ASSIGNMENT-2

TOPIC: Analysis is to find inherent clusters in sites (locations) and Plant Species

NAMES:1) MONIKA-(PRN 2134)

2)PRAKRITI PARSAILA(PRN-2145)

MENTOR: Dr.Akanksha Kashikar

OBJECTIVE: The objective of the analysis is to find relationships / inherent clusters in sites (locations) and/or plant species. The counts of various species at multiple quadrats at multiple locations are provided in various sheets in the excel file.

Data Importing and understanding:

Originally we are considering the data in the file named All sites All quadrats tree list final.xlsx which contains information of approx 682 plants species at different locations. The loactions present are:

Malshej_Ghat,Naneghat,Bhimashankar,Malegaon_Budruk,Dahuli,Lonavala,Tailbaila, Mulshi,Tamhini_Ghat,Gunjavane_dam,Abhepuri,Mahabaleshwar.

Data Preprocessing:

1) The missing values in the data-There are some missing values in the data after importing.we can replace NA values by Zero value,this is because plant species count loaction wise is given.
2) After that,we are joining totals of plants species of all the locations that gives us total variables equals to 12;
3) Finally,from plant species column we are finding unique species corresponding to different locations and clubbing them all that gives us total plant sepecies for consideration is equal to 214.

Hence,our data for final Analysis becomes of dimension (214*12). Code for the same is attached below-

```
rm(list = ls())
library(tidyverse)

library(readxl)


D1=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
            sheet = "Malshej Ghat"))
```

```
D1=na.omit(D1)
D1=D1[,-c(1,3:12)]
colnames(D1)=c("Plantname","MG")

D2=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
              sheet = "Naneghat"))

D2=na.omit(D2)
D2=D2[,-c(1,3:12)]
colnames(D2)=c("Plantname","NG")

D3=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
              sheet = "Bhimashankar"))

D3=na.omit(D3)
D3=D3[,-c(1,3:13)]
colnames(D3)=c("Plantname","BS")

D4=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
               sheet = "Malegaon Budruk"))

D4=na.omit(D4)
D4=D4[,-c(1,3:12)]
colnames(D4)=c("Plantname","MB")

D5=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
              sheet = "Dahuli"))

D5=na.omit(D5)
D5=D5[,-c(1,3:12)]
colnames(D5)=c("Plantname","DH")

D6=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
              sheet = "Lonavala"))

D6=na.omit(D6)
D6=D6[,-c(1,3:12)]
colnames(D6)=c("Plantname","LV")

D7=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
              sheet = "Tailbaila"))
D7=na.omit(D7)
D7=D7[,-c(1,3:16)]
colnames(D7)=c("Plantname","TB")
```

```r
D8=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
               sheet = " Mulshi Dam"))

D8=na.omit(D8)
D8=D8[,-c(1,3:16)]
colnames(D8)=c("Plantname","MD")

D9=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final.
xlsx",
              sheet = "Tamhini Ghat"))

D9=na.omit(D9)
D9=D9[,-c(1,3:16)]
colnames(D9)=c("Plantname","TG")

D10=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final
.xlsx",
              sheet = "Gunjavane Dam"))

D10=na.omit(D10)
D10=D10[,-c(1,3:12)]
colnames(D10)=c("Plantname","GD")

D11=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final
.xlsx",
              sheet = "Abhepuri"))

D11=na.omit(D11)
D11=D11[,-c(1,3:12)]
colnames(D11)=c("Plantname","AP")

D12=data.frame(read_excel("~/Downloads/All sites all quadrats tree list final
.xlsx",
              sheet = "Mahabaleshwar"))

D12=na.omit(D12)
D12=D12[,-c(1,3:23)]
colnames(D12)=c("Plantname","ML")

D=full_join(D1,D2)

## Joining, by = "Plantname"

D=full_join(D,D3)

## Joining, by = "Plantname"

D=full_join(D,D4)
```

```
## Joining, by = "Plantname"

D=full_join(D,D5)

## Joining, by = "Plantname"

D=full_join(D,D6)

## Joining, by = "Plantname"

D=full_join(D,D7)

## Joining, by = "Plantname"

D=full_join(D,D8)

## Joining, by = "Plantname"

D=full_join(D,D9)

## Joining, by = "Plantname"

D=full_join(D,D10)

## Joining, by = "Plantname"

D=full_join(D,D11)

## Joining, by = "Plantname"

D=full_join(D,D12)

## Joining, by = "Plantname"

D[is.na(D)]=0

rownames(D)=D[,1]
D=D[,-1]

 ##final data for analysis
head(D)

##                                  MG NG  BS MB DH LV TB MD TG GD AP  ML
## Acacia concinna (Willd.) DC       0  0   0  0  0  0  0  0  0  4  0   0
## Actinodaphne hookeri Meisn.      19 10 138  0  2  6  0  0  2 22 11 140
## Allophylus cobbe (L.) Raeusch    22 11  23 23 12  8 22  0  5  8 10  51
## Anodendron paniculatum A.DC       1  0   0  0  0  0  0  0  0  0  0   0
## Atalantia racemosa Wight ex Hook 36  6  73  1 29 22  5  0  4  5  0  17
## Bambusa bambos (L.) Voss         14  0   0  0  3 14  0  0  0  1  0   0
```

```
Malshej_Ghat=as.numeric(D[,1])
Naneghat=as.numeric(D[,2])
Bhimashankar=as.numeric(D[,3])
Malegaon_Budruk=as.numeric(D[,4])
Dahuli=as.numeric(D[,5])
Lonavala=as.numeric(D[,6])
Tailbaila=as.numeric(D[,7])
Mulshi=as.numeric(D[,8])
Tamhini_Ghat=as.numeric(D[,9])
Gunjavane_dam=as.numeric(D[,10])
Abhepuri=as.numeric(D[,11])
Mahabaleshwar=as.numeric(D[,12])

data=cbind(Malshej_Ghat,Naneghat,Bhimashankar,Malegaon_Budruk,Dahuli,Lonavala
,Tailbaila,Mulshi,Tamhini_Ghat,Gunjavane_dam,Abhepuri,Mahabaleshwar)
```

PART A} Identifing which species are most similar or most dissimilar.

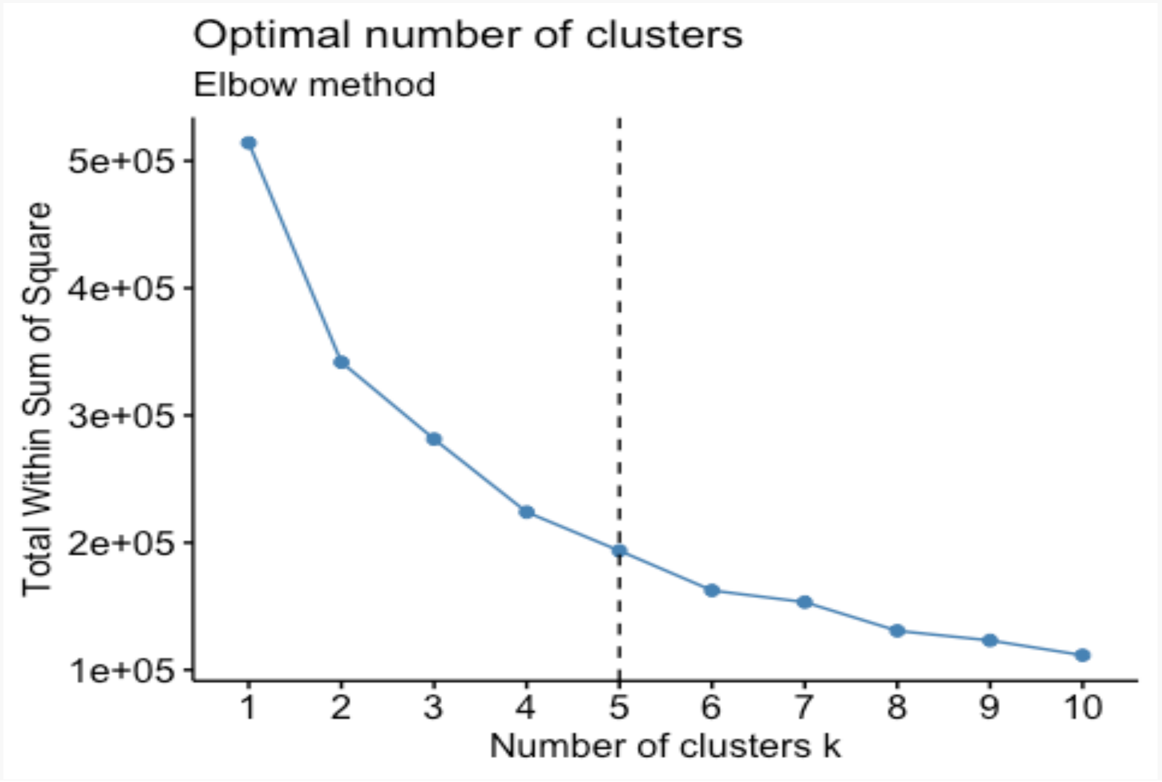
We will be clustering PLANT SPECIES first:

Methods to get optimal no of clusters:first step is to find the optimal no. of clusters for clustering species:

```
library(NbClust)
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://g
oo.gl/ve3WBa

library(cluster)
```

Elbow method:

```
fviz_nbclust(D, kmeans, method = "wss") +geom_vline(xintercept = 5, linetype
= 2)+labs(subtitle = "Elbow method")
```

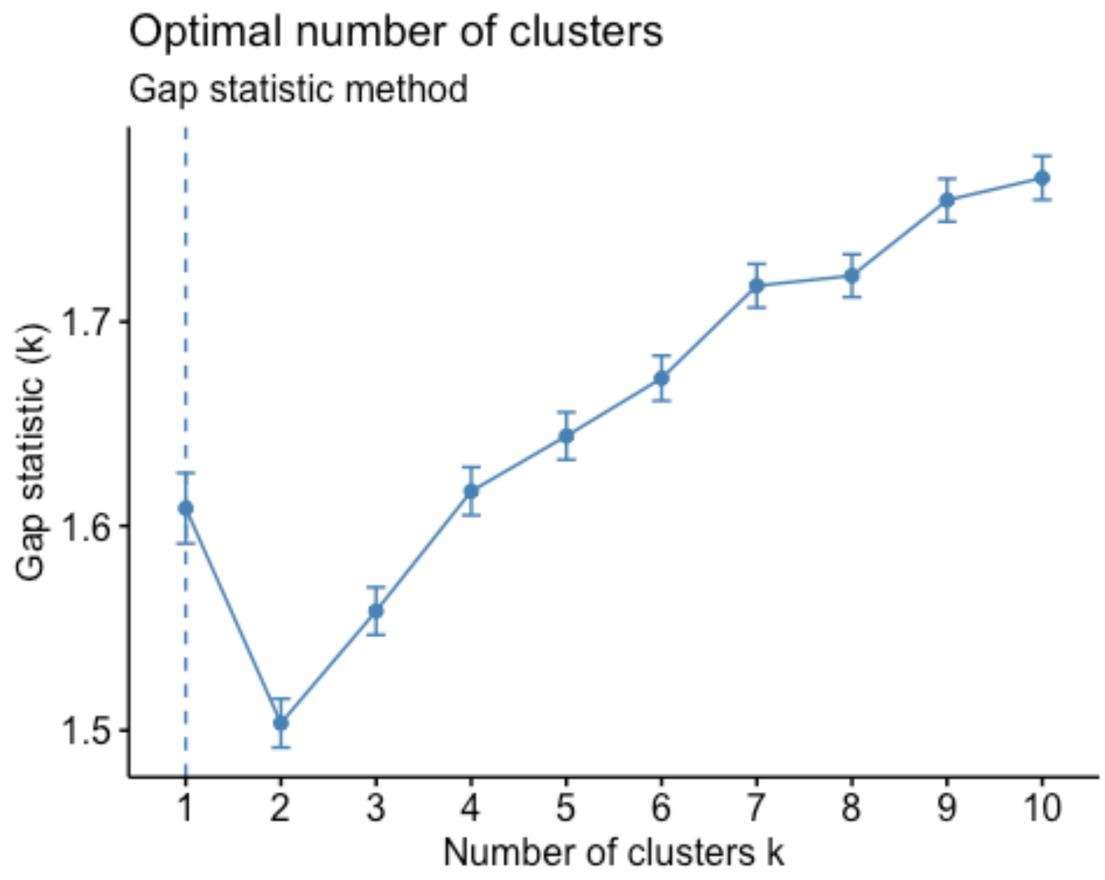


Gap statistic

The gap statistic compares the total intracluster variation for different values of k with their expected values under null reference distribution of the data.

nboot = 50 to keep the function speedy. recommended value: nboot= 50 for our analysis.

```
set.seed(123)
fviz_nbclust(data, kmeans, nstart = 25,  method = "gap_stat", nboot = 50)+
  labs(subtitle = "Gap statistic method")
```

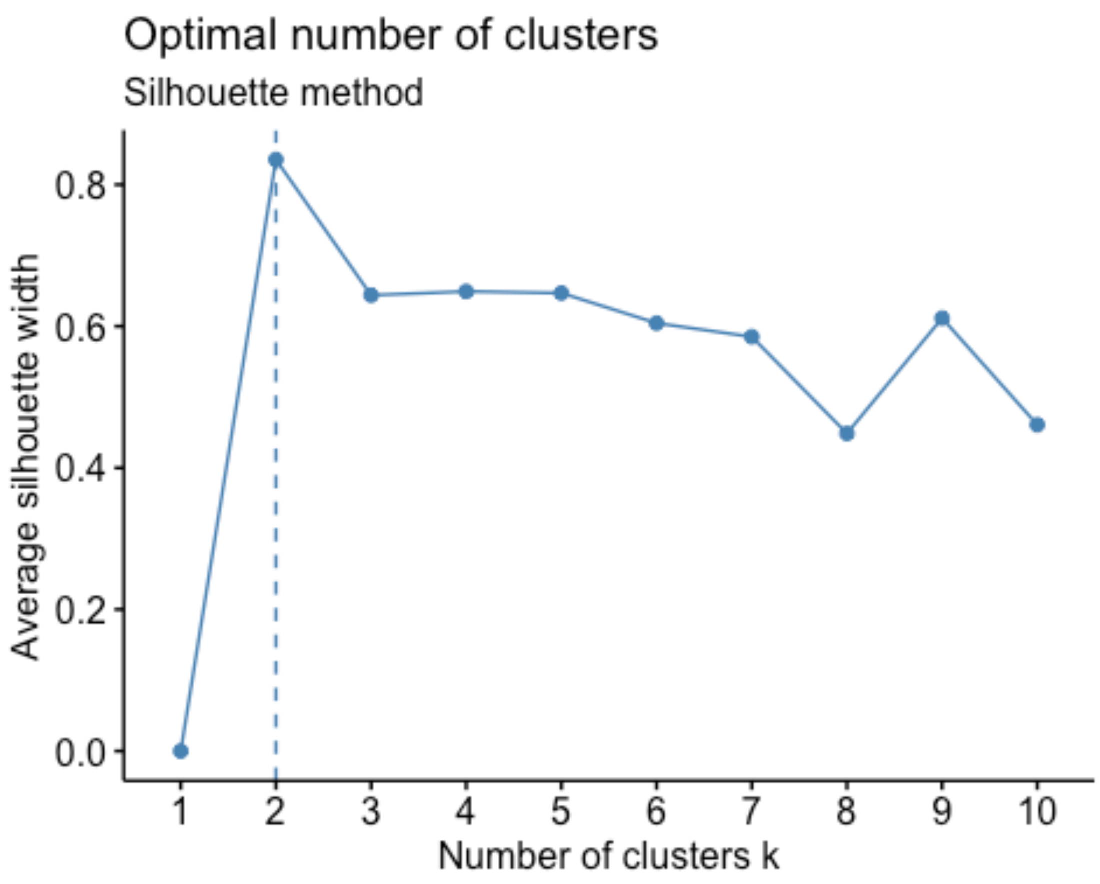


Silhouette method

The average silhouette approach measures the quality of a clustering. It determines how well each observation lies within its cluster.

A high average silhouette width indicates a good clustering. The silhouette method computes the silhouette of observations for different values of k.

```
fviz_nbclust(data, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



Elbow method: 5 clusters solution suggested.

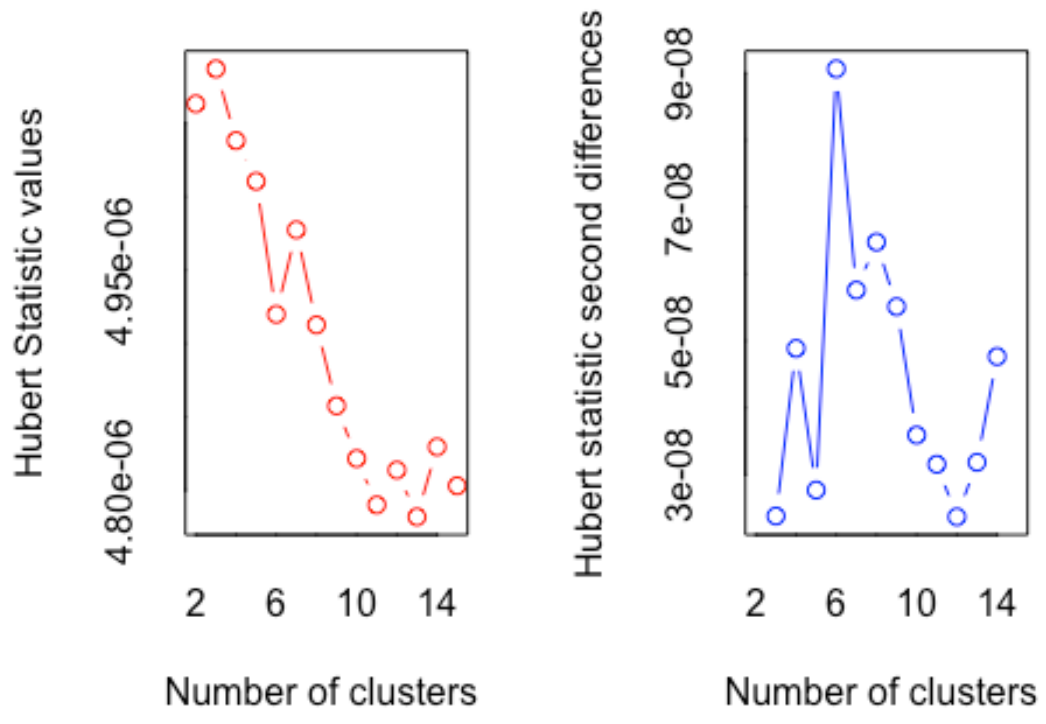
Silhouette method: 2 clusters solution suggested.

Gap statistic method: 1 clusters solution suggested.

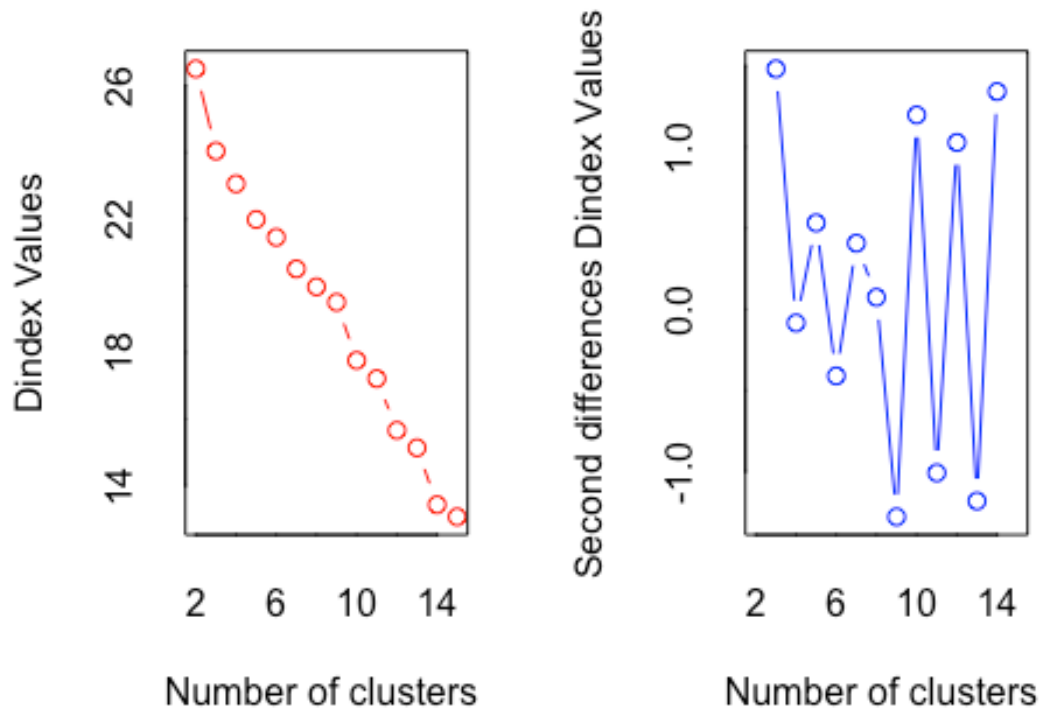
COMMENT:- The disadvantage of elbow and average silhouette methods is that, they measure a global clustering characteristic only. A more sophisticated method is to use the gap statistic which provides a statistical procedure to formalize the elbow/silhouette heuristic in order to estimate the optimal number of clusters.

Further, we will be using NbClust command to find optimal clusters from all the 30 indices.

```
NbClust(data = data, diss = NULL, distance = "euclidean", min.nc = 2, max.nc
= 15,
        method = "complete", index = "all", alphaBeale = 0.1)
```



```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee th
at corresponds to a
##                 significant increase of the value of the measure i.e the s
ignificant peak in Hubert
##                 index second differences plot.
##
```



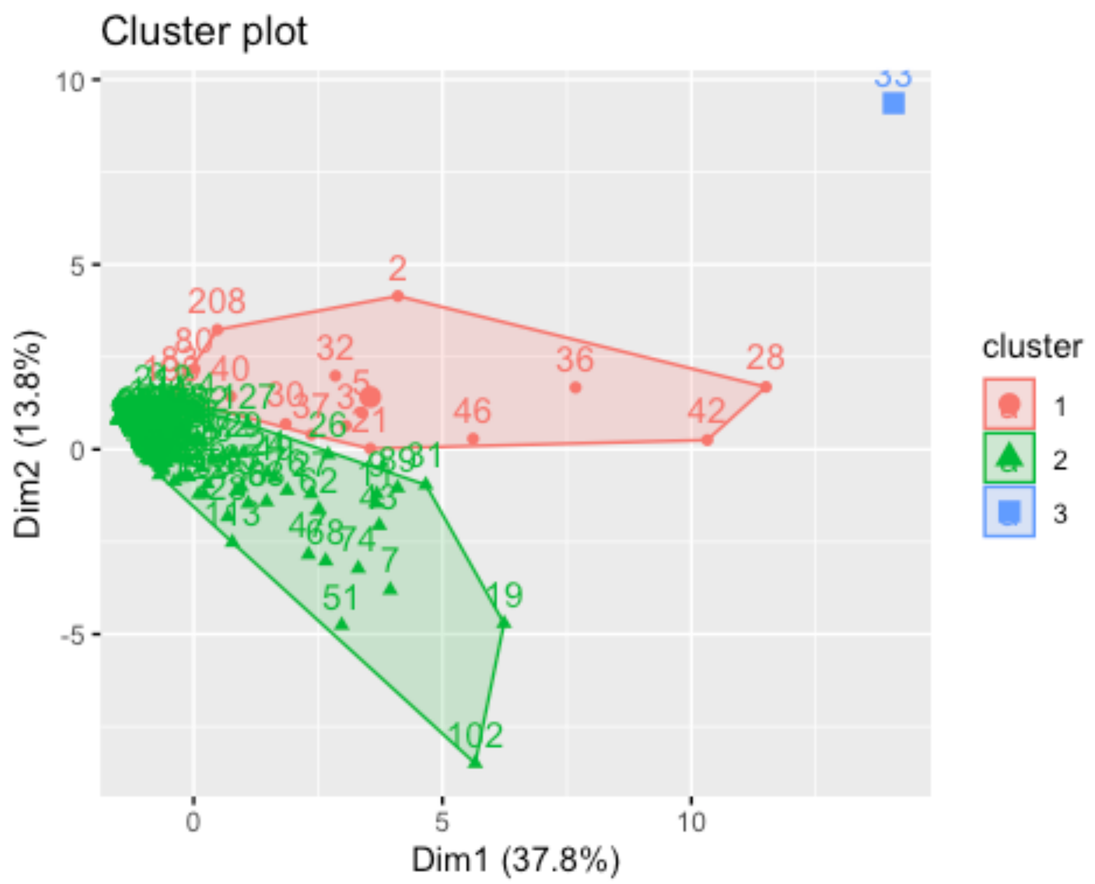
```
## *** : The D index is a graphical method of determining the number of clust
ers.
##                 In the plot of D index, we seek a significant knee (the si
gnificant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 6 proposed 2 as the best number of clusters
## * 8 proposed 3 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 2 proposed 9 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 13 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
```

Note:- According to the majority rule, the best number of clusters is 3. Hence,k=3.

kmeans clustering in R where we are going to use k=3:

```
library(factoextra)
k1= kmeans(D, centers = 3, nstart = 25)
str(k1)

## List of 9
##  $ cluster     : Named int [1:214] 2 1 1 2 1 2 2 2 2 2 ...
##   ..- attr(*, "names")= chr [1:214] "Acacia concinna (Willd.) DC" "Actinod
aphne hookeri Meisn." "Allophylus cobbe (L.) Raeusch" "Anodendron paniculatum
A.DC" ...
##  $ centers     : num [1:3, 1:12] 20.75 1.45 24 15.69 2.23 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "1" "2" "3"
##   .. ..$ : chr [1:12] "MG" "NG" "BS" "MB" ...
##  $ totss       : num 514121
##  $ withinss    : num [1:3] 113066 150750 0
##  $ tot.withinss: num 263816
##  $ betweenss   : num 250305
##  $ size        : int [1:3] 16 197 1
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

fviz_cluster(k1, data = data)
```

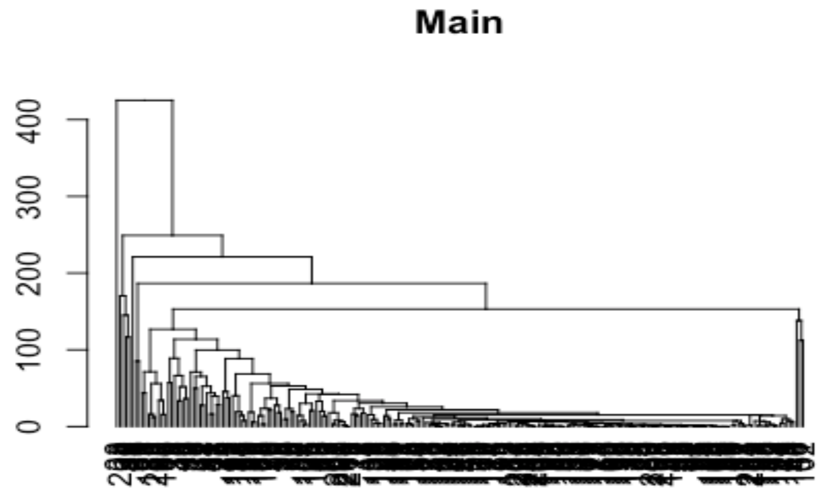


Here we are viewing our kmeans results by using fviz_cluster. This provides a beautiful illustration of the clusters.From the above graph,it is clear that 33rd species i.e,"Memecylon umbellatum Burm.f" is in one cluster ,rest all the 213 species are in other 2 clusters.This method is more sensitive to outliers.

```
distance = dist(data) #compute distance matrix
```

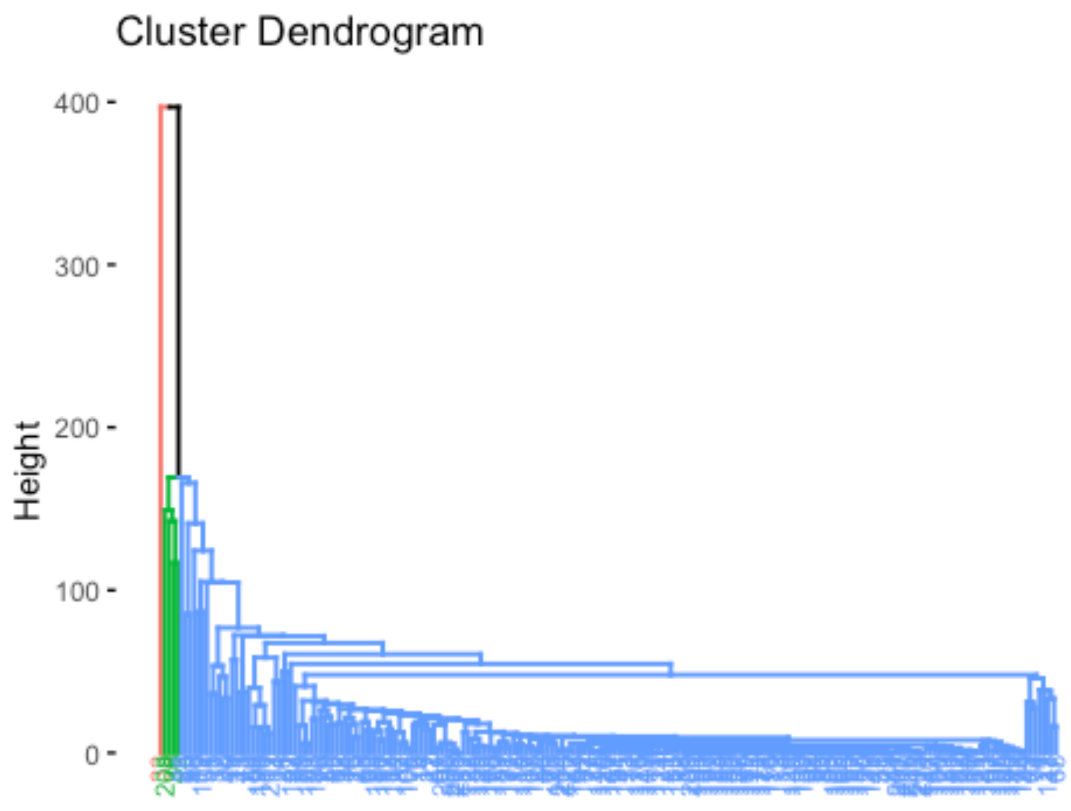
Hierarchical agglomerative clustering:

```
data.hclust = hclust(distance)
hcd=as.dendrogram(data.hclust)

par(mfrow=c(1,1))

plot(hcd, main="Main")
```



Hierarchical agglomerative clustering using "average" linkage:

```
data.hclust=hclust(distance,method="average")
fviz_dend(data.hclust,k=3, cex = 0.6,sub = "")
```



Cluster membership:

```
member = cutree(data.hclust,3)
table(member)

## member
##   1   2   3
## 209   4   1
```

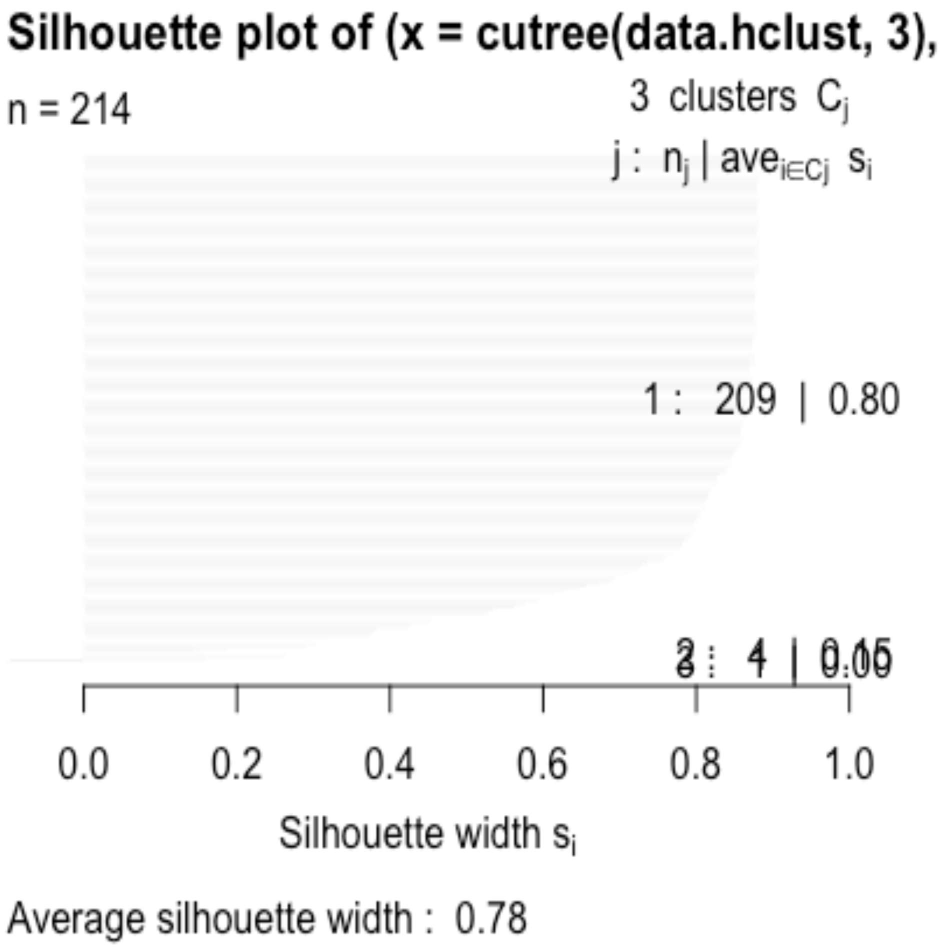
This table provides information regarding no. of plants species in each cluster.

Characterizing clusters:

```
aggregate(data,list(member),mean)

##   Group.1 Malshej_Ghat  Naneghat Bhimashankar Malegaon_Budruk    Dahuli
## 1       1     2.712919  2.822967     3.464115        2.980861  3.703349
## 2       2    12.750000 25.250000    61.000000       16.250000  5.000000
## 3       3    24.000000 22.000000   311.000000       13.000000 16.000000
##    Lonavala Tailbaila   Mulshi Tamhini_Ghat Gunjavane_dam   Abhepuri
## 1  2.483254  3.425837 2.875598     3.435407      2.157895   3.014354
## 2 20.500000 20.000000 6.250000    20.500000      8.750000  22.500000
## 3 44.000000 40.000000 0.000000    25.000000      8.000000 111.000000
##   Mahabaleshwar
## 1      3.751196
## 2    127.000000
## 3    221.000000
```

Silhouette Plot

```
library(cluster)
plot(silhouette(cutree(data.hclust,3), distance))
```



Based on the above plot, if any bar comes as negative side then we can conclude particular data is an outlier.It is clearly observed that the species of 3rd cluster is an outlier .

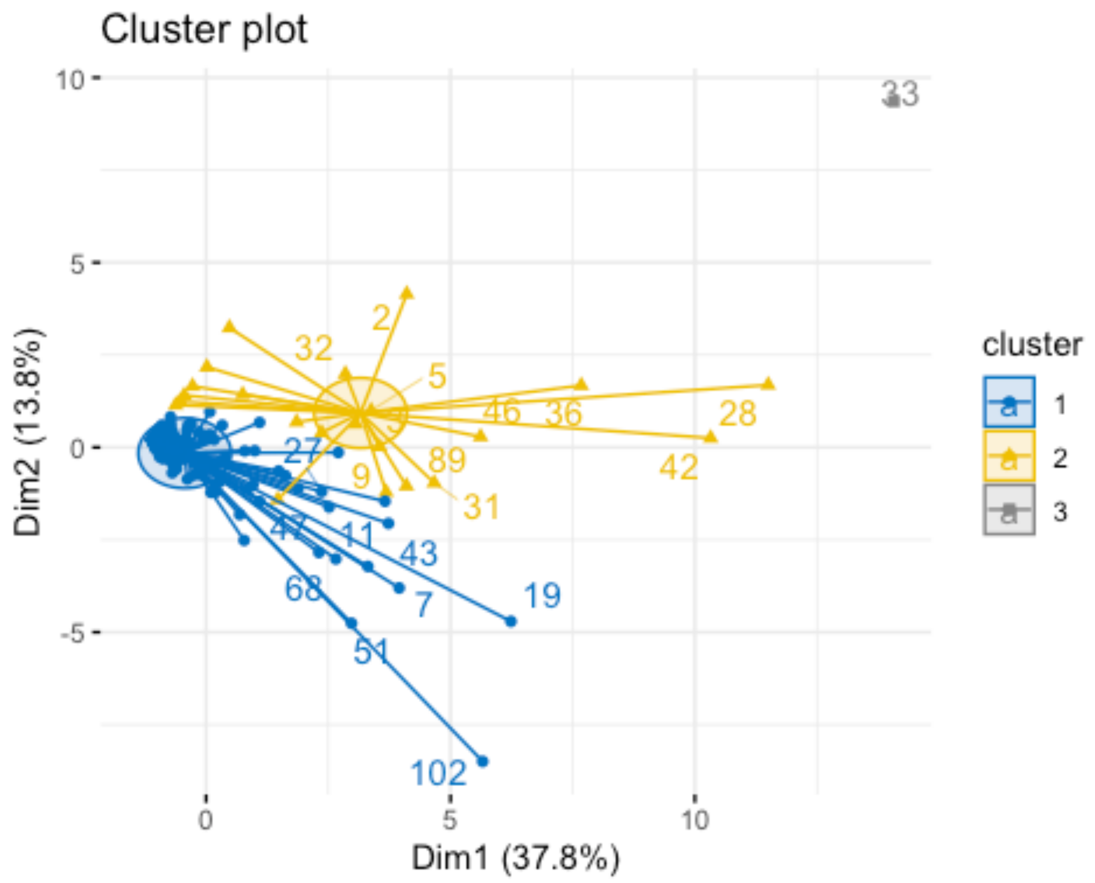
Comment:

K-means clustering is a very simple and fast algorithm and it can efficiently deal with very large data sets.K-means clustering needs to provide a number of clusters as an input, Hierarchical clustering is an alternative approach that does not require that we commit to a particular choice of clusters. Hierarchical clustering has an added advantage over K-means clustering because it has an attractive tree-based representation of the observations (dendrogram).

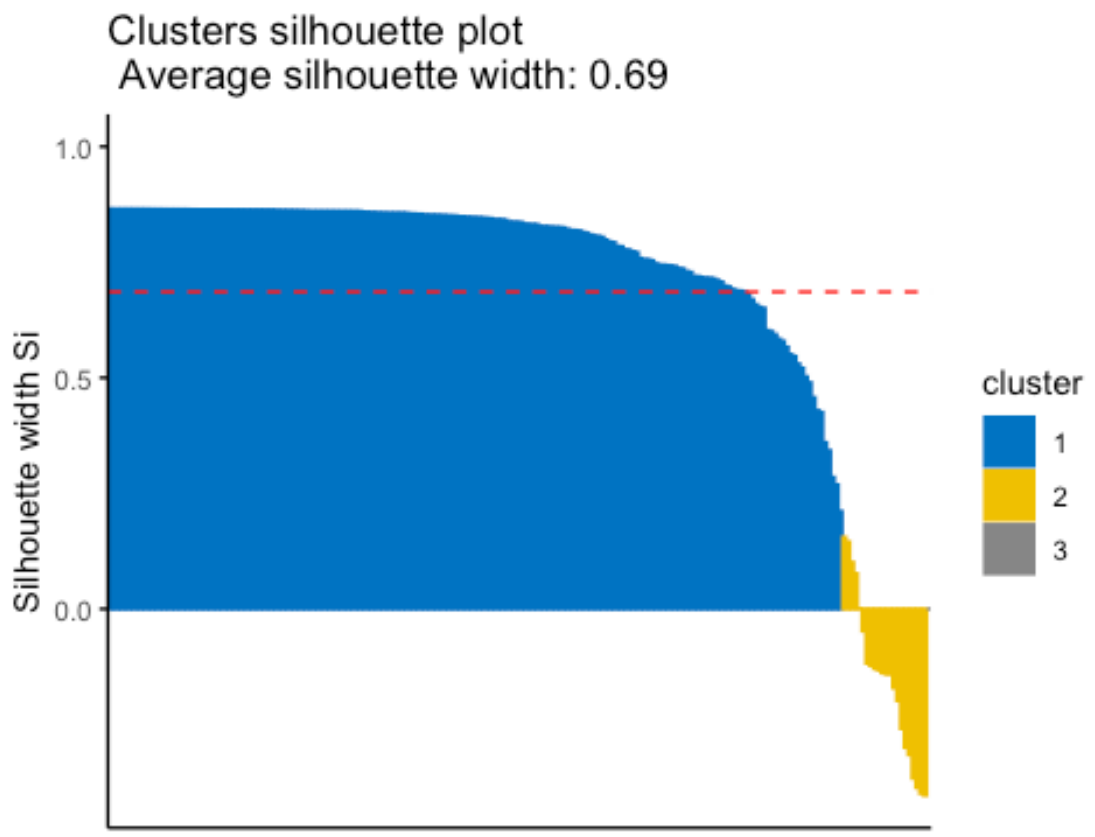
K-medoids:

K-medoid is a robust alternative to k-means clustering. This means that, the algorithm is less sensitive to noise and outliers, compared to k-means, because it uses medoids as cluster centers instead of means (used in k-means).

```
library(cluster)
library(factoextra)
k=3  ## no. of optimal cluster
PAM=pam(data, k,metric = "euclidean", stand = FALSE)
fviz_cluster(PAM, data = data, palette ="jco", ellipse.type = "euclid",
             star.plot = TRUE,
             repel = TRUE,
             ggtheme = theme_minimal())

## Too few points to calculate an ellipse

## Warning: ggrepel: 193 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```
fviz_silhouette(PAM, palette ="jco", ggtheme = theme_classic())

##   cluster size ave.sil.width
## 1       1  191          0.79
## 2       2   22         -0.15
## 3       3    1          0.00
```



Our 3-cluster solution gives an average silhouette coefficient of 0.69. Values close to 1 indicates that object is well clustered. One of the objects in cluster 2 has a negative silhouette coefficient indicating it may be in the wrong cluster. From this cluster plot we can clearly observe that 33rd species i.e,“Memecylon umbellatum Burm.f” is an outlier.

NOTE:-

The K-medoids algorithm, PAM, is a robust alternative to k-means for partitioning a data set into clusters of observation.

In k-medoids method, each cluster is represented by a selected object within the cluster. The selected objects are named medoids and corresponds to the most centrally located points within the cluster.

SELF ORGANISING MAPS(SOM) FOR CLUSTERING:

SOMs is less prone to local optima than k-means. During tests it is quite evident that the search space is better explored by SOMs. If we are interested in clustering then k-means obviously is not the best choice, because it is sensitive to initialization. SOMs provide a more robust learning. k-means is more sensitive to the noise present in the dataset compared to SOMs.

```
library(aweSOM)
train.data = data[, c(2,5,6,9)]  ## Select variables
train.data =scale(train.data)  ### Scale training data

## Train SOM
data.som = kohonen::som(train.data, grid = kohonen::somgrid(4, 4, "hexagonal"
), rlen = 100, alpha = c(0.05, 0.01), radius=c(2.65,-2.65),dist.fcts = "sumof
squares")

#Assessing the quality of the map
somQuality(data.som, train.data)

##
## ## Quality measures:
##  * Quantization error     :  0.4374883
##  * (% explained variance) :  89.01
##  * Topographic error      :  0.09813084
##  * Kaski-Lagus error      :  1.485165
##
## ## Number of obs. per map cell:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
##   9   2   4   3  11   4   3   3  19   5   2   7 117   8   6  11
```

Quantization error: Average squared distance between the data points and the map's prototypes to which they are mapped. Lower is better.

Percentage of explained variance: Similar to other clustering methods, the share of total variance that is explained by the clustering (equal to 1 minus the ratio of quantization error to total variance). Higher is better.

Topographic error: Measures how well the topographic structure of the data is preserved on the map. It is computed as the share of observations for which the best-matching node is not a neighbor of the second-best matching node on the map. Lower is better. 0 indicates excellent topographic representation (all best and second-best matching nodes are neighbors), 1 is the maximum error (best and second-best nodes are never neighbors).

Kaski-Lagus error: Combines aspects of the quantization and topographic error. It is the sum of the mean distance between points and their best-matching prototypes, and of the mean geodesic distance (pairwise prototype distances following the SOM grid) between the points and their second-best matching prototype.

It is common to further cluster the SOM map into superclasses, groups of cells with similar profiles. This is done using classic clustering algorithms on the map's prototypes.

Two methods are implemented in the web-based interface, PAM (k-medians) and hierarchical clustering. here, we choose 3 superclasses. We will return to the choice of superclasses below:

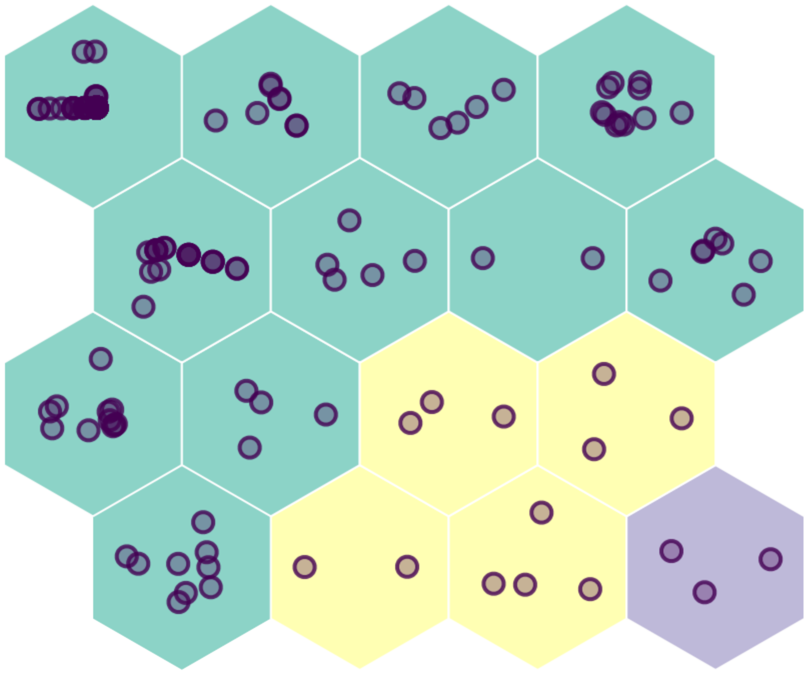
```
superclust_pam=cluster::pam(data.som$codes[[1]], 3)
superclasses_pam= superclust_pam$clustering
```

```
superclust_hclust=hclust(dist(data.som$codes[[1]]), "complete")
superclasses_hclust=cutree(superclust_hclust, 3)
```

Plotting general map information:

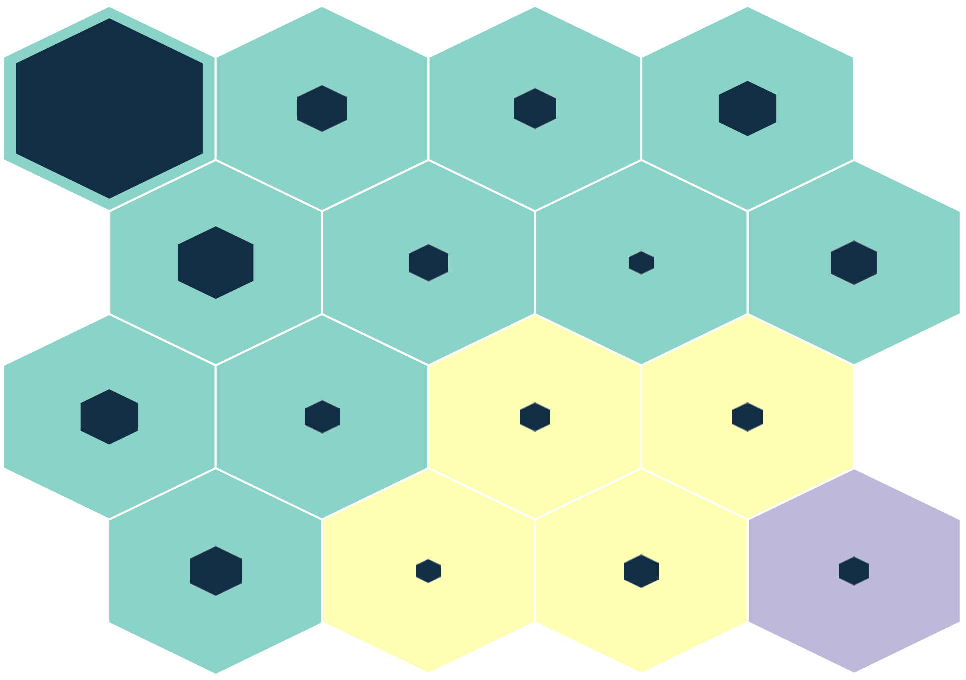
1)Observations Cloud : Visualize the observations as points inside their cell. Points can optionally be colored according to a variable of the data, as here by the Species of the plants (a variable not used during training). Other variables can be displayed in an interactive title box for each observation.

```
aweSOMplot(som = data.som,type="Cloud",data=data,superclass=superclass_pam)
```



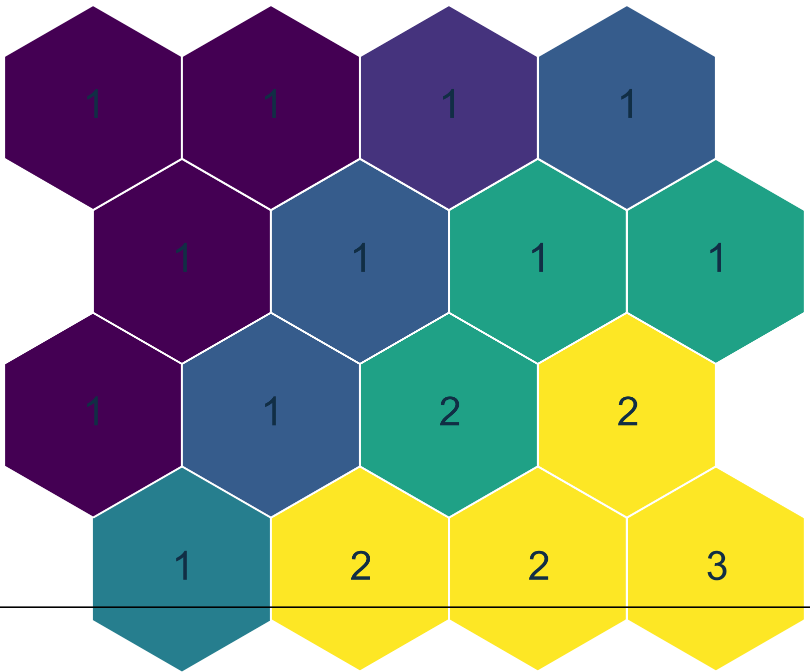
2) Hitmap or population map : Visualizes the number of observation per cell. The areas of the inner shapes are proportional to their population. The background colors indicate the superclasses. The palette of these colors can be adapted using the “palsc” argument.

```
aweSOMplot(som = data.som,type="Hitmap",data=data,superclass=superclass_pam)
```



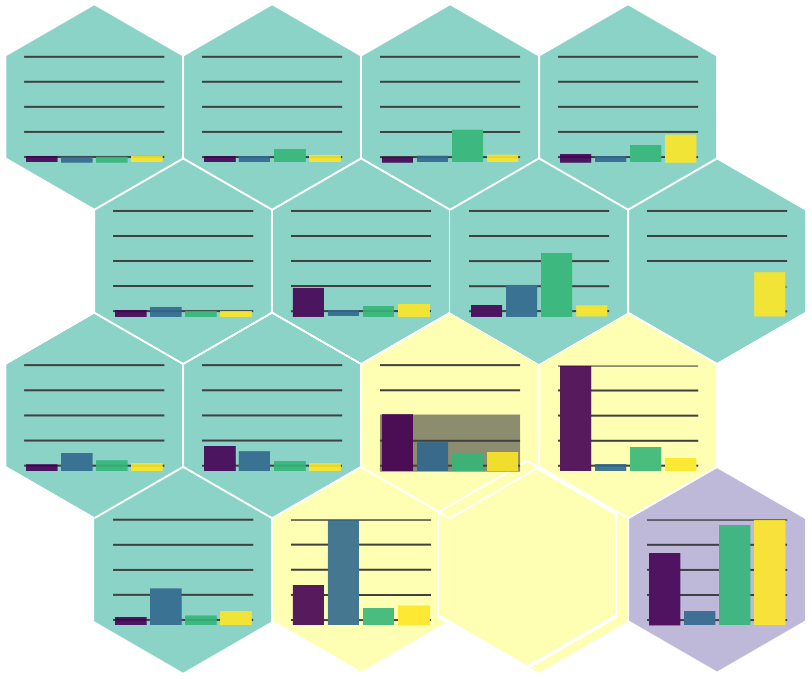
3) UMatrix : It is a way to explore the topography of the map. It shows the average distance between each cell and its neighbors’ prototypes, on a color scale. On this map, the darker cells are close to their neighbors, while the brighter cells are more distant from their neighbors.

```
aweSOMplot(som = data.som,type="UMatrix",data=data,superclass=superclass_pam)
```



On the following barplot, we plot the protoype values instead of the observations means.

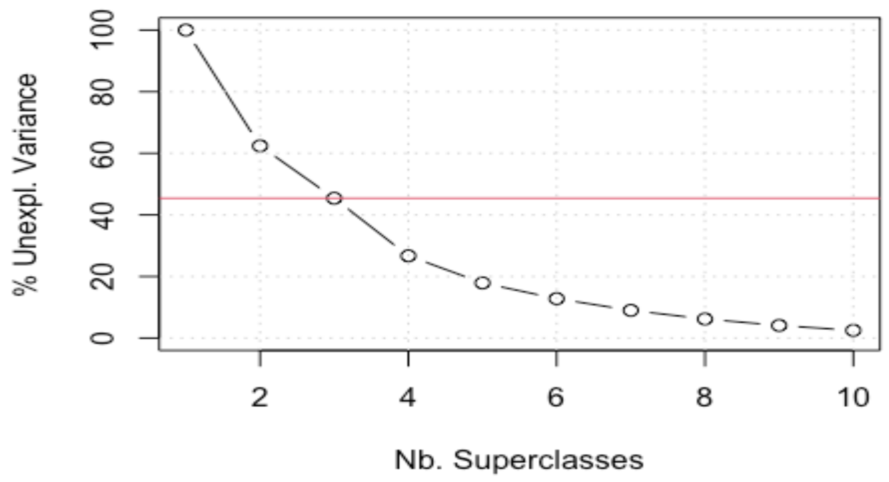
```
aweSOMplot(som = data.som,type="Barplot",data=data,superclass=superclass_pam,
values="prototypes")
```



Choosing the number of superclasses:

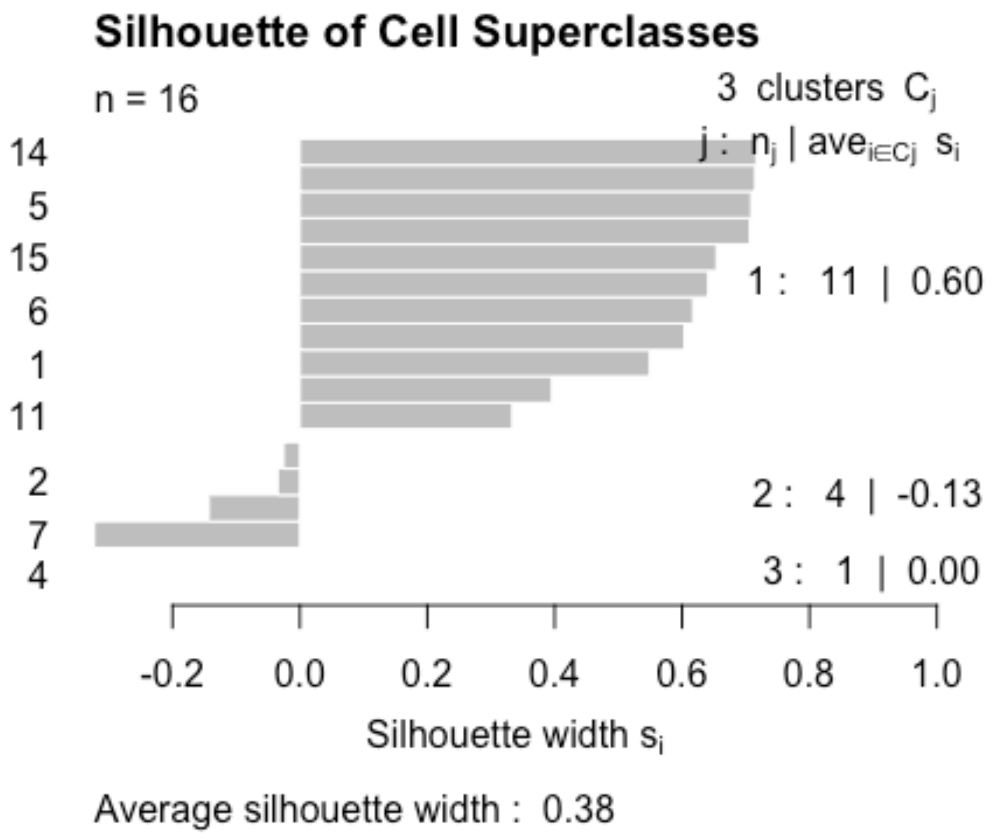
aweSOM offers three diagnostics plot for choosing the number of superclasses. “aweSOMscreeplot” produces a scree plot, that shows the quality of the clustering (percentage of unexplained variance of the prototypes, lower is better) for varying numbers of superclasses. It supports hierarchical and pam clustering. The rule of thumb is to choose the number of superclasses at the inflection point of this curve.

```
aweSOMscreeplot(som = data.som, method = "pam", nclass = 3)
```



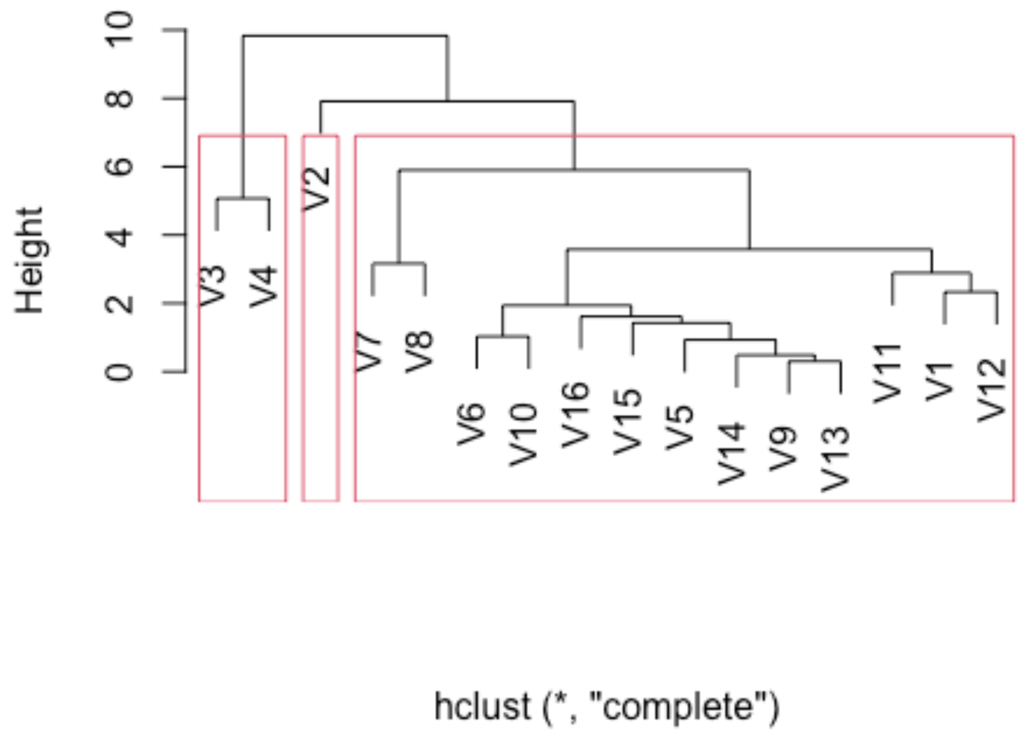
aweSOMsilhouette returns a silhouette plot of the chosen superclasses (hierarchical, pam, or other). The higher the silhouettes, the better.

```
aweSOMsilhouette(data.som, superclasses_pam)
```



For hierarchical clustering, "aweSOMdendrogram" produces a dendogram, along with the chosen cuts.

```
aweSOMdendrogram(clust = superclust_hclust, nclass = 3)
```



MULTI-DIMENSIONAL SCALING:

Multidimensional scaling (MDS) is a multivariate data analysis approach that is used to visualize the similarity/dissimilarity between samples by plotting points in two dimensional plots.

MDS returns an optimal solution to represent the data in a lower-dimensional space, where the number of dimensions k is pre-specified by the analyst. For example, choosing k = 2 optimizes the object locations for a two-dimensional scatter plot.

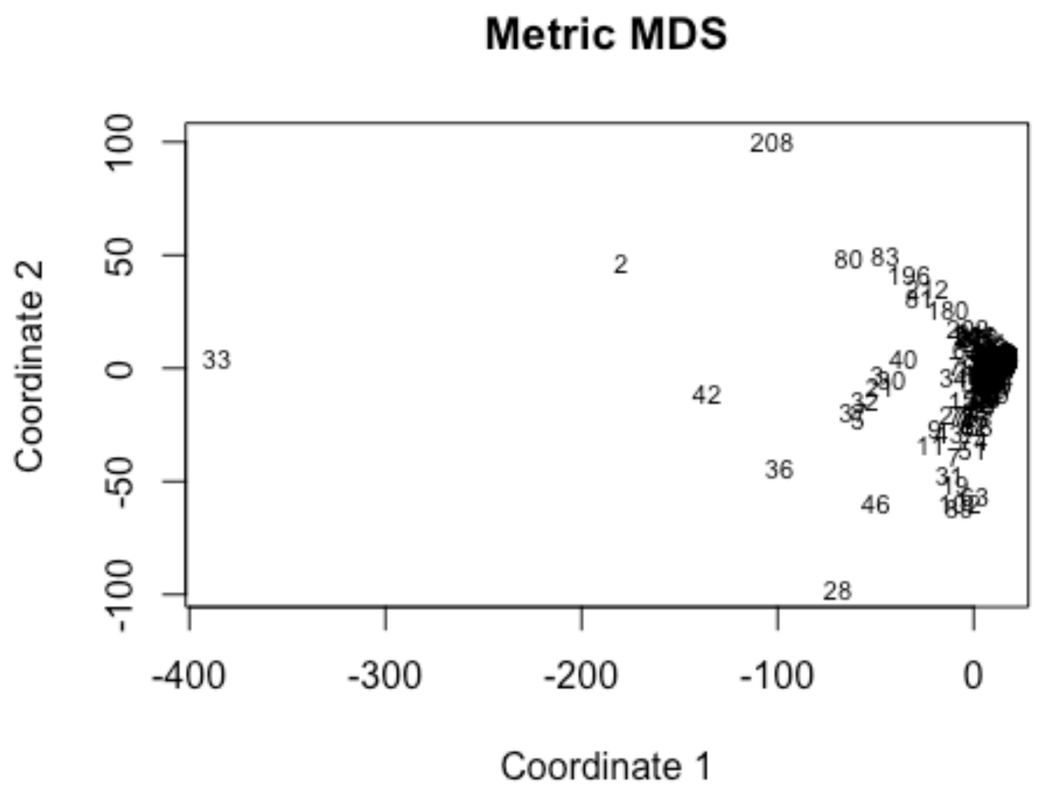
Compute MDS:

```
library(MASS)

##
## Attaching package: 'MASS'

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

```
library(magrittr)

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract

library(dplyr)
library(ggpubr)
library(MASS)

Dis=as.matrix(dist(D)) # euclidean distances between the rows
```

After that, we run Multidimensional Scaling (MDS) with function cmdscale(), and get x and y coordinates.

```
mds=cmdscale(Dis,eig=TRUE, k=2) # k is the number of dim
x = mds$points[,1]
y = mds$points[,2]
plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
     main="Metric MDS", type="n")
text(x, y, cex=.7,xpd=TRUE)
```



Comment-Then we visualise the result, which shows the positions of plant species are very close to their relative locations on a map.

Association Rule Mining for species:

```
library(arules)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode
## The following objects are masked from 'package:base':
##
##     abbreviate, write

library(arulesViz)
```

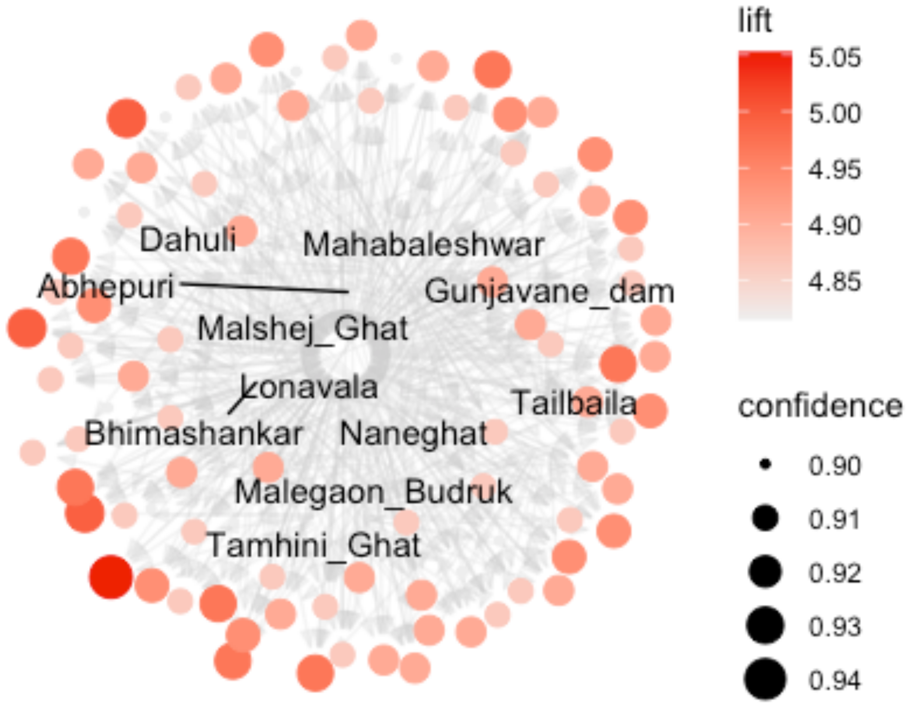
Fitting model Training Apriori on the dataset

```
set.seed = 20   #Setting seed
rules =apriori(data = data,
               parameter = list(support = 0.004,
                                confidence = 0.2))

## Warning in asMethod(object): matrix contains values other than 0 and 1! Se
tting
## all entries != 0 to 1.

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5   0.004      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 0
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[12 item(s), 214 transaction(s)] done [0.00s].
## sorting and recoding items ... [12 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10

## Warning in apriori(data = data, parameter = list(support = 0.004, confiden
ce
## = 0.2)): Mining stopped (maxlen reached). Only patterns up to a length of
10
## returned!

##  done [0.00s].
## writing ... [24430 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

rules.sorted=sort(rules, by="lift")
```

Visualising the results:

```
inspect(sort(rules, by = 'lift')[1:214])
```

```
##       lhs                       rhs                 support confidence   coverag
e     lift count
## [1]   {Naneghat,
##        Bhimashankar,
##        Lonavala,
##        Tamhini_Ghat}     => {Malshej_Ghat} 0.07943925  0.9444444 0.0841121
5 5.052778    17
## [2]   {Bhimashankar,
##        Lonavala,
##        Mahabaleshwar}    => {Malshej_Ghat} 0.06542056  0.9333333 0.0700934
6 4.993333    14
## [3]   {Naneghat,
##        Bhimashankar,
##        Dahuli,
##        Lonavala}         => {Malshej_Ghat} 0.06542056  0.9333333 0.0700934
6 4.993333    14
## [4]   {Naneghat,
##        Bhimashankar,
##        Dahuli,
##        Lonavala,
##        Tamhini_Ghat}     => {Malshej_Ghat} 0.06542056  0.9333333 0.0700934
6 4.993333    14
       ………………………………………………………………………………………………………

       ………………………………………………………………………………………………………

       ………………………………………………………………………………………………………


## [213] {Naneghat,
##        Bhimashankar,
##        Lonavala,
##        Mulshi}           => {Malshej_Ghat} 0.03271028  0.8750000 0.0373831
8 4.681250     7
## [214] {Bhimashankar,
##        Malegaon_Budruk,
##        Lonavala,
##        Tamhini_Ghat}     => {Malshej_Ghat} 0.06542056  0.8750000 0.0747663
6 4.681250    14


plot(rules, method = "graph",
     measure = "confidence", shading = "lift")

## Warning: Too many rules supplied. Only plotting the best 100 using
## 'lift' (change control parameter max if needed).
```



The network graph shows associations between selected locations. Larger circles imply higher support, while red circles imply higher lift. Graphs only work well with very few rules.

CLUSTERING FOR LOCATIONS-{PART B}

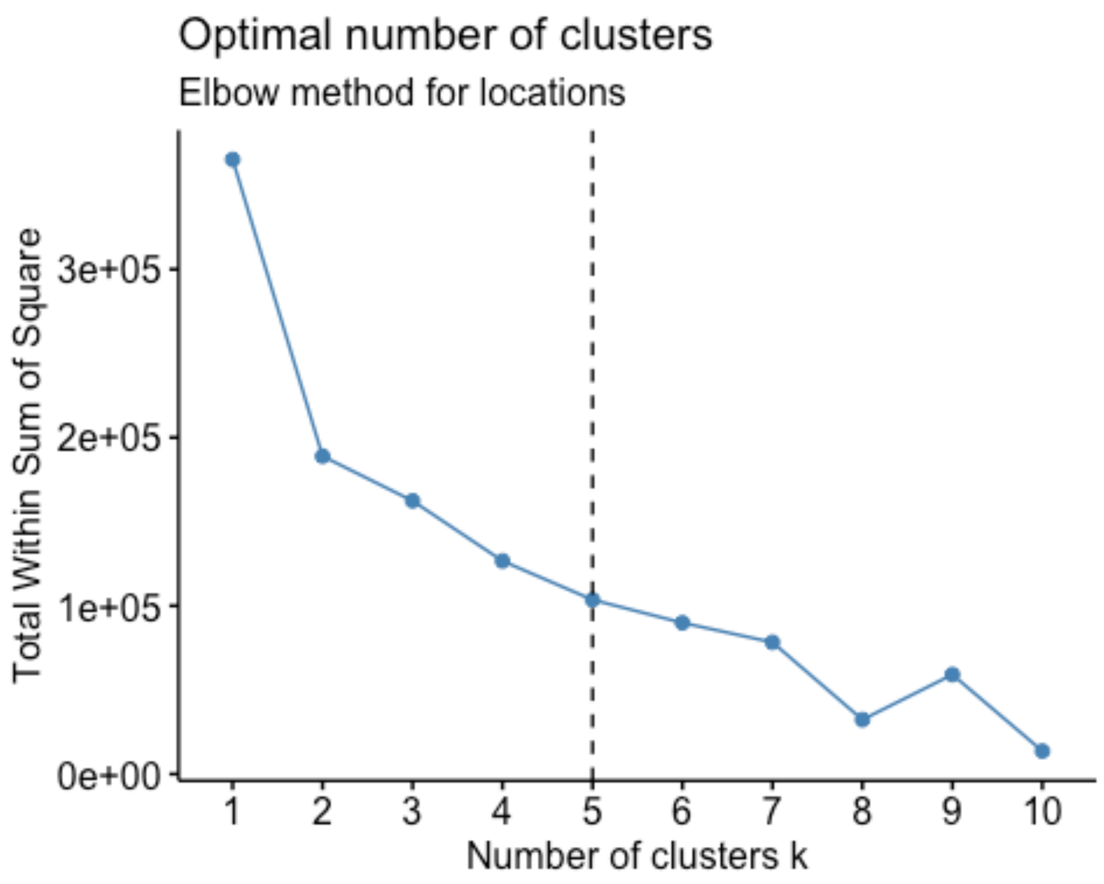
```
locationdata=t(D)    #our data for loctions as observations
ldata=t(data)  #numeric data final
A=t(data)
```

Finding Optimal no of clusters using three approaches:

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

Elbow method

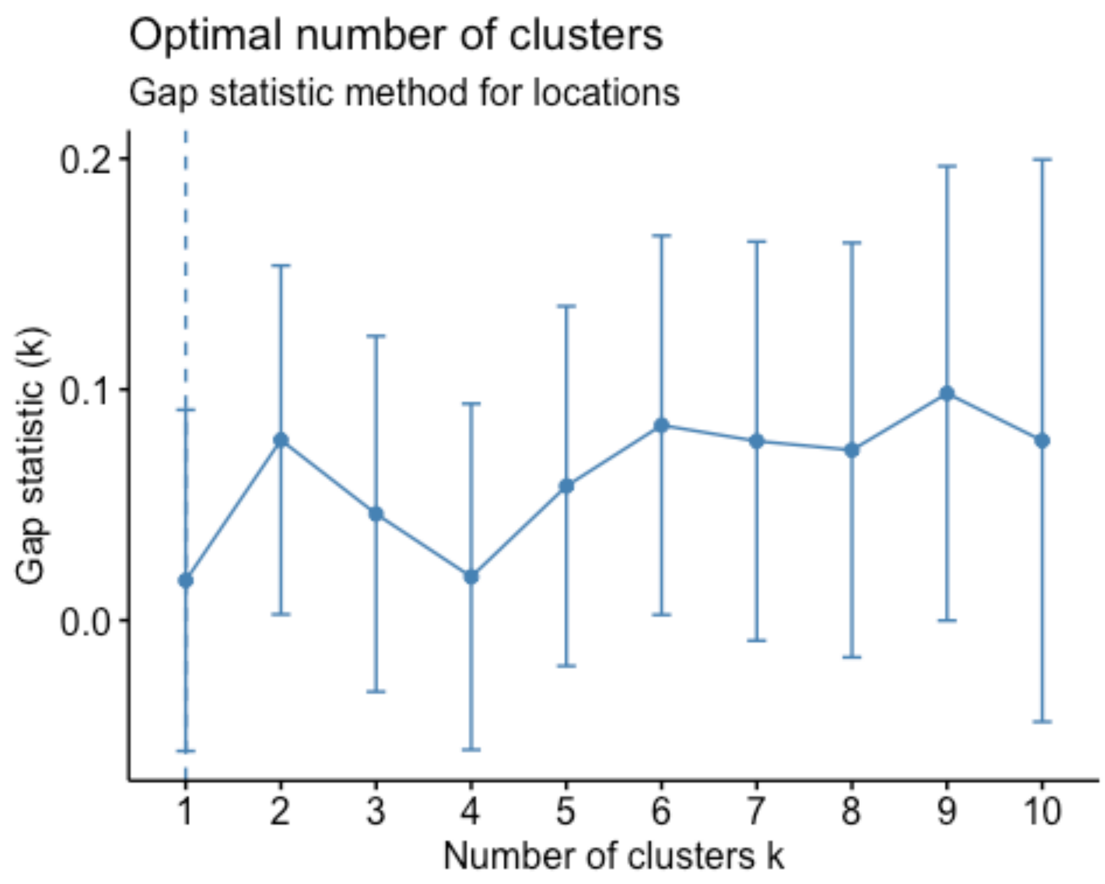
```
fviz_nbclust(locationdata, kmeans, method = "wss") +geom_vline(xintercept = 5
, linetype = 2)+labs(subtitle = "Elbow method for locations")
```



Gap statistic

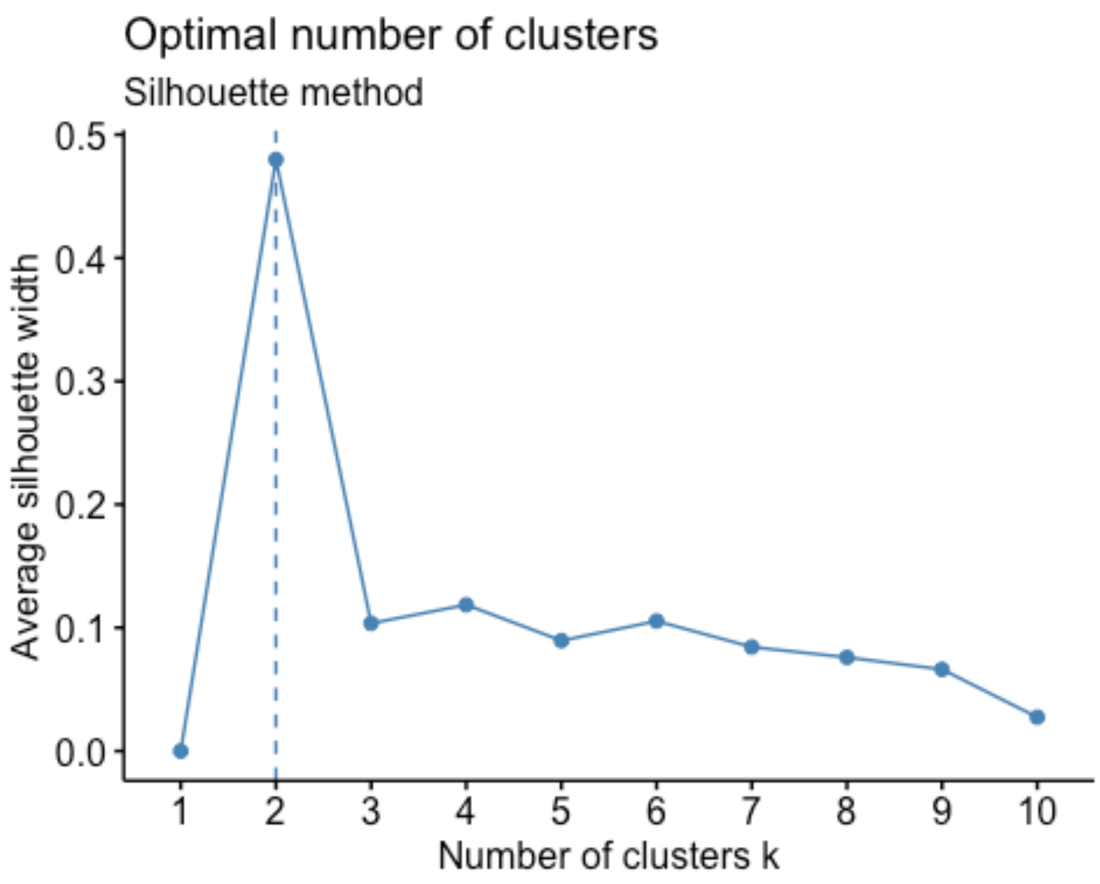
nboot = 50 to keep the function speedy. recommended value: nboot= 50 for our analysis.

```
set.seed(123)
fviz_nbclust(ldata, kmeans, nstart = 25,  method = "gap_stat", nboot = 50)+
  labs(subtitle = "Gap statistic method for locations")
```



Silhouette method

```
fviz_nbclust(locationdata, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



Elbow method: 5 clusters solution suggested.

Silhouette method: 2 clusters solution suggested.

Gap statistic method: 1 clusters solution suggested.

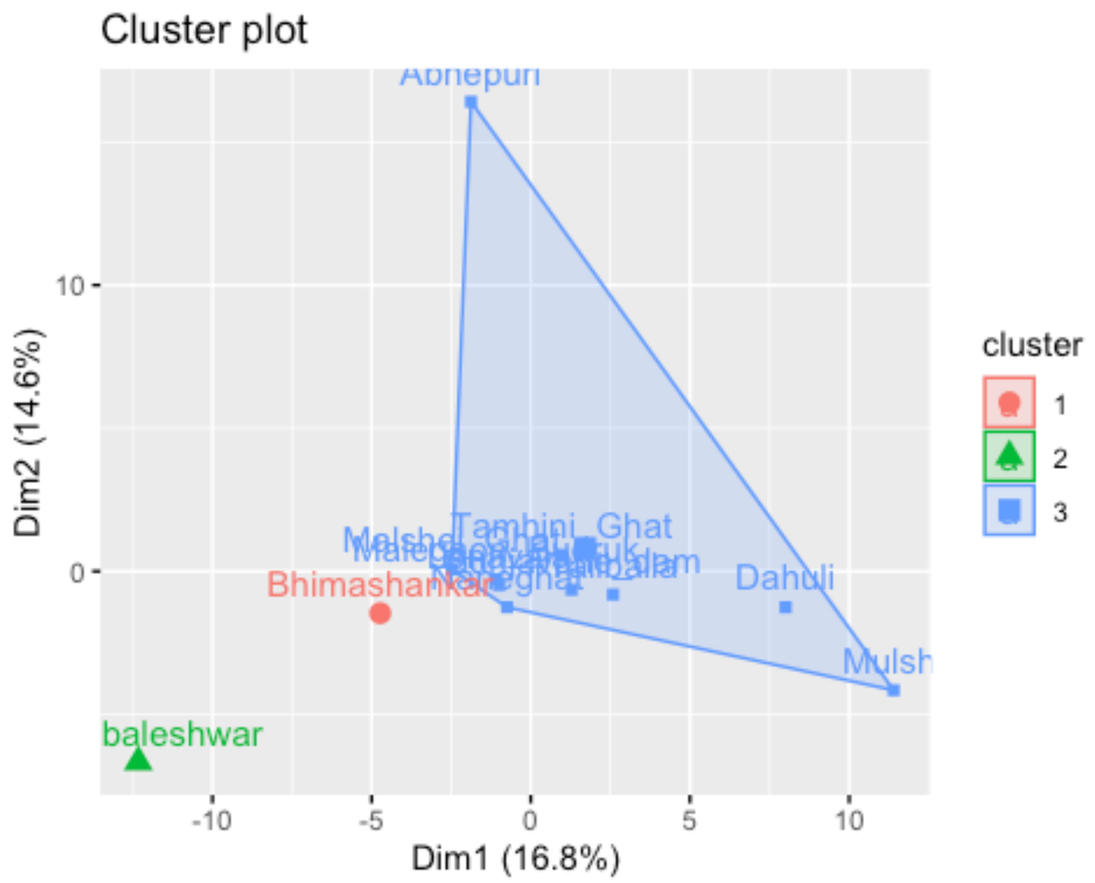
NOTE:- The disadvantage of elbow and average silhouette methods is that, they measure a global clustering characteristic only. A more sophisticated method is to use the gap statistic which provides a statistical procedure to formalize the elbow/silhouette heuristic in order to estimate the optimal number of clusters.

```
B=ldata[,which(colSums(ldata)!=0)]

#our data after removing columns with sum=0
```

kmeans clustering in R for locations

```
library(factoextra)
k1= kmeans(B, centers = 3, nstart = 25)
str(k1)
```

```
## List of 9
##  $ cluster     : Named int [1:12] 3 3 1 3 3 3 3 3 3 3 ...
##   ..- attr(*, "names")= chr [1:12] "Malshej_Ghat" "Naneghat" "Bhimashankar
" "Malegaon_Budruk" ...
##  $ centers     : num [1:3, 1:211] 0 0 0.4 138 140 7.2 23 51 12.1 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "1" "2" "3"
##   .. ..$ : NULL
##  $ totss       : num 365197
##  $ withinss    : num [1:3] 0 0 143458
##  $ tot.withinss: num 143458
##  $ betweenss   : num 221740
##  $ size        : int [1:3] 1 1 10
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

fviz_cluster(k1, data = B)
```

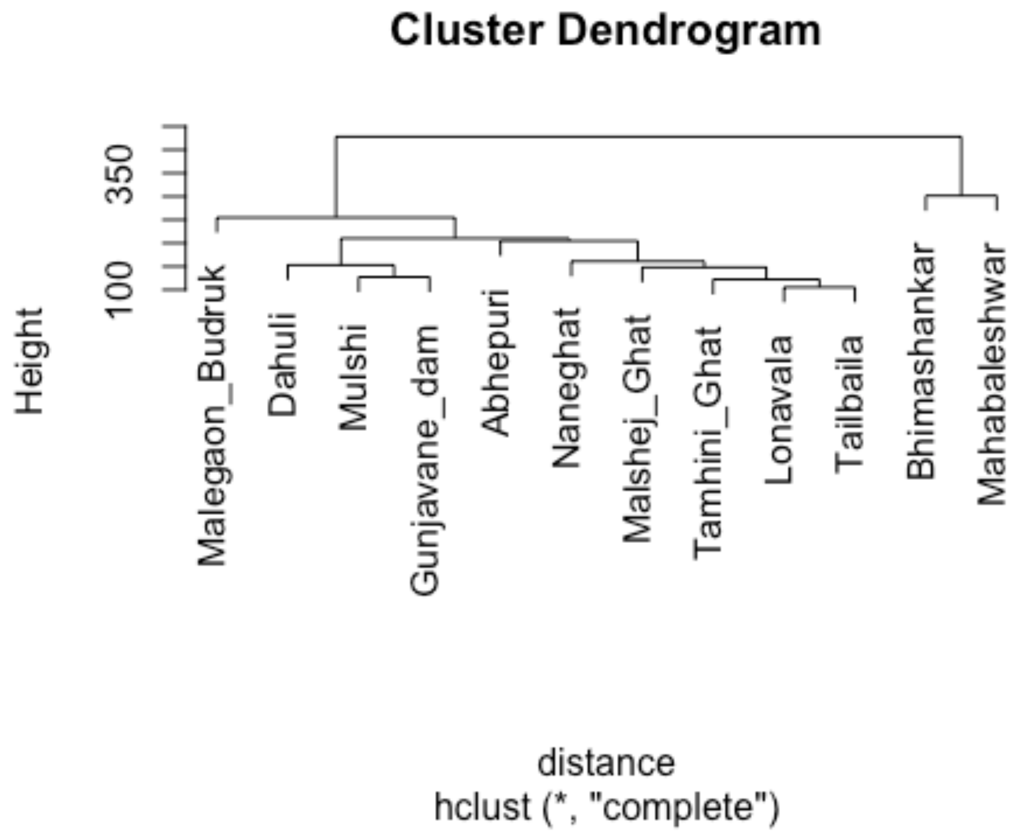


Comment-: From the cluster plot we can clearly observe that Mahabaleshwar and Bhimashankar loctions differ significantly from remaining 10 locations. Here, we are viewing kmeans results by using fviz_cluster. This provides a beautiful illustration of the clusters.

```
distance = dist(ldata) #compute distance matrix
```

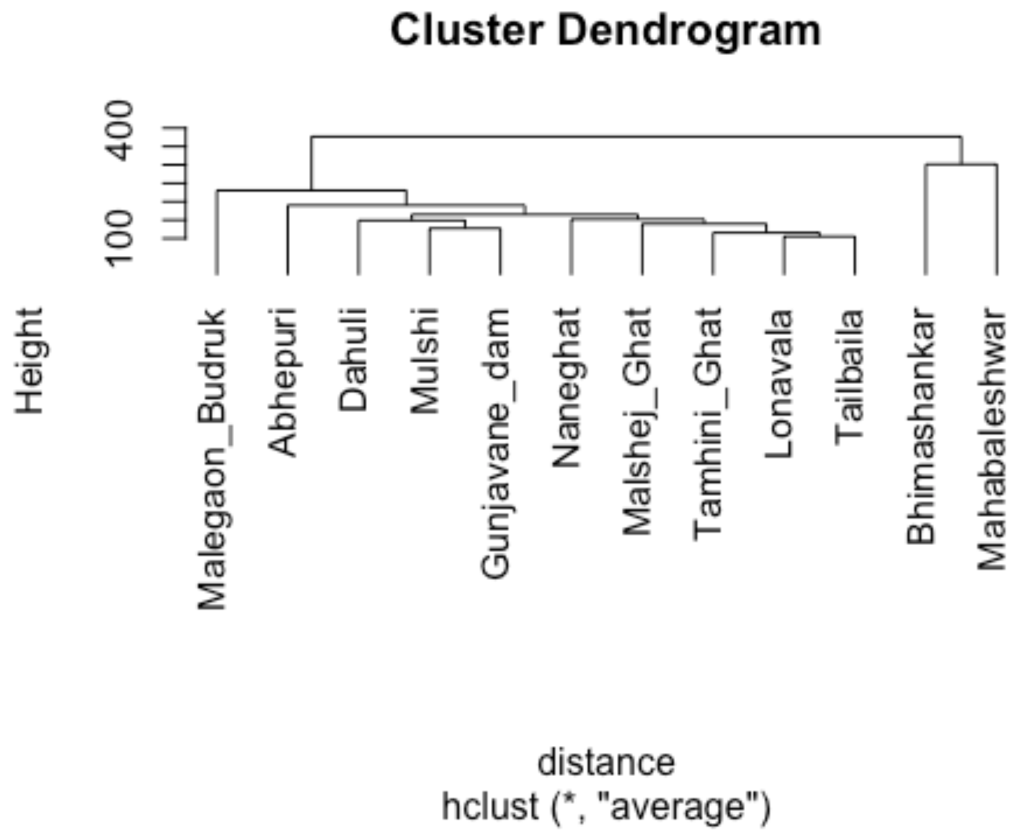
Hierarchical agglomerative clustering:

```
data2.hclust = hclust(distance)
plot(data2.hclust)
```



Hierarchical agglomerative clustering using "average" linkage:

```
data2.hclust=hclust(distance,method="average")
plot(data2.hclust,hang=-1)
```



Cluster membership

```
member = cutree(data2.hclust,3)
table(member)

## member
##  1  2  3
## 10  1  1
```

Characterizing clusters

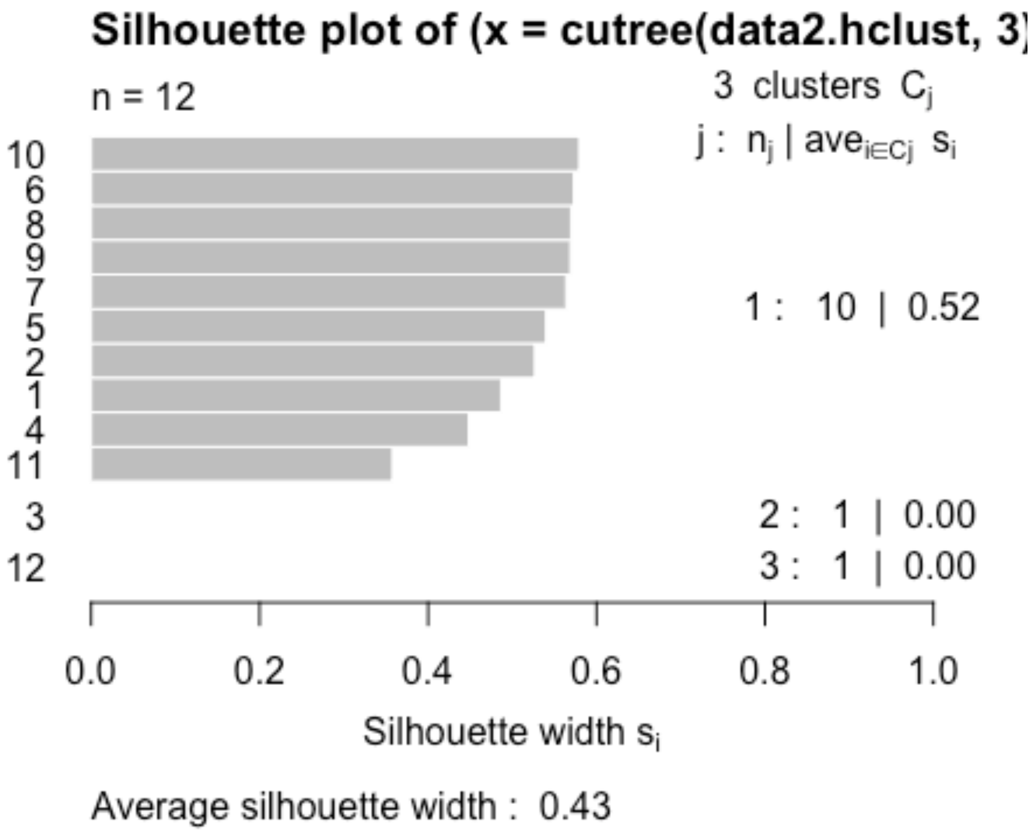
```
aggregate(ldata,list(member),mean)

##   Group.1  V1    V2   V3  V4   V5  V6   V7  V8   V9 V10  V11 V12 V13 V14
V15
## 1       1 0.4   7.2 12.1 0.1 10.8 3.2 17.7 2.3 12.7 0.3 14.8 1.7 0.2 3.7
1.3
## 2       2 0.0 138.0 23.0 0.0 73.0 0.0 10.0 5.0 29.0 0.0 27.0 2.0 0.0 0.0 1
8.0
## 3       3 0.0 140.0 51.0 0.0 17.0 0.0  1.0 0.0  0.0 0.0  0.0 0.0 0.0 0.0
0.0
##    V16 V17 V18  V19 V20  V21 V22 V23 V24 V25 V26  V27  V28 V29  V30  V31
V32
## 1  3.7   4 5.2 22.5 0.1 11.6 4.6 7.3 9.1   0 9.8 11.1 34.7 5.9  7.3 18.9
9.9
```

```
## 2 15.0   0 0.0  0.0 0.0 30.0 0.0 0.0 0.0   0 0.0  8.0 57.0 0.0 46.0  7.0 5
9.0
## 3  0.0   4 0.0  4.0 0.0 43.0 0.0 0.0 0.0   0 0.0  6.0  0.0 0.0 24.0  0.0 2
5.0
##     V33  V34 V35  V36  V37 V38 V39 V40 V41   V42  V43 V44 V45  V46  V47 V4
8 V49
## 1  30.3  2.8 0.1 25.5  8.1 2.6 0.2   3   0  30.4 14.8   0   0 23.5 11.6 0.
1 7.6
## 2 311.0 26.0 0.0 64.0 84.0 0.0 0.0  43   1  42.0 12.0   0   0 41.0  0.0 0.
0 0.0
## 3 221.0  2.0 0.0 58.0 15.0 0.0 0.0  23   0 132.0  4.0   0   0  0.0  4.0 2.
0 0.0
##   V50  V51 V52 V53 V54 V55 V56 V57 V58 V59 V60  V61  V62  V63 V64 V65 V66
V67
## 1   6 17.6   4 0.4   2 0.6 1.7 0.6 7.5   1 5.3  0.4 11.6 18.2 0.5 6.8 2.9
2.4
## 2   0  0.0   0 0.0   1 0.0 0.0 4.0 5.0   0 0.0  0.0  0.0  0.0 0.0 0.0 0.0
0.0
## 3   0  0.0   0 0.0   0 0.0 0.0 0.0 1.0   0 0.0 21.0  0.0  0.0 0.0 0.0 0.0
19.0
##    V68 V69 V70 V71 V72 V73  V74 V75  V76 V77 V78  V79 V80 V81 V82  V83 V84
V85
## 1 12.2 1.8 2.9 0.3 6.6 2.9 14.1 1.7  9.1   0   0  0.3   0   0 3.3  0.1 0.1
3.1
## 2  0.0 1.0 0.0 0.0 0.0 0.0  0.0 1.0 12.0   4   5 25.0  29  11 6.0  4.0 0.0
0.0
## 3  0.0 0.0 0.0 0.0 0.0 0.0  0.0 0.0  0.0   0   0  0.0  87  50 0.0 84.0 0.0
0.0
##   V86 V87 V88  V89  V90 V91 V92 V93 V94 V95 V96 V97 V98 V99 V100 V101 V102
V103
## 1   2 1.9 0.3 20.9  0.9 0.8 0.2 0.1 3.8 0.1 0.1 1.2 0.1 4.1    6  2.6   27
1.4
## 2   0 0.0 0.0  0.0  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0    0  0.0    0
0.0
## 3   0 0.0 2.0  3.0 18.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0    0  0.0    1
0.0
##   V104 V105 V106 V107 V108 V109 V110 V111 V112 V113 V114 V115 V116 V117 V1
18
## 1  2.3  0.6    2  0.3  1.2  0.2  0.3  0.4  0.6  6.5  3.9  0.7  0.2  1.4  1
.1
## 2  0.0  0.0    0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
## 3  0.0  0.0    0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
##   V119 V120 V121 V122 V123 V124 V125 V126 V127 V128 V129 V130 V131 V132 V1
33
## 1  0.3  0.9  1.6    4  0.8  0.4  0.3  0.5  7.6  1.1  0.7  0.1  1.5  0.6  1
.5
## 2  0.0  0.0  0.0    0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
```

```
## 3  0.0  0.0  2.0    0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
##   V134 V135 V136 V137 V138 V139 V140 V141 V142 V143 V144 V145 V146 V147 V1
48
## 1  0.1  0.2  0.1  0.4  1.8  0.1  0.9  0.5  0.3  1.4  0.6  0.1  0.4    1  0
.5
## 2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0    0  0
.0
## 3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0    0  0
.0
##   V149 V150 V151 V152 V153 V154 V155 V156 V157 V158 V159 V160 V161 V162 V1
63
## 1  0.9  0.1  0.5  0.4  0.5  0.8  0.3  0.2  0.1  0.1  0.1  0.8  0.2  0.4  0
.1
## 2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
## 3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
##   V164 V165 V166 V167 V168 V169 V170 V171 V172 V173 V174 V175 V176 V177 V1
78
## 1  0.1  0.3  0.1  0.5  1.1  0.2  1.5  0.1  0.2  0.8  0.1  0.1  0.3  1.4  0
.2
## 2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
## 3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
##   V179 V180 V181 V182 V183 V184 V185 V186 V187 V188 V189 V190 V191 V192 V1
93
## 1  0.1  0.1  0.3  0.8  0.2  0.2  0.3  0.3  1.7  0.6  0.3  0.2  0.1  0.5  0
.1
## 2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
## 3  0.0 39.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0
.0
##   V194 V195 V196 V197 V198 V199 V200 V201 V202 V203 V204 V205 V206 V207 V2
08
## 1  0.7    1  0.2  0.2  3.3  0.2  0.1    0    0    0    0    0    0    0
0
## 2  0.0    0  0.0  0.0  0.0  0.0  0.0    0    0    0    0    0    0    0
0
## 3  0.0    0 69.0  0.0  0.0  0.0  0.0    5   23    2   10    1   17    4  1
78
##   V209 V210 V211 V212 V213 V214
## 1    0    0    0    0    0    0
## 2    0    0    0    0    0    0
## 3   15    3   17   55    8    2
```

Silhouette Plot

```
library(cluster)
plot(silhouette(cutree(data2.hclust,3), distance))
```



Comment- Based on the above plot,we can say that 3rd location i.e, Bhimashankar and 12th location i.e, Mahabaleshwar differ signficantly from other locations,Also, if any bar comes as negative side then we can conclude particular data is an outlier can remove from our analysis.

Conclusion:

K-means clustering is a very simple and fast algorithm and it can efficiently deal with very large data sets.K-means clustering needs to provide a number of clusters as an input, where as;

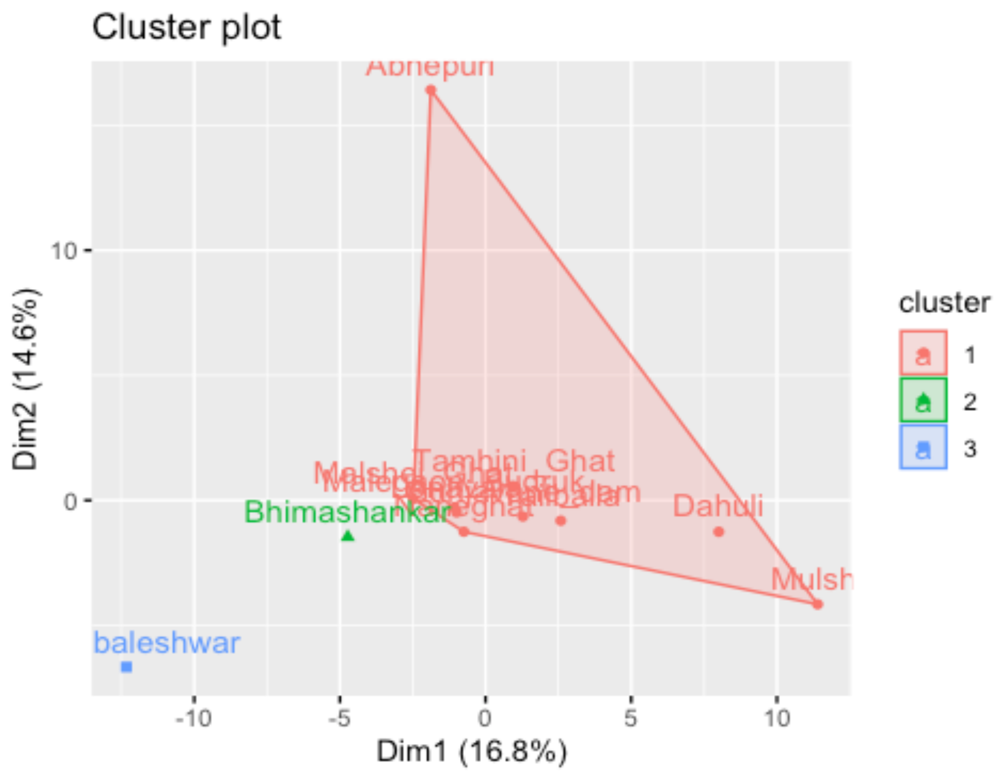
Hierarchical clustering is an alternative approach that does not require that we commit to a particular choice of clusters.Hierarchical clustering has an added advantage over K-means clustering because it has an attractive tree-based representation of the observations (dendrogram).

K-medoids:

K-medoids is a robust alternative to k-means clustering. This means that, the algorithm is less sensitive to noise and outliers, compared to k-means, because it uses medoids as cluster centers instead of means (used in k-means).

The K-medoids algorithm, PAM, is a robust alternative to k-means for partitioning a data set into clusters of observation.In k-medoids method, each cluster is represented by a selected object within the cluster. The selected objects are named medoids and corresponds to the most centrally located points within the cluster.

```
library(cluster)
library(factoextra)
k=3  ## no. of optimal cluster
PAM2=pam(B, k)
PAM2.med=PAM2$medoids
fviz_cluster(PAM2)
```



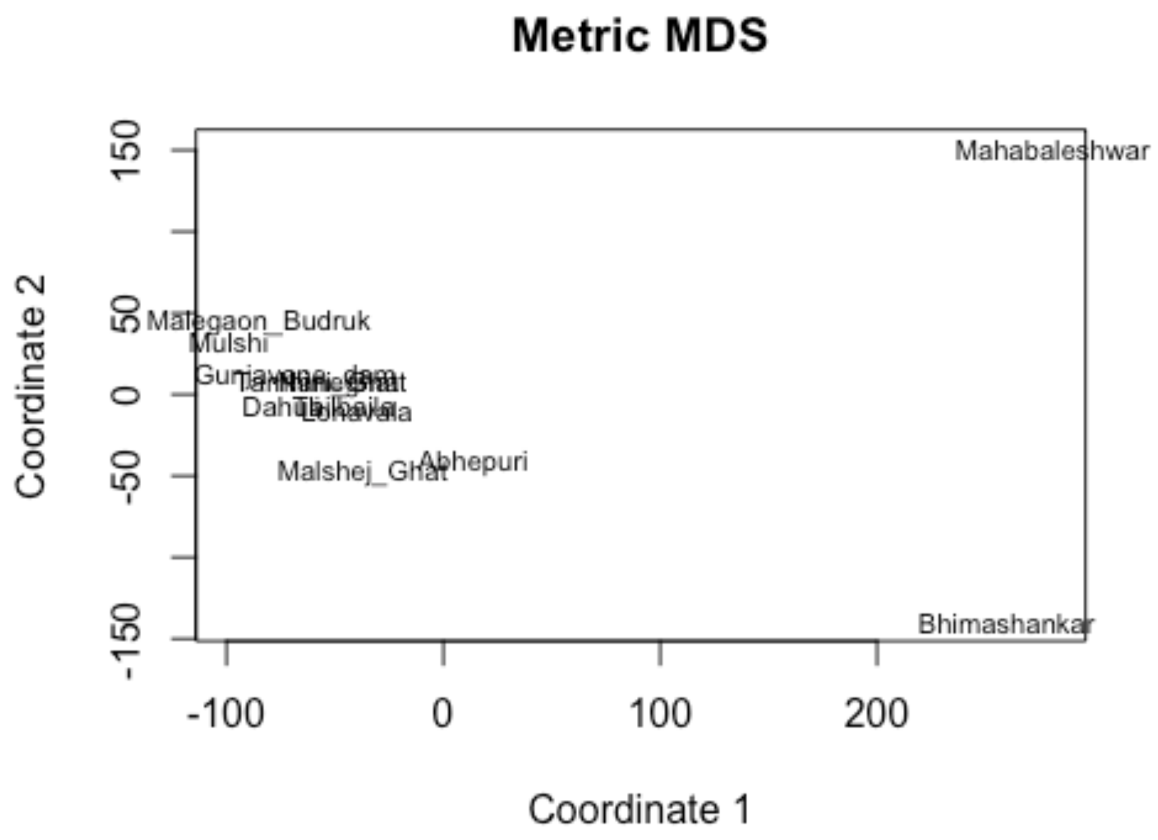
Comment-from this cluster plot we get the same visualisation results as of cluster plot from K-means.

MULTI-DIMENSIONAL SCALING(MDS) for locations:

```
library(MASS)
library(magrittr)
library(dplyr)
library(ggpubr)

par(mfrow=c(1,1))
Dist2=as.matrix(dist(locationdata)) # euclidean distances between the rows
```

After that, we run Multidimensional Scaling (MDS) with function cmdscale(), and get X and Y coordinates.s

```
mds2=cmdscale(Dist2,eig=TRUE, k=2) # k is the number of dim
X=mds2$points[,1]
Y=mds2$points[,2]
plot(X,Y, xlab="Coordinate 1", ylab="Coordinate 2",
     main="Metric MDS", type="n")
text(X,Y,labels = rownames(ldata),cex=.7,xpd=TRUE)
```



Comment-Based on the Metric MDS ,we can say that 3rd location i.e, Bhimashankar and 12th location i.e, Mahabaleshwar are dissimilar from remaining 10 locations.

CONCLUSION:

By applying various clustering methods we found out that-

1) In locations , “MAHABALESHWAR” lies in one cluster “BHIMASHANKAR” lies in second cluster “Malshej_Ghat”,“Naneghat”,“Malegaon_Budruk”,“Dahuli”, “Lonavala”,“Tailbaila”,“Mulshi”,“Tamhini_Ghat”, “Gunjavane_dam”,“Abhepuri”lies in third cluster.

2) In Plant Species,33rd plant species i.e,“Memecylon umbellatum Burm.f” is in one cluster(differ significantly from other plant sepcies) ,rest all the 213 species are in other 2 clusters.