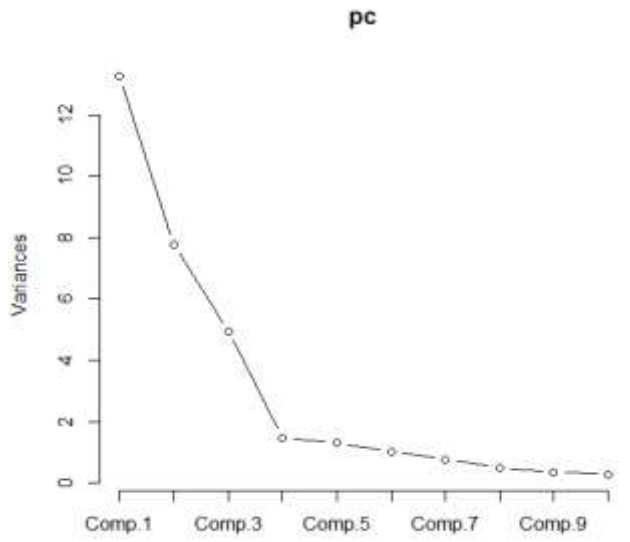
```
# Proceed with principal components

> pc <- princomp(data2)

> plot(pc)
```



> plot(pc, type='l')



> summary(pc)

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	3.6401340	2.7883991	2.2225223	1.21058843	1.14073049
Proportion of Variance	0.4108706	0.2410905	0.1531661	0.04544262	0.04034933
Cumulative Proportion	0.4108706	0.6519611	0.8051271	0.85056976	0.89091910

	Comp.6	Comp.7	Comp.8	Comp.9	Comp.10

Standard deviation 1.01495084 0.87821658 0.70479857 0.5848836 0.529084330

Proportion of Variance 0.03194187 0.02391517 0.01540282 0.0106074 0.008680007

Cumulative Proportion 0.92286097 0.94677614 0.96217896 0.9727864 0.981466375

Comp.11 Comp.12 Comp.13 Comp.14

Standard deviation 0.407860804 0.362177038 0.298829473 0.238584294

Proportion of Variance 0.005158153 0.004067355 0.002768963 0.001765038

Cumulative Proportion 0.986624528 0.990691883 0.993460846 0.995225884

Comp.15 Comp.16 Comp.17 Comp.18

Standard deviation 0.1981036 0.1757588137 0.1444047596 0.1385927988

Proportion of Variance 0.0012169 0.0009578654 0.0006465964 0.0005955958

Cumulative Proportion 0.9964428 0.9974006493 0.9980472457 0.9986428415

Comp.19 Comp.20 Comp.21 Comp.22

Standard deviation 0.1173372505 0.1045097508 0.0853608392 0.0791201165

Proportion of Variance 0.0004269157 0.0003386756 0.0002259371 0.0001941083

Cumulative Proportion 0.9990697572 0.9994084328 0.9996343699 0.9998284782

Comp.23 Comp.24 Comp.25 Comp.26

Standard deviation 5.177928e-02 4.469526e-02 0.0221107242 1.530274e-02

Proportion of Variance 8.313469e-05 6.194314e-05 0.0000151592 7.261202e-06

Cumulative Proportion 9.999116e-01 9.999736e-01 0.9999887152 9.999960e-01

Comp.27 Comp.28 Comp.29 Comp.30

Standard deviation 1.080092e-02 3.619390e-03 4.267325e-08 4.071751e-08

Proportion of Variance 3.617359e-06 4.062012e-07 5.646532e-17 5.140824e-17

Cumulative Proportion 9.999996e-01 1.000000e+00 1.000000e+00 1.000000e+00

Comp.31 Comp.32 Comp.33

Standard deviation 9.643864e-09 7.10068e-09 0

Proportion of Variance 2.883848e-18 1.56340e-18 0

Cumulative Proportion 1.000000e+00 1.00000e+00 1

The principal components is applied to the scaled data. This can also be done automatically in call to the prcomp() function by setting the parameter scale=TRUE. Now the result is better than the previous graph. All the pc components are displayed in the graph and plotted across their variance. The summary is also displayed with all the standard deviation, Proportion of variance and Cumulative Proportion.

In the PC graph the variance of three components are high and from the fourth component it is decreased. Similarly in the second graph the variance is dropped gradually.

Now after scaling the data and applying PCA provides a better result. There are various rules of thumb for selecting the number of principal components to retain in an analysis of this type.

1. We can select the number of components which has 85% or more than 85% of the variance.
2. From second graph it is seen that we can use the elbow method to identify the components.

From both the graph we get that the first four principle components show greater variance. So we can take these four PC into data frame and plot.

7. Create PCA representation

```
# Get principal component vectors using prcomp instead of princomp
pc <- prcomp(data2)
# First for principal components

comp <- data.frame(pc$x[,1:4])
```

The principle component vectors are obtained and in that first four principle components are considered.
Using this principle components graphs can be plotted. The spacing between the variable, the variance and the visualisation of it is better.

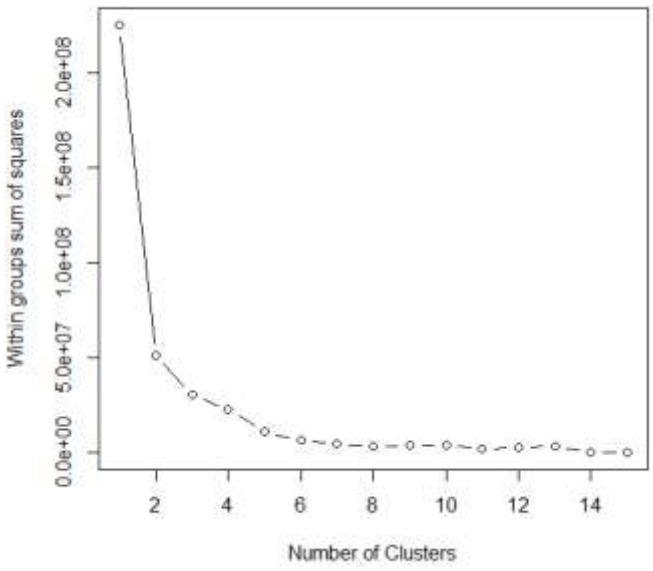
8. a. Detemine the number of clusters: how many will you use and why?

```
# Determine number of clusters

wss <- (nrow(data)-1)*sum(apply(data,2,var))

for (i in 2:15) wss[i] <- sum(kmeans(data, centers=i)$withinss)

> plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")
```



In this graph the delta scaled data frame is used. The elbow method for kmeans() function is used. It is similar to elbow method where kmeans partition function is used. The k values are varied from 2 to 15 and wss values are calculated for each k values.

The values are plotted in the graph according to k means clustering. The wss value is decreased for every iteration of k value. From observation, there is a marginal gain drop in the graph at a particular cluster value. This k value is considered as the appropriate cluster value. The location of the bend(knee) indicates the number of clusters.

It can be seen that the marginal gain decreases nad the elbow curve is found. From k 7 to 14 there is no decrease in the marginal gain and it is almost maintained. From k =4 it drops to a lower level and then it is maintained. So from the graph it can be found that it suggest 4 cluster solution. So k=4 from elbow method. Using k as 4 the further analysis and clustering can be done.

8.b. Apply kmeans. How many clusters k will you use and why?

> k <- kmeans(comp, 4, nstart=25, iter.max=1000)

K= 4. We have used 4 clusters for kmeans.

k-means is an iterative method which minimizes the within-class sum of squares for a given number of clusters. The algorithm starts with an initial guess for cluster centers, and each observation is placed in the cluster to which it is closest. The cluster centers are then updated, and the entire process is repeated until the cluster centers no longer move. k-means is said to be a reallocation method.
The general principle:
1. Select as many points as the number of desired clusters to create initial centers.
2. Each observation is then associated with the nearest center to create temporary clusters.

3. The gravity centers of each temporary cluster are calculated and these become the new cluster centers.
4. Each observation is reallocated to the cluster which has the closest center.
5. This procedure is iterated until convergence.

From the above Elbow method, the number of clusters for the delta data set was found. We altered the k values from 2 to 15 and the wss values were calculated for each k values. The values are plotted in the graph. According to the elbow method, the bend point where there is a marginal gain is dropped and after some additional clusters it remains the same denotes the appropriate number of cluster. So after analysing it from the elbow method and graph, the number of clusters or k we chose for kmeans clustering is 4.

9. Since there are no actual class labels for this data, we do exploratory analysis of the clustering results to evaluate them. Look at the exact clusters below, in order of increasing size: do they make sense to you?

```
# Cluster sizes

> sort(table(k$clust))

 1  3  4  2

 1  4 15 24

> clust <- names(sort(table(k$clust)))

# First cluster

> row.names(data[k$clust==clust[1],])

[1] "Airbus A319 VIP"

# Second cluster

> row.names(data[k$clust==clust[2],])

[1] "CRJ 100/200 Pinnacle/SkyWest" "CRJ 100/200 ExpressJet"

[3] "E120"                         "ERJ-145"

# Third Cluster

> row.names(data[k$clust==clust[3],])

 [1] "Airbus A330-200"         "Airbus A330-200 (3L2)"

 [3] "Airbus A330-200 (3L3)"   "Airbus A330-300"

 [5] "Boeing 747-400 (74S)"    "Boeing 757-200 (75E)"

 [7] "Boeing 757-200 (75X)"    "Boeing 767-300 (76G)"

 [9] "Boeing 767-300 (76L)"    "Boeing 767-300 (76T)"
```

[11] "Boeing 767-300 (76Z V.1)" "Boeing 767-300 (76Z V.2)"

[13] "Boeing 767-400 (76D)" "Boeing 777-200ER"

[15] "Boeing 777-200LR"

Fourth Cluster

> row.names(data[k$clust==clust[4],])

[1] "Airbus A319" "Airbus A320" "Airbus A320 32-R"

[4] "Boeing 717" "Boeing 737-700 (73W)" "Boeing 737-800 (738)"

[7] "Boeing 737-800 (73H)" "Boeing 737-900ER (739)" "Boeing 757-200 (75A)"

[10] "Boeing 757-200 (75M)" "Boeing 757-200 (75N)" "Boeing 757-200 (757)"

[13] "Boeing 757-200 (75V)" "Boeing 757-300" "Boeing 767-300 (76P)"

[16] "Boeing 767-300 (76Q)" "Boeing 767-300 (76U)" "CRJ 700"

[19] "CRJ 900" "E170" "E175"

[22] "MD-88" "MD-90" "MD-DC9-50"

The k value is set as 4 and the kmeans clustering is done. So we have four clusters. The four clusters are ordered according to their increasing size. Each cluster is then printed. From the clusters and their values it is seen that the first cluster contains a single aircraft, the Airbus A319 VIP. The name says that its not for a regular use and its only for VIP's.

The second cluster contains four aircraft. Two CRJ 100/200's and the Embraer E120 and ERJ-145. On analyzing the data of these four aircraft it is found that these are the smallest passenger aircraft, with the smallest accommodations. As such, there is only economy seating in these planes which is what distinguishes them from the remainder of the fleet.

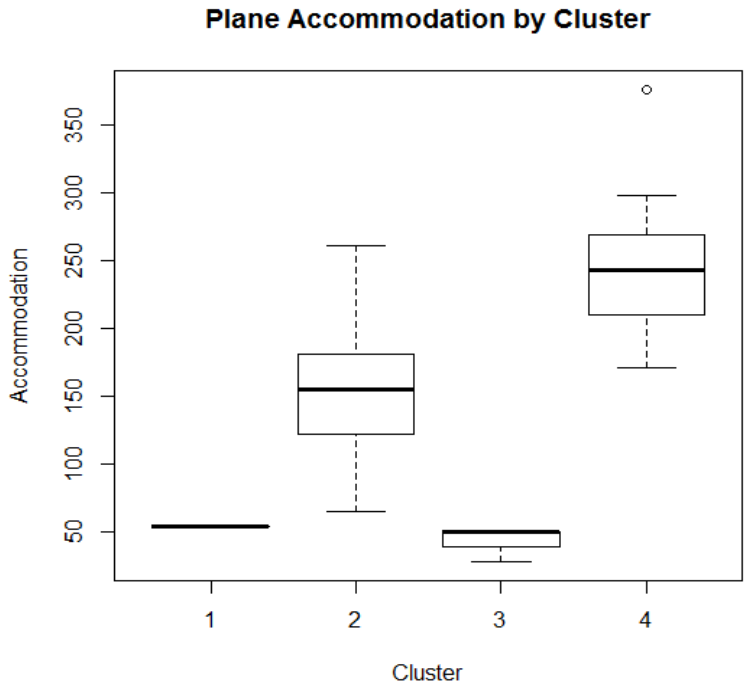
The third cluster contains 15 aircrafts and the fourth cluster has 24 aircraft. These two clusters has the remaining planes which most of the people will use for travelling. The aircrafts include Boeing, Airbus.

These two clusters can be split again based on the plane size though there is crossover in Boeing aircraft.

From the analysis it is found that the clusters are formed with data which share similar size, accommodation and purpose. So these four clusters denote four groups of aircraft which is used for different purposes.

10. Attribute analysis vs clusters

```
# Compare accommodation by cluster in boxplot
 boxplot(data$Accommodation ~ k$cluster, xlab='Cluster', ylab='Accommodation', main='Plane Accommodation by Cluster')
```



Compare presence of seat classes in largest clusters
> data[k$clust==clust[3],30:33]

	First.Class	Business	Eco.Comfort	Economy
Airbus A330-200	0	1	1	1
Airbus A330-200 (3L2)	0	1	1	1
Airbus A330-200 (3L3)	0	1	1	1
Airbus A330-300	0	1	1	1
Boeing 747-400 (74S)	0	1	1	1
Boeing 757-200 (75E)	0	1	1	1
Boeing 757-200 (75X)	0	1	1	1
Boeing 767-300 (76G)	0	1	1	1
Boeing 767-300 (76L)	0	1	1	1
Boeing 767-300 (76T)	0	1	1	1
Boeing 767-300 (76Z V.1)	0	1	1	1
Boeing 767-300 (76Z V.2)	0	1	1	1
Boeing 767-400 (76D)	0	1	1	1
Boeing 777-200ER	0	1	1	1

```
Boeing 777-200LR                0       1         1       1
> data[k$clust==clust[4],30:33]
                    First.Class Business Eco.Comfort Economy
Airbus A319                   1       0         1       1
Airbus A320                   1       0         1       1
Airbus A320 32-R              1       0         1       1
Boeing 717                    1       0         1       1
Boeing 737-700 (73W)          1       0         1       1
Boeing 737-800 (738)          1       0         1       1
Boeing 737-800 (73H)          1       0         1       1
Boeing 737-900ER (739)        1       0         1       1
Boeing 757-200 (75A)          1       0         1       1
Boeing 757-200 (75M)          1       0         1       1
Boeing 757-200 (75N)          1       0         1       1
Boeing 757-200 (757)          1       0         1       1
Boeing 757-200 (75V)          1       0         1       1
Boeing 757-300                1       0         1       1
Boeing 767-300 (76P)          1       0         1       1
Boeing 767-300 (76Q)          1       0         1       1
Boeing 767-300 (76U)          0       1         1       1
CRJ 700                       1       0         1       1
CRJ 900                       1       0         1       1
E170                          1       0         1       1
E175                          1       0         1       1
MD-88                         1       0         1       1
MD-90                         1       0         1       1
MD-DC9-50                     1       0         1       1
>
```

Though the data had quantitative and catergorical attributes. The data are compared and separated through clustering method. So after clustering analysis the aircraft has four attributes. They are First.Class, Business, Eco.Comfort, Economy. The clusters are formed based on similarities in these four attributes. Similar kind of aircrafts with attributes value nearly same are grouped into separate clusters.

When each cluster is analysed separately it can be found easily. The two large clusters can be divided again based on the seating and the size of the aircraft.

From the above two results of the two large clusters the main difference found is one cluster has First Class seating whereas the other cluster has only Business, Eco.Comfort, Economy. The cluster which does not have First Class attribute has all the values of the First class attribute as 0.

The cluster which has the First class attribute has 1 as the value of the attribute except Boeing 767-300 (76U)]. The value of Business class attribute is 0 in this cluster since it has the First class seating attribute.

II.2 Pima dataset

1. Apply PCA to the Pima data, that you used in HW 2. Discuss the number of PC component you will use and why. Compare the PCA representation to the result of the feature selection you did in HW 2. Do you see any commonalities?

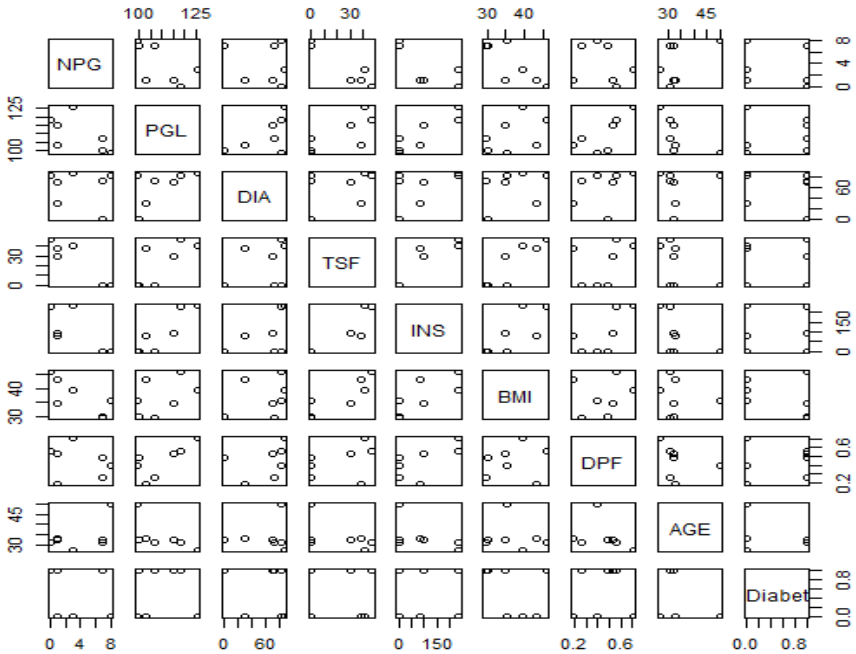
Load the Pima data set.

Pima Indians Diabetes dataset is a dataset of 786 samples of diabetic and healthy individuals. The Data set characteristics is multivariate and the attribute characteristics are integer and real. It has 8 attributes and one class variable. It has missing values also.

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

```
data <- read.csv(file="delta.csv", header=T, sep=",", row.names=1)

# scatterplot matrix of intermediary (size/non-categorical) variables
plot(data[16:22,])
```

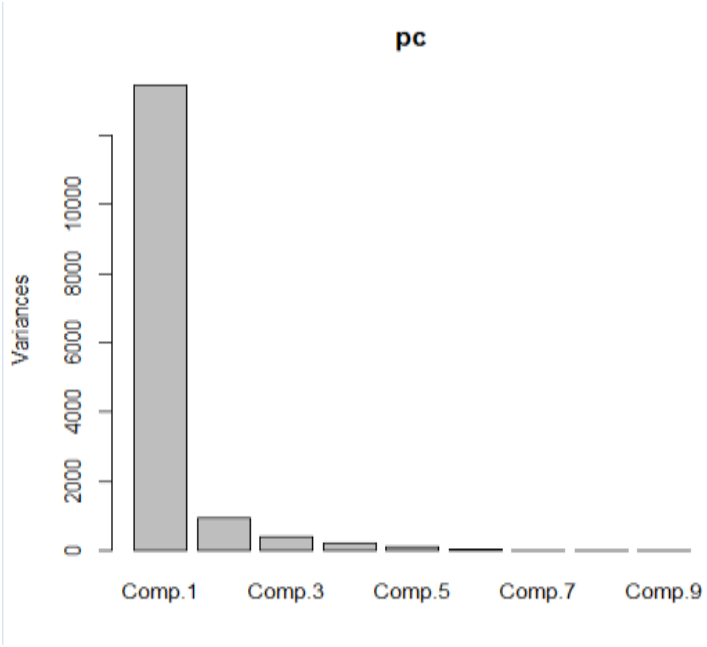


There are strong positive correlations between all these variables, as all of them are related to the each other. But there are some variables which are not related. In order to analyse the high dimensional data we can go for PCA method.

Apply the PCA on raw data.

apply principal components analysis to raw data and plot

pc <- princomp(data)
plot(pc)



Let us analyse the data and the PCA of the raw data.

```
> summary(pc)
Importance of components:
                          Comp.1      Comp.2      Comp.3      Comp.4
Standard deviation     115.9269425 30.52198183 19.75017224 14.06861766
Proportion of Variance   0.8885336  0.06159289  0.02578973  0.01308603
Cumulative Proportion    0.8885336  0.95012648  0.97591621  0.98900224
                           Comp.5      Comp.6      Comp.7      Comp.8
Standard deviation     10.608798639 6.765921909 2.7844997381 4.051504e-01
Proportion of Variance  0.007441106 0.003026628 0.0005126246 1.085269e-05
Cumulative Proportion   0.996443349 0.999469977 0.9999826012 9.999935e-01
                           Comp.9
Standard deviation     3.146579e-01
Proportion of Variance 6.546095e-06
Cumulative Proportion  1.000000e+00

> loadings(pc)

Loadings:
     Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
NPG                         0.152        0.986
PGL         -0.972 -0.143 -0.120
DIA         -0.142  0.922  0.263 -0.232
TSF                0.307 -0.884  0.260 -0.221
INS    -0.993
BMI                0.132 -0.193        0.971
DPF                                     0.239  0.971
AGE         -0.140  0.125  0.301  0.920       -0.163
```

Diabet 0.971 -0.239

Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9

SS loadings 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Proportion Var 0.111 0.111 0.111 0.111 0.111 0.111 0.111 0.111 0.111

Cumulative Var 0.111 0.222 0.333 0.444 0.556 0.667 0.778 0.889 1.000

>

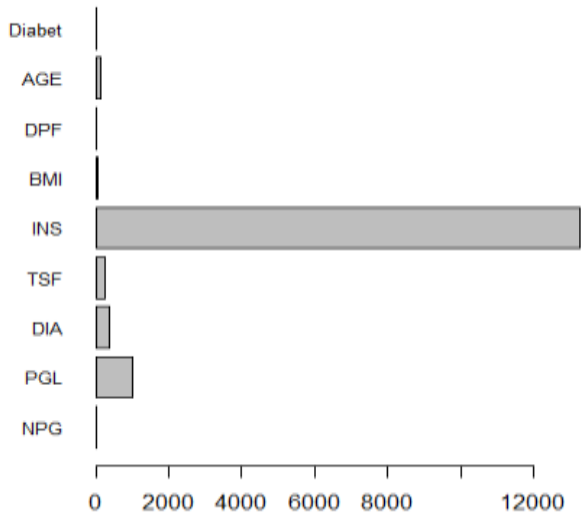
On applying PCA on the raw diabetes data set we can find many pc components. But the variance of the first pc component is dominating the other components. The summary and loadings display the importance of components like standard deviation, Proportion of Variance, Cumulative proportion. Proportion variance and cumulative variance is displayed by loading pc.

Verify the variance of the column by plotting

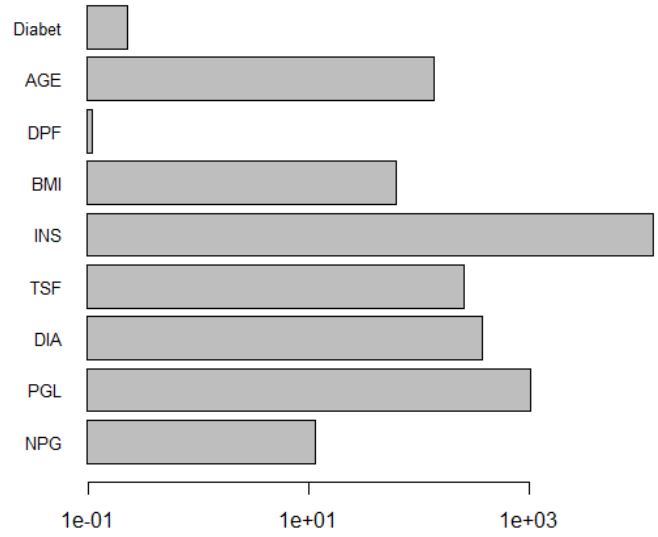
> mar <- par()$mar

> par(mar=mar+c(0,5,0,0))

> barplot(sapply(data, var), horiz=T, las=1, cex.names=0.8)



> barplot(sapply(data, var), horiz=T, las=1, cex.names=0.8, log='x')



> par(mar=mar)

The dataset variable scaling is different and all variables are not of the same type. Due to the scale of the different variables,the data set is quite variable. By plotting the variance in dataframe for all different columns we can see a major difference.

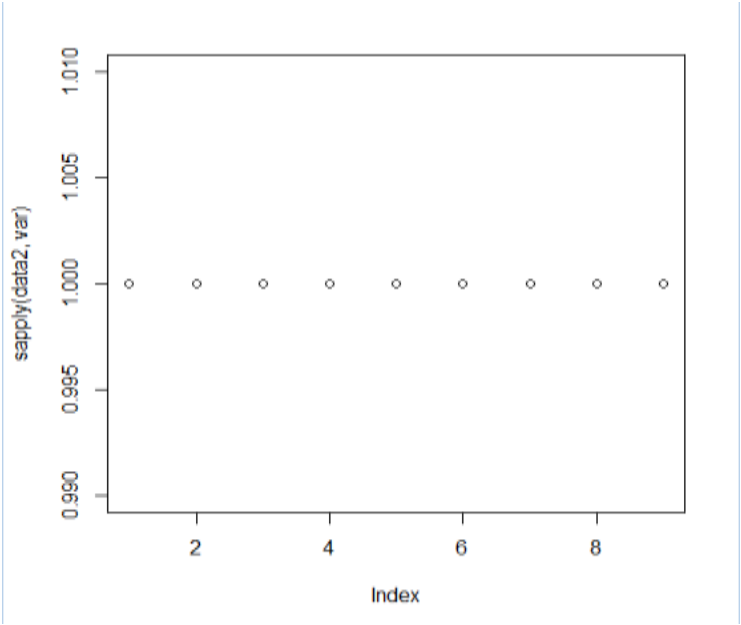
When the graph is plotted by regularly only one variable is dominant in it and the variance of it is shown. When log x is added all the variables involved are displayed and mapped according to the variance of it. The variance of the variables are not constant and there are lots of differences.

This difference is due to scalability and it can be over come by scaling the dataset. The PCA on raw data is not efficient and it misleads.

Let us scale the Pima dataset

> data2 <- data.frame(scale(data))

> plot(sapply(data2, var))

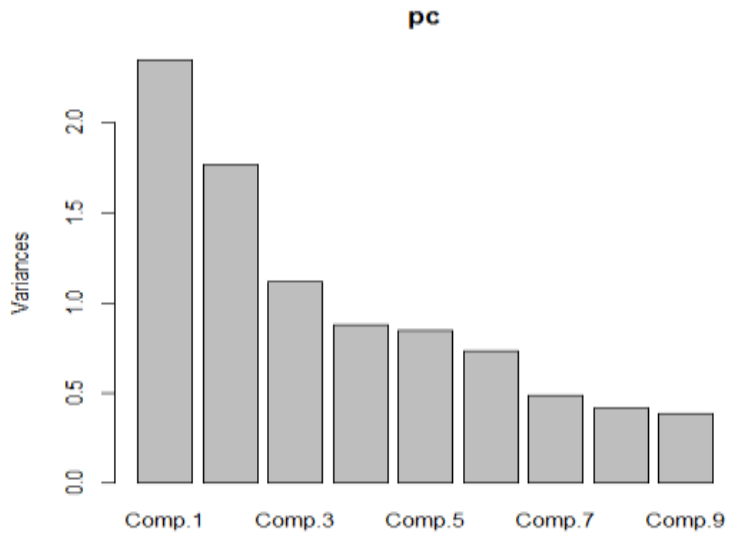


On scaling the dataset the variables variance are equal. This scaled data set can be used for PCA.

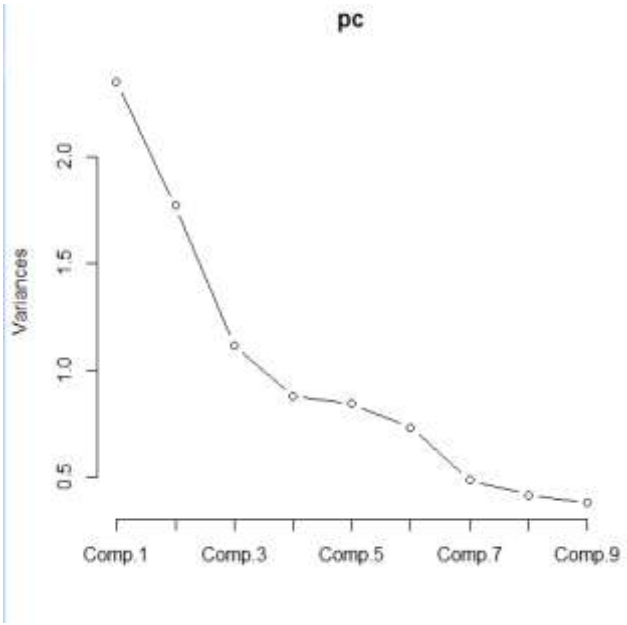
Let us apply the PCA to the scaled data and create PCA representation

> pc <- princomp(data2)

> plot(pc)



> plot(pc, type='l')



> summary(pc)

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	1.5327878	1.3311656	1.0577176	0.93851292	0.91843543
Proportion of Variance	0.2613891	0.1971458	0.1244695	0.09799499	0.09384705
Cumulative Proportion	0.2613891	0.4585348	0.5830043	0.68099929	0.77484634

	Comp.6	Comp.7	Comp.8	Comp.9
Standard deviation	0.85668628	0.69841781	0.64624811	0.6200073
Proportion of Variance	0.08165203	0.05426927	0.04646457	0.0427678
Cumulative Proportion	0.85649836	0.91076763	0.95723220	1.0000000

> pc <- prcomp(data2)

> comp <- data.frame(pc$x[,1:4])

After applying PCA on scaled data set we got two plotted graphs. Analyzing the scaled data is more easier when compared to the previous one. There are various rules of thumb for selecting the number of principal components to retain in an analysis of this type

1. Pick the number of components which has 1.0 or greater of the variance
2. Use the 'elbow' method of the screen plot

From both the graph it suggest that 3 components can be used to. The elbow method shows that there is a marginal drop at k=3.

We can retain these 3 components for further analysis and create PCA representation.

In assignment 2 the feature selection was done using weka tool.

InfoGainAttributeEval : Evaluates the worth of an attribute by measuring the information gain with respect to the class.
CorrelationAttributeEval : Evaluates the worth of an attribute by measuring the correlation (Pearson's) between it and the class. By measuring the information gain and correlation of the attributes, the attributes will be selected for the classifier.

In PCA we don't use information gain of a particular attribute or the correlation of the attributes. Instead we use standard deviation and variance and covariance of p variables.
It takes a data matrix of n objects by p variables, which may be correlated, and summarizes it by uncorrelated axes that are linear combinations of the original p variables.

The first k components display as much as possible of the variation among objects. The centroid of the points is defined by the mean of each variable the variance of each variable is the average squared deviation of its n values around the mean of that variable degree to which the variables are linearly correlated is represented by their covariances.

PCA uses Euclidean Distance calculated from the p variables as the measure of dissimilarity among the n objects.In feature selection we don't use any distance to measure the dissimilarity.

Yes there are commonalities in selecting the feature. Based on Infogain and Correlation, attribute with highest rank is selected for the classification. Similarly the variable with high variance is selected as the first PC1 p variable with which the next variables are selected. It is found in both case the attribute age will be selected as the feature and it will be selected as the p variable with high variance.

2. Compute kmeans clustering with the PCA representation. Compare the results to the clustering on the original Iris data that you created in the first part of the assignment:

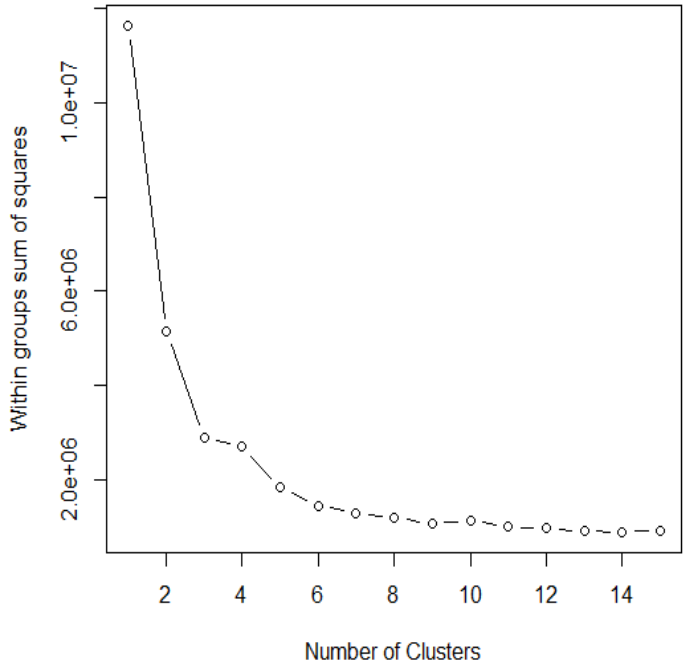
a. Discuss the number of clusters you use
b. Discuss the WSS measures for both clusterings. Note that your PCA data is different from the original in the number of dimension and the ranges of the values. Discuss how it matters for WSS

Determine number of clusters

```
> wss <- (nrow(data)-1)*sum(apply(data,2,var))

> for (i in 2:15) wss[i] <- sum(kmeans(data, centers=i)$withinss)

> plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")
```




```
> k <- kmeans(comp, 3, nstart=25, iter.max=1000)
> sort(table(k$clust))

  3   2   1
171 217 380
> k <- kmeans(comp, 4, nstart=25, iter.max=1000)
> sort(table(k$clust))

  4   2   1   3
141 156 197 274

> k <- kmeans(comp, 5, nstart=25, iter.max=1000)
> sort(table(k$clust))

  1   2   5   3   4
 35 117 142 153 321
```

```
> clust <- names(sort(table(k$clust)))
> row.names(data[k$clust==clust[1],])
  [1] "2"   "4"   "6"   "7"   "8"   "11"  "16"  "19"  "28"  "33"  "34"  "36"
 [13] "39"  "41"  "47"  "48"  "50"  "51"  "52"  "53"  "56"  "60"  "61"  "63"
 [25] "64"  "66"  "69"  "70"  "72"  "74"  "75"  "76"  "78"  "79"  "80"  "81"
 [37] "82"  "83"  "84"  "86"  "88"  "90"  "91"  "92"  "95"  "97"  "98"  "99"
 [49] "102" "103" "104" "105" "106" "107" "108" "109" "110" "113" "114" "118"
 [61] "119" "120" "122" "123" "125" "127" "128" "135" "136" "137" "138" "139"
 [73] "140" "142" "143" "146" "150" "152" "157" "158" "159" "164" "167" "168"
 [85] "169" "170" "173" "174" "175" "181" "182" "183" "184" "191" "197" "198"
 [97] "201" "202" "203" "204" "206" "209" "211" "218" "223" "225" "226" "227"
[109] "230" "233" "234" "235" "240" "241" "242" "243" "250" "252" "253" "254"
[121] "256" "257" "258" "262" "263" "267" "269" "270" "272" "273" "274" "276"
[133] "278" "280" "281" "289" "290" "291" "298" "301" "303" "306" "308" "311"
[145] "312" "314" "316" "317" "319" "321" "322" "323" "325" "326" "330" "332"
[157] "335" "337" "341" "342" "343" "347" "348" "349" "350" "351" "352" "353"
[169] "354" "355" "366" "368" "369" "372" "373" "374" "377" "378" "381" "382"
[181] "383" "384" "385" "386" "390" "391" "393" "394" "397" "398" "399" "406"
[193] "408" "411" "412" "414" "417" "419" "420" "422" "423" "424" "427" "431"
[205] "432" "433" "434" "435" "436" "438" "439" "442" "443" "447" "448" "449"
[217] "450" "451" "452" "453" "455" "458" "462" "463" "464" "466" "467" "468"
[229] "472" "473" "475" "478" "482" "483" "484" "485" "489" "491" "492" "493"
[241] "495" "497" "498" "501" "502" "503" "505" "508" "509" "512" "514" "515"
[253] "521" "522" "523" "525" "526" "527" "528" "529" "530" "531" "532" "533"
[265] "534" "535" "536" "537" "544" "545" "548" "551" "552" "554" "555" "556"
[277] "557" "563" "564" "565" "566" "567" "568" "569" "571" "572" "573" "574"
[289] "576" "577" "578" "582" "586" "588" "590" "592" "594" "597" "598" "600"
[301] "601" "602" "603" "606" "608" "610" "611" "614" "616" "618" "620" "621"
```

[313] "622" "624" "625" "626" "627" "628" "630" "632" "633" "634" "638" "640"

[325] "641" "642" "644" "645" "650" "651" "652" "653" "654" "655" "657" "666"

[337] "669" "672" "678" "679" "680" "681" "683" "686" "687" "688" "689" "693"

[349] "695" "698" "699" "700" "701" "704" "705" "706" "708" "714" "715" "719"

[361] "721" "722" "726" "727" "728" "729" "730" "734" "736" "737" "739" "742"

[373] "743" "752" "753" "759" "761" "765" "766" "768"

> row.names(data[k$clust==clust[2],])

[1] "1" "3" "10" "12" "13" "15" "18" "22" "23" "24" "25" "26"

[13] "27" "29" "30" "31" "35" "37" "38" "42" "43" "45" "49" "62"

[25] "65" "68" "73" "77" "85" "87" "89" "93" "94" "116" "117" "124"

[37] "130" "132" "134" "141" "144" "147" "149" "155" "160" "161" "162" "165"

[49] "166" "171" "177" "179" "180" "185" "192" "193" "194" "195" "205" "208"

[61] "210" "213" "215" "219" "220" "222" "224" "239" "246" "247" "251" "255"

[73] "264" "265" "266" "275" "277" "279" "282" "283" "284" "285" "286" "295"

[85] "299" "300" "304" "305" "307" "315" "318" "320" "324" "328" "331" "333"

[97] "334" "338" "340" "344" "345" "346" "356" "358" "359" "362" "363" "364"

[109] "367" "379" "387" "388" "392" "395" "401" "402" "404" "405" "407" "409"

[121] "437" "440" "444" "445" "454" "456" "457" "460" "461" "465" "469" "474"

[133] "476" "479" "480" "490" "496" "499" "504" "506" "510" "511" "513" "517"

[145] "518" "519" "520" "524" "538" "543" "550" "553" "558" "559" "560" "561"

[157] "579" "583" "584" "587" "591" "593" "599" "604" "605" "615" "617" "619"

[169] "623" "629" "631" "635" "636" "637" "643" "649" "659" "661" "665" "667"

[181] "668" "670" "671" "673" "675" "676" "677" "684" "685" "691" "692" "702"

[193] "707" "709" "712" "713" "718" "720" "724" "725" "731" "732" "735" "738"

[205] "740" "744" "746" "750" "751" "755" "757" "758" "760" "762" "763" "764"

[217] "767"

> a. From the elbow method applied to the scaled pima data set, we have plotted a graph. The k values is varied from 2 to 15. The wss is calculated for each value of k. From the graph obtained it is seen that the marginal gain is dropping gradually for increasing the k values. After some k

values there is not a big difference in the marginal gain drop. The elbow curve is found in the graph. From k = 3 to 15 the curve can be said as elbow curve. The location of the bend indicates the number of clusters that has to be used. In the graph the bend indicates the 3 cluster solution.

So the k value used for kmeans clustering is k=3. 3 cluster solution.

b. The basic idea behind partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation (known as total within-cluster variation or total within-cluster sum of square) is minimized:

Minimize $(\sum_{k=1}^{k} W(C_k))$ Where Ck is the kth cluster and W(Ck) is the within-cluster variation.

The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible

The within-cluster sum of squares is: $\sum_{k=1}^{k} \sum_{i \epsilon S_k} \sum_{j=1}^{p} (x_{ij} - \sim x_{kj})^2$ where S_k is the set of observatioins in the kth cluster and $\sim x_{kj}$ is the j th variable of the cluster center for the k th cluster.

The algebra of PCA is

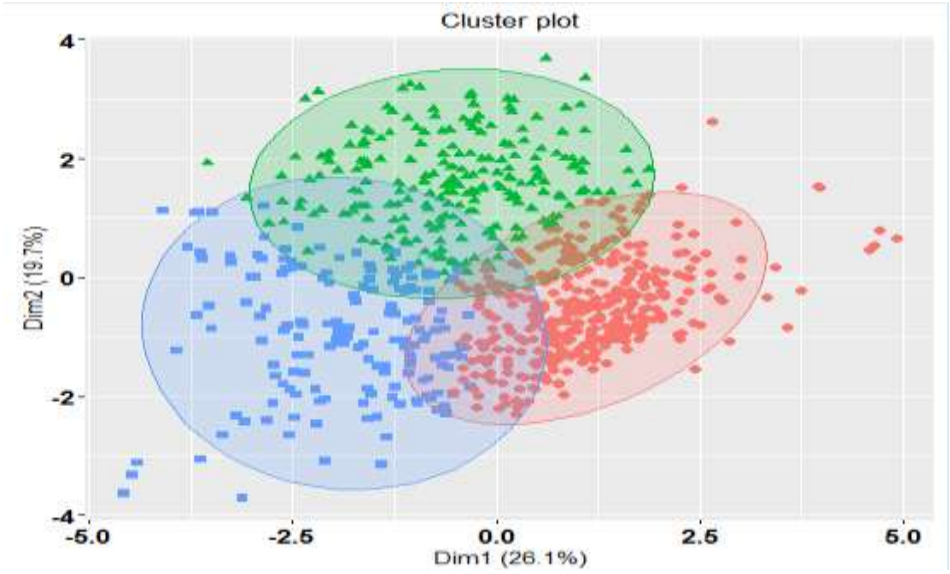
The coordinates of each object i on the kth principal axis, known as the scores on PC k, are computed as

$$z_{ki} = u_{1k}x_{1i} + u_{2k}x_{2i} + \cdots + u_{pk}x_{pi}$$

where Z is the n x k matrix of PC scores, X is the n x p centered data matrix and U is the p x k matrix of eigenvectors. Each eigen vector represents contribution of the given variable to a component.

So the wss values calculated for each k value is used to for the centered data matrix and from that eigen vectors can be identified. The wss or the variance used in PCA raw data are not of equal variance. It differs a lot and it is difficult to find the number of components from that. When the data is scaled the variance of the variables are equal and this can be used to find the number of components. The scaled data is much better and the PCA done on scaled data is good.

When we compare the k means clustering of the Iris data set and the Pima data set we see that the elbow method suggest 3 cluster solution. Since Iris had three class the 3 clusters are taken and the species are clustered based on that. In Pima data set based on the missing values and the accuracy the k means suggest 3 cluster solution. When the data set is clustered using k =3 the accuracy of the classification is more accurate. In Iris k means clustering we determined the cluster using elbow and hierarchical clustering methods and we plotted the clusters and displayed confusion matrix. In Pima we used elbow method to find the cluster number and then proceeded with kmeans which is under PCA. Though both dataset and kmeans are done in different way, the result would be similar in both the ways.



References:
http://publishingindia.com/GetBrochure.aspx?query=UERGQnJvY2h1cmVzfC8xODA5LnBkZnwvMTgwOS5wZGY=, http://www.everydayanalytics.ca/2014_06_01_archive.html, http://rpubs.com/najla/kmeansPCAdiabetes