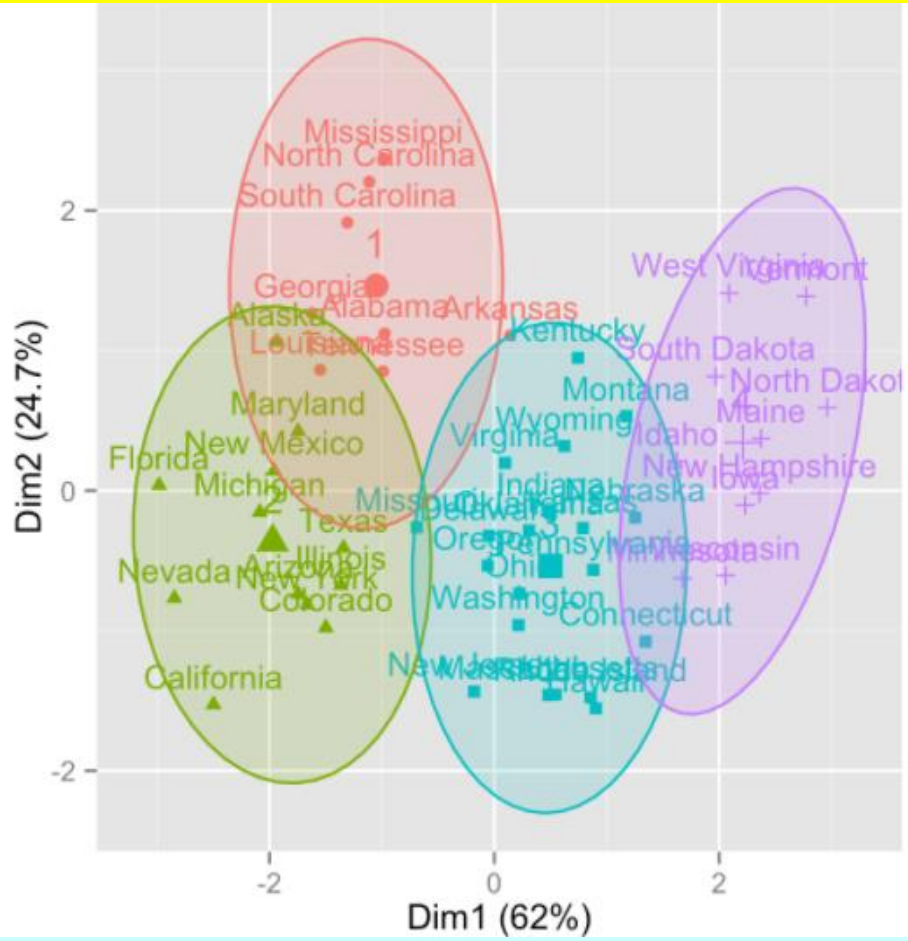
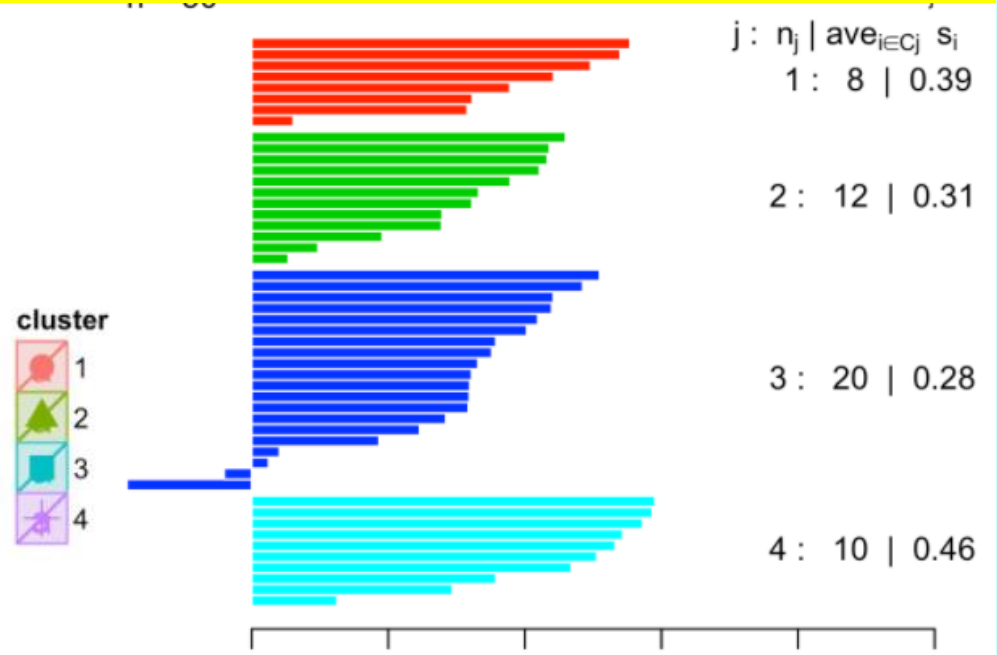
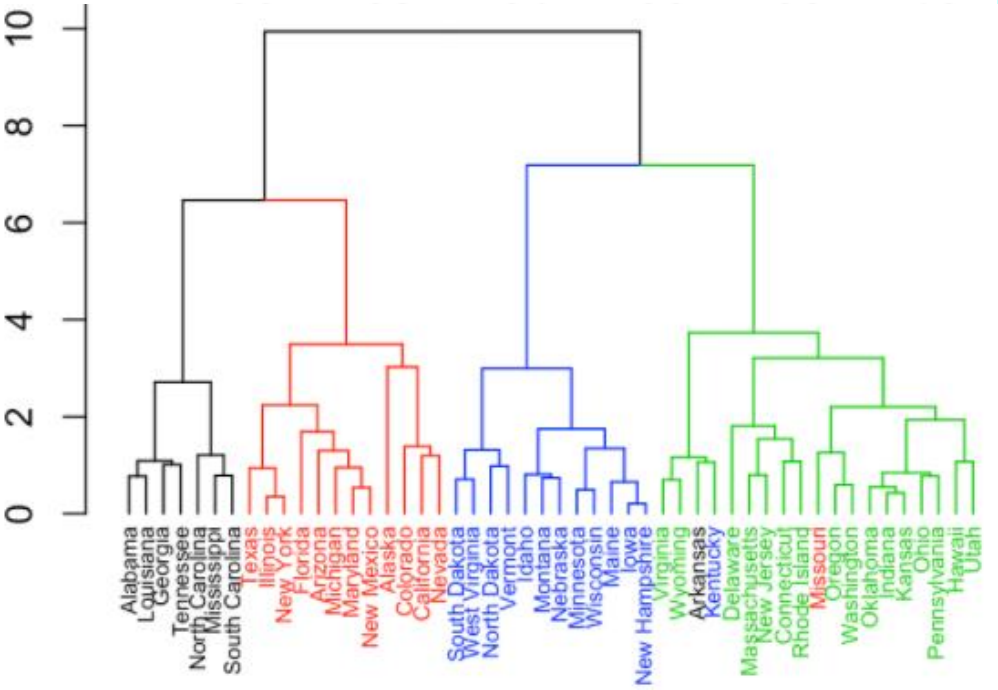
Clustering



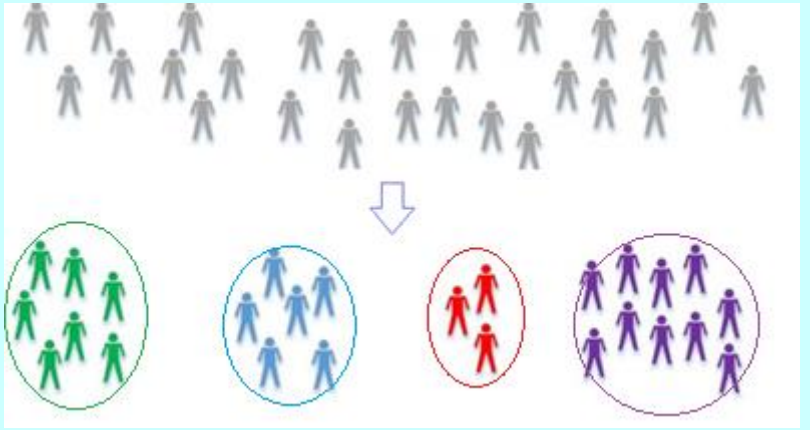




1. Optimal Number of Clusters
2. Partitioning Clustering
3. Hierarchical Clustering

Introduction to Clustering

- Aim of cluster analysis is make homogeneous subgroups called clusters, where the objects in the same cluster are similar to each other than to those in other clusters.



- It is very useful for **data mining** and **big data** because it **automatically finds patterns** in the data.

- Two methods of clustering:
  (1) **Partitioning clustering** (2) Hierarchical clustering

Ref: https://www.udemy.com/cluster-analysis-unsupervised-machine-learning-python/

1. Optimal Number of Clusters

- Determining the **optimal number of clusters** in a data set is a fundamental issue in partitioning clustering.
- There is no definitive answer to this question.
- The optimal number of clusters depends on the method used for measuring similarities and the parameters used for partitioning.

http://www.sthda.com/english/articles/29-cluster-validation-essentials/96-determining-the-optimal-number-of-clusters-3-must-know-methods/

Sample Dataset: Wine

```
> library(gclus)
> library(dplyr)
> data(wine)
> scaled_wine <- scale(wine) %>% as.data.frame()
> scaled_wine2 <- scaled_wine[-1]
> head(scaled_wine2,2)
    Alcohol      Malic        Ash Alcalinity  Magnesium   Phenols Flavanoids
1 1.5143408 -0.5606682  0.2313998  -1.166303 1.90852151 0.8067217  1.0319081
2 0.2455968 -0.4980086 -0.8256672  -2.483841 0.01809398 0.5670481  0.7315653
  Nonflavanoid Proanthocyanins  Intensity       Hue    OD280   Proline
1   -0.6577078       1.2214385  0.2510088 0.3610679 1.842721 1.0101594
2   -0.8184106      -0.5431887 -0.2924962 0.4048188 1.110317 0.9625263
```

(1) Determining the best number of clusters using NbClust

NbClust(data=NULL, distance="euclidean", method=NULL, ...) {NbClust}
NbClust provides 30 indices for determining the number of clusters and proposes to user the best clustering scheme from the different results obtained by varying all combinations of number of clusters, distance measures, and clustering methods.

```
> #(1) Determine the best numbers of clusters using NbClust
> library(NbClust)
> ?NbClust
> #hartigan: max difference btwn heirarchy levels of indices
> NbClust(scaled_wine2,method="complete",index="hartigan")$Best.nc
Number_clusters     Value_Index
         3.0000         27.8957
> #maximum value of the index
> NbClust(scaled_wine2,method="complete",index="kl")$Best.nc
Number_clusters     Value_Index
         5.0000         14.2208
```

Different computations (like Hartigan and Krzanowski-Lai) could identify different number of optimal clusters.

(2) Gap statistics for a number of clusters

[Ref] J. P. Lander, R for Everyone: Advanced Analytics and Graphics, 2nd ed. (Addison-Wesley, Boston, 2017) pp.395-397.

clusGap(x, FUNcluster, K.max, ...) {cluster}
clusGap() calculates a goodness of clustering measure, the "gap" statistic.

```
> #(2) Gap statistics for estimating the number of clusters
> library(cluster)
> ?clusGap
> Gap <- clusGap(scaled_wine2, FUNcluster=pam, K.max=15)
Clustering k = 1,2,..., K.max (= 15): .. done
Bootstrapping, b = 1,2,..., B (= 100)  [one "." per sample]:
.................................................. 50
.................................................. 100
```

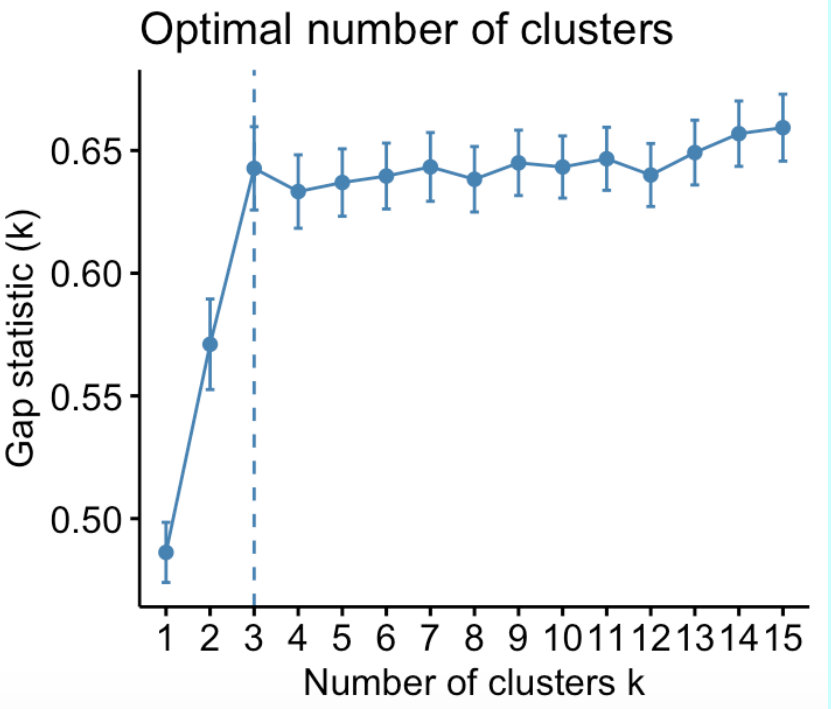
Gap statistic is a goodness of clustering measure, where for each hypothetical number of clusters k, it compares two functions: log of within-cluster sum of squares (wss) with its expectation under the null reference distribution of the data. In essence, it standardizes wss. It chooses the value where the log(wss) is the farthest below the reference curve, ergo the gap statistic is maximum.

https://tinyurl.com/y56fhnc3

(2) Gap statistics for a number of clusters

```
> print(Gap, method = "firstmax")
Clustering Gap statistic ["clusGap"] from call:
clusGap(x = scaled_wine2, FUNcluster = pam, K.max = 15)
B=100 simulated reference sets, k = 1..15; spaceH0="scaledPCA"
 --> Number of clusters (method 'firstmax'): 3
          logW   E.logW       gap     SE.sim
 [1,] 5.377557 5.863773 0.4862166 0.01224974
 [2,] 5.209591 5.780616 0.5710254 0.01845457
 [3,] 5.079884 5.722616 0.6427312 0.01698189
 [4,] 5.045616 5.678893 0.6332771 0.01495859
 [5,] 5.008127 5.645081 0.6369539 0.01373635
 [6,] 4.975594 5.615190 0.6395960 0.01342690
 [7,] 4.945702 5.589019 0.6433170 0.01400396
 [8,] 4.925718 5.563997 0.6382794 0.01334567
 [9,] 4.896147 5.541122 0.6449751 0.01333058
[10,] 4.876730 5.520008 0.6432781 0.01267107
[11,] 4.853909 5.500542 0.6466327 0.01284788
[12,] 4.841563 5.481574 0.6400115 0.01284148
[13,] 4.813724 5.462889 0.6491651 0.01319916
[14,] 4.789402 5.446266 0.6568637 0.01332145
[15,] 4.770006 5.429324 0.6593176 0.01363829
```

```
> library(factoextra)
> fviz_gap_stat(Gap)
```



According to the Gap statistics, three clusters is optimal for the Wine dataset.

2. Partitioning Clustering

Partitioning clustering are clustering methods used to classify observations, within a data set, into multiple groups based on their similarity.

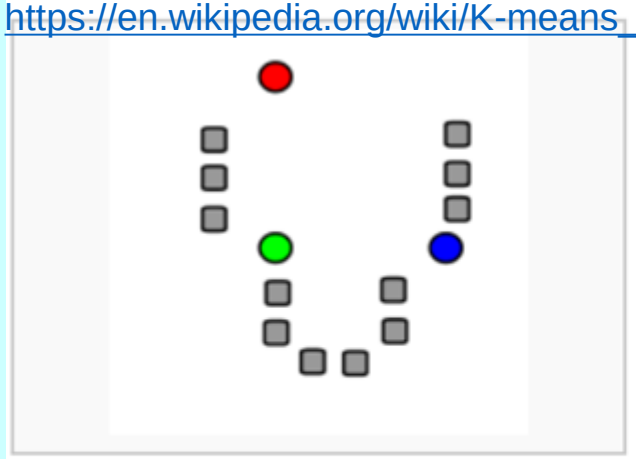
Ref: http://www.sthda.com/english/articles/27-partitioning-clustering-essentials/

- **K-means clustering**: Each cluster is represented by the center or means of the data points belonging to the cluster.
It is sensitive to anomalous data points and outliers.
- **K-medoids clustering** or **PAM** (*Partitioning Around Medoids*): Clustering of the data into k clusters "around medoids", a more robust version of K-means.
PAM is less sensitive to outliers compared to k-means.
- **CLARA algorithm** (*Clustering Large Applications*):
It is an extension to PAM adapted for large data sets.

(1) K-Means Clustering

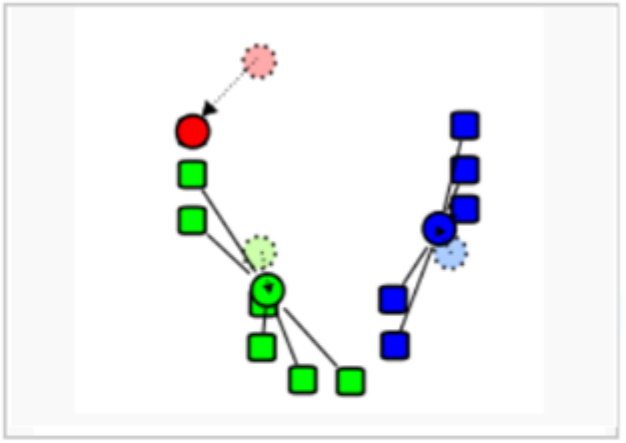
Demonstration of the standard algorithm

https://en.wikipedia.org/wiki/K-means_clustering

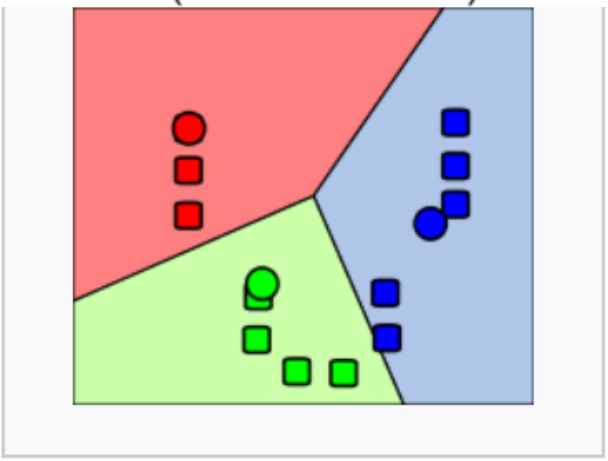


1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean.



3. The centroid of each of the *k* clusters becomes the new mean.



4. Steps 2 and 3 are repeated until convergence has been reached.

Each observation is assigned to the cluster with the smallest value of:

$$SS(k) = \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - \overline{x}_{kj})^2$$

k is the cluster, x_{ij} is the value of the j^{th} variable for the i^{th} observation, and $\overline{x}_{kj}$ is the mean of the j^{th} variable for the k^{th} cluster.

kmeans(x, centers, ...) {stats}
Perform k-means clustering on a data matrix.

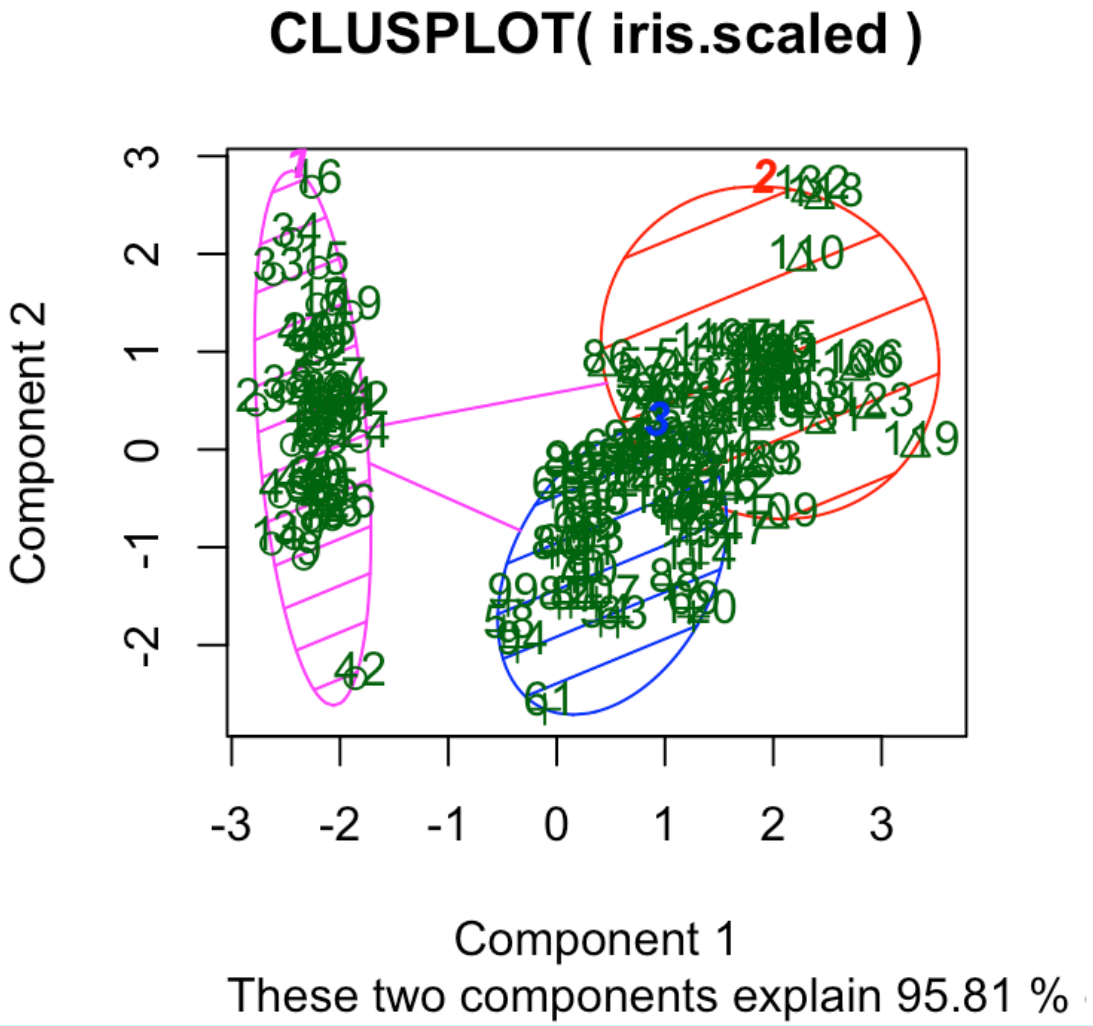
```
data(iris)
head(iris)
iris.scaled <- scale(iris[, -5])

library(NbClust)
nb <- NbClust(iris.scaled, distance = "euclidean", min.nc = 2,
      max.nc = 10, method = "complete", index ="all")
n  <- nb$Best.nc[1]
kc <- kmeans(iris.scaled, centers=n, nstart=4)
kc
```

```
> kc
K-means clustering with 3 clusters of sizes 50, 47, 53

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2   1.13217737  0.08812645    0.9928284   1.0141287
3  -0.05005221 -0.88042696    0.3465767   0.2805873

Clustering vector:
  [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
 [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 3 3 3 3 3 3 3 3 2 3 3 3 3
 [71] 2 3 3 3 3 2 2 2 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2
[106] 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 2 2 2 2 2 2 3 3 2 2 2 3 2
[141] 2 2 3 2 2 2 3 2 2 3

Within cluster sum of squares by cluster:
[1] 47.35062 47.45019 44.08754
 (between_SS / total_SS =  76.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

clusplot(x, ...) {cluster}

Bivariate Cluster Plot (of a Partitioning Object)

```
library(cluster)
clusplot(iris.scaled,kc$cluster,color=TRUE,shade=TRUE,labels=2)
```



Bivariate cluster plot explains 95.81% of variance.

▪k-means(cluster) → silhouette plot

silhouette(x, dist, dmatrix, ...) {cluster}
Compute silhouette information according to a given clustering in k clusters.

- Silhouette analysis can be used to study the separation distance between the resulting clusters.
- The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters.
- This measure has a range of [-1, 1].
- Criterion: average silhouette width > 0.4

Average Silhouette Width

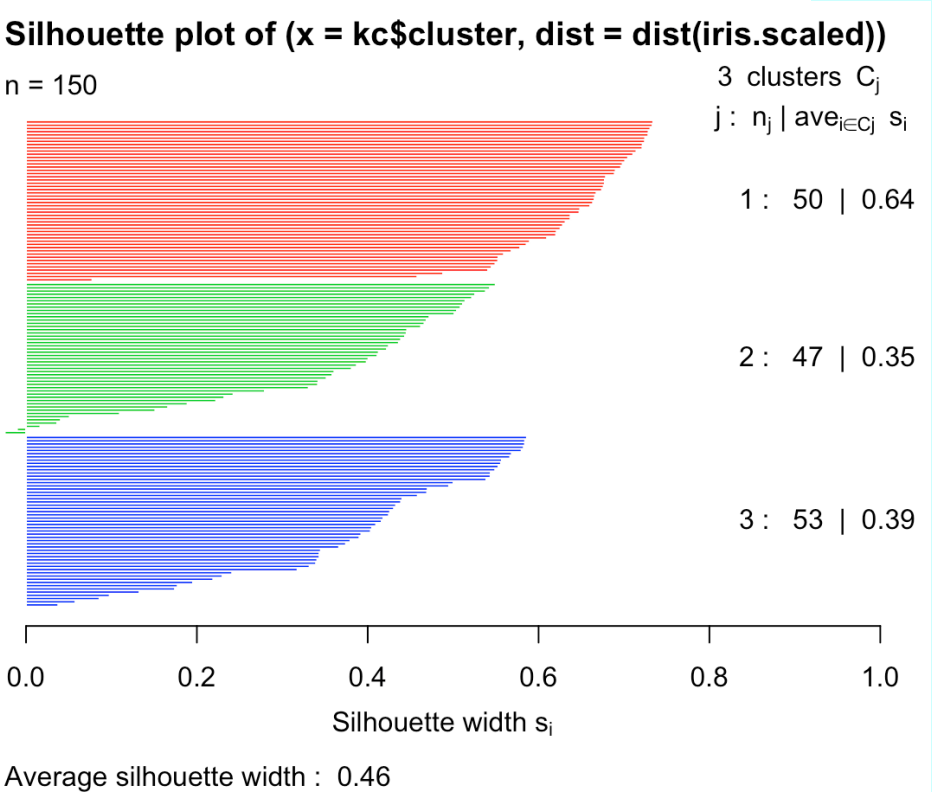
Range	Interpretation
0.71-1.0	A strong structure has been found
0.51-0.70	A reasonable structure has been found
0.26-0.50	The structure is weak and could be artificial
< 0.25	No substantial structure has been found

Ref: https://www.stat.berkeley.edu/~spector/s133/Clus.html

▪k-means(cluster) → silhouette plot

```
> sobj <- silhouette(kc$cluster, dist(iris.scaled))
> summary(sobj)
Silhouette of 150 units in 3 clusters from silhouette.default(x = kc$cluster,
dist = dist(iris.scaled)) :
 Cluster sizes and average silhouette widths:
       50        47        53
0.6363162 0.3473922 0.3933772
Individual silhouette widths:
    Min.  1st Qu.    Median      Mean  3rd Qu.     Max.
-0.02489  0.35914  0.47113  0.45995  0.58883  0.73419
> plot(sobj, col=2:4)
```

In the silhouette plot, each silhouette represents a cluster and their width represents the strength of each observation's membership in a cluster.



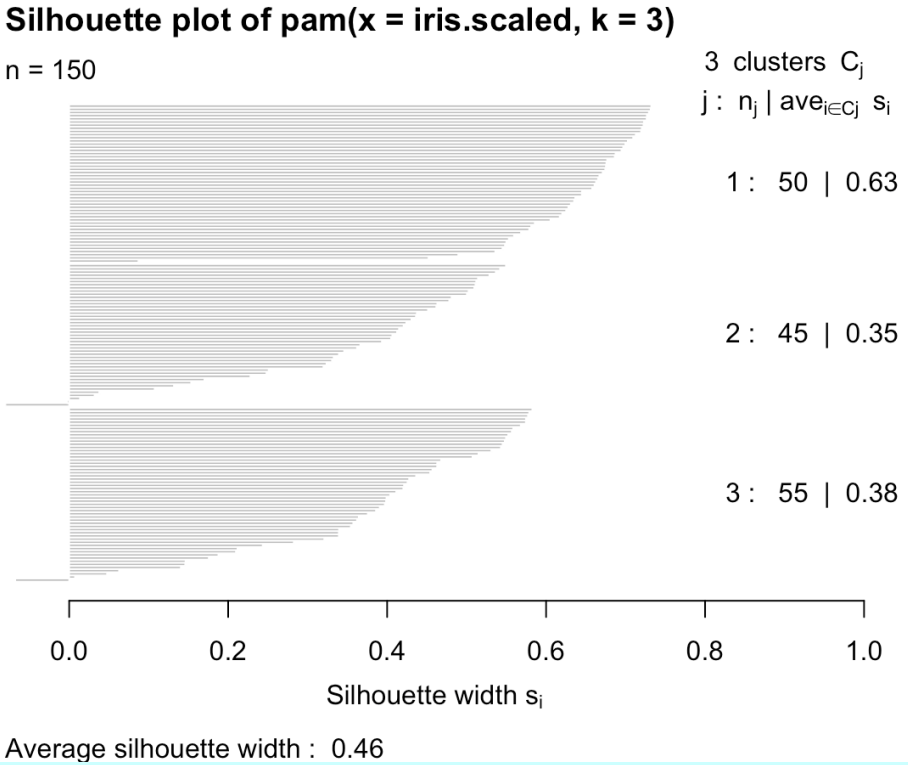
(2) Partitioning Around Medoids (PAM)

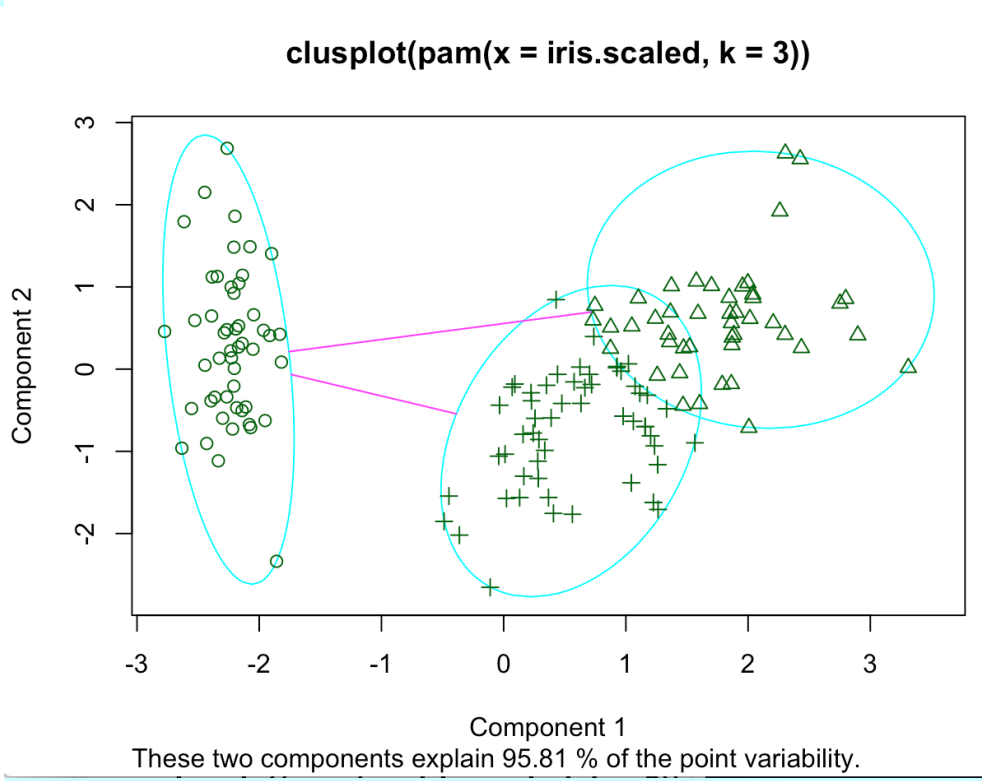
Two problems with K-means clustering:
(1) It does not work with categorical data.
(2) It is susceptible to outliers.
The most common K-medoids algorithm is PAM.

pam(x, k, ...) {cluster}
Partitioning (clustering) of the data into k clusters “around medoids”, a more robust version of K-means.

```
> library(cluster)
> pamx <- pam(iris.scaled, 3)
> summary(pamx)
Medoids:
      ID Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,]   8   -1.0184372   0.7861738   -1.2791040  -1.3110521
[2,] 113    1.1553023  -0.1315388    0.9868021   1.1816087
[3,]  56   -0.1730941  -0.5903951    0.4203256   0.1320673
Clustering vector:
  [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
 [34] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 3 3 3 3 3 3 3 3 2
 [67] 3 3 3 3 3 3 3 3 3 2 2 2 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3
[100] 3 2 3 2 2 2 2 3 2 2 2 2 2 2 3 2 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2
[133] 2 3 3 2 2 2 3 2 2 2 3 2 2 2 3 2 2 3
Objective function:
    build      swap
0.9205107 0.8757051
```

```
plot(pamx)
```

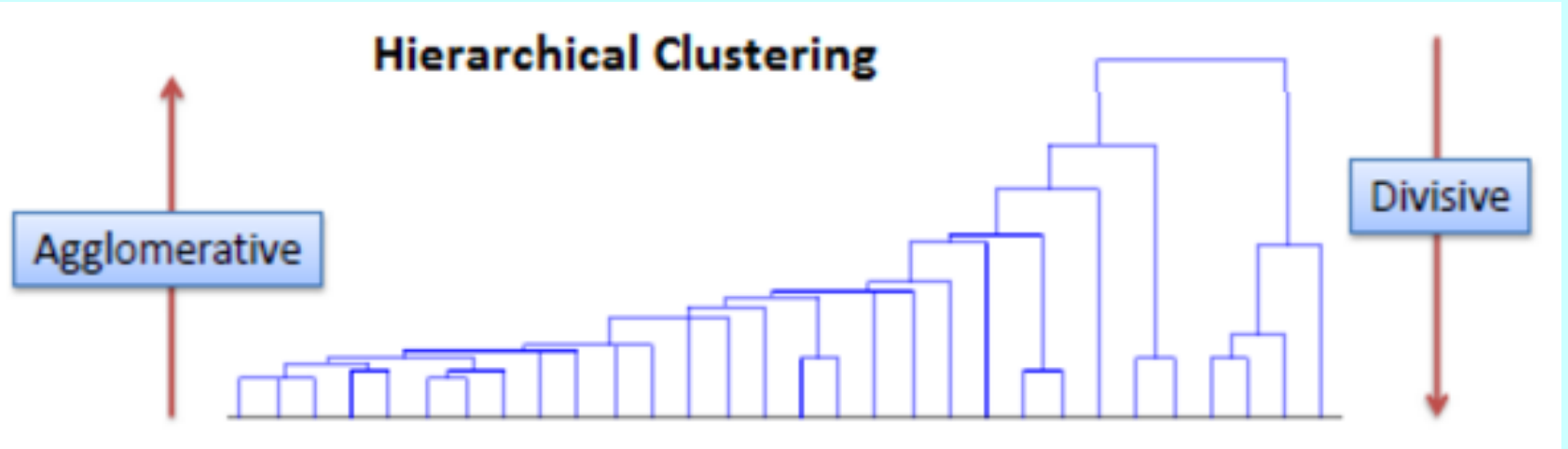

Silhouette plot of pam(x = iris.scaled, k = 3)
n = 150
3 clusters C_j
j : n_j | $ave_{i∈Cj}$ s_i
1 : 50 | 0.63
2 : 45 | 0.35
3 : 55 | 0.38
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.46


clusplot(pam(x = iris.scaled, k = 3))
Component 2
3
2
1
0
-1
-2
-3
-2
-1
0
1
2
3
Component 1
These two components explain 95.81 % of the point variability.

3. Hierarchical Clustering

In hierarchical clustering, the data is not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place, which may run from a single cluster containing all objects to *n* clusters that each contain a single object.

Ref: http://www.solver.com/xlminer/help/hierarchical-clustering-intro



- Agglomerative: Build up cluster from individual observations
- Divisive: Start with whole group of observations and split off clusters
- Divisive clustering has much larger computational burden Agglomerative clustering is commonly used.

Example: USArrests

USArrests {datasets}

Violent Crime Rates by US State

This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973.

```
> head(USArrests)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

·Number of Clusters

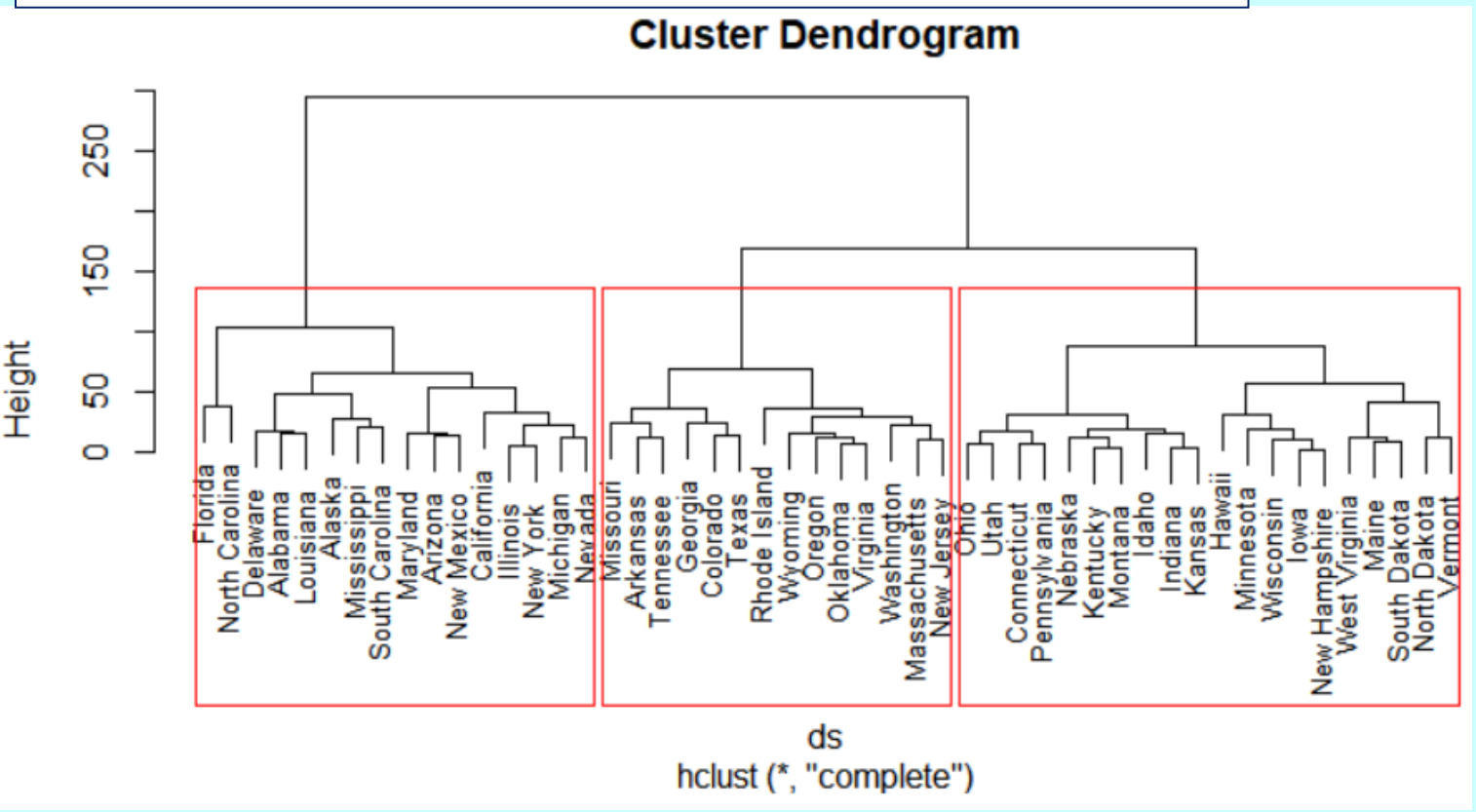
```
> library(NbClust)
> NbClust(USArrests, method="complete", index="hartigan")$Best.nc
Number_clusters     Value_Index
         3.0000         50.9205
```

•Visualizing Clusters

hclust(d, method="complete") {stats}
Hierarchical cluster analysis on a set of dissimilarities and methods for analyzing it.

dist(x, method="euclidean") {stats}
This function computes and returns the distance matrix

```
ds <- dist(USArrests, method="euclidean")
hcst <- hclust(ds,method="complete")
plot(hcst, labels=rownames(USArrests), cex=0.8)
rect.hclust(hcst, 3)
```



The cluster dendrogram looks nice and is extremely useful to have similar elements grouped together.

·Cluster Membership in a Vector

cutree(tree, k)

Cut a tree into groups of data.

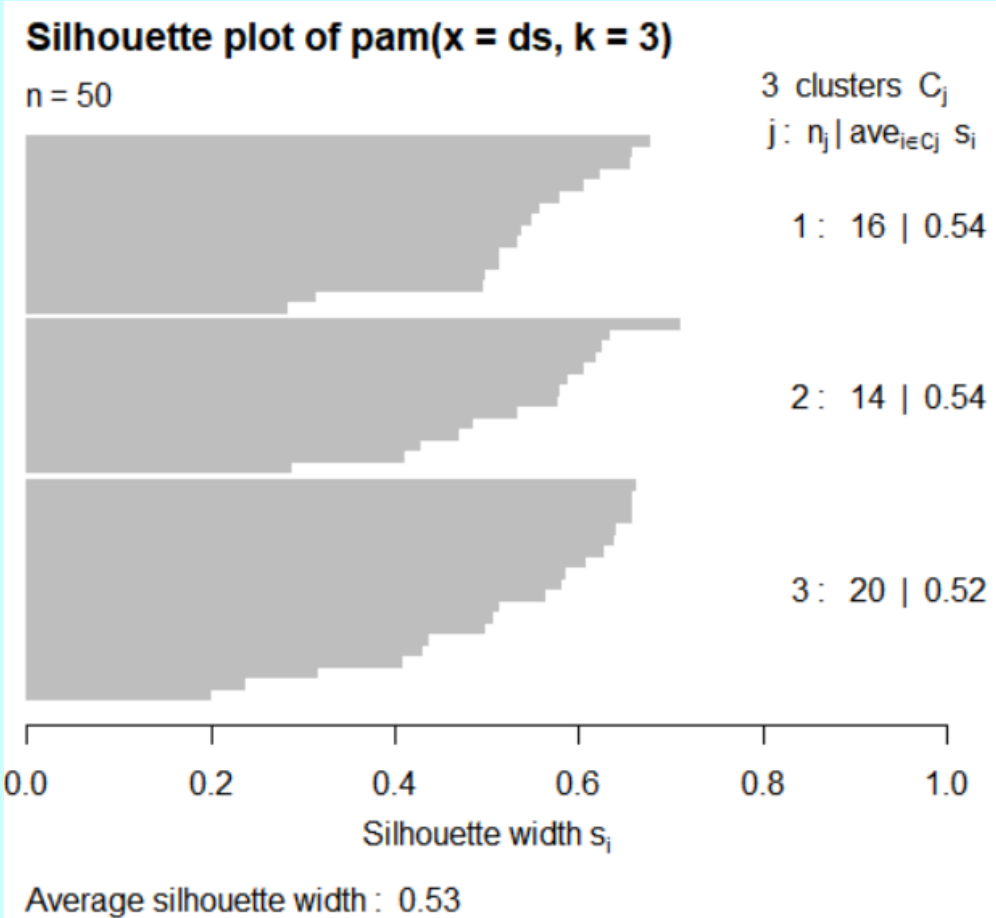
```
> cn <- cutree(hcst, k=3)
> cn
       Alabama         Alaska        Arizona       Arkansas     California       Colorado
             1              1              1              2              1              2
   Connecticut       Delaware        Florida        Georgia         Hawaii          Idaho
             3              1              1              2              3              3
      Illinois        Indiana           Iowa         Kansas       Kentucky      Louisiana
             1              3              3              3              3              1
         Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi
             3              1              2              1              3              1
      Missouri        Montana       Nebraska         Nevada  New Hampshire     New Jersey
             2              3              3              1              3              2
    New Mexico       New York North Carolina   North Dakota           Ohio       Oklahoma
             1              1              1              3              3              2
        Oregon   Pennsylvania   Rhode Island South Carolina   South Dakota      Tennessee
             2              3              2              1              3              2
         Texas           Utah        Vermont       Virginia     Washington  West Virginia
             2              3              3              2              2              3
     Wisconsin        Wyoming
             3              2
> table(cn)
cn
 1  2  3
16 14 20
> aggregate(USArrests,FUN=mean, by=list(cn))
  Group.1    Murder  Assault UrbanPop     Rape
1       1 11.812500 272.5625 68.31250 28.37500
2       2  8.214286 173.2857 70.64286 22.84286
3       3  4.270000  87.5500 59.75000 14.39000
```

•Silhouette Plot with PAM

PAM provides a nice example of an alternative technique to hierarchical clustering.

```
ds <- dist(USArrests, method="euclidean")
library(cluster)
pamd <- pam(ds, 3)
plot(pamd)
```

The silhouette plot is very useful in locating groups in a cluster analysis and can be used to help select the proper number of clusters.



```
sobj <- silhouette(pamd)
plot(sobj,col=2:4)
```