Clustering

Nik Bear Brown

In this lesson, we cover clustering theory, and distance & similarity measures. We use R apply and evaluate various clustering techniques such as hierarchical clustering, k-means clustering and pam. We look at techniques to help choose the number of clusters as well as how to evaluate cluster performance.

# Additional packages needed

To run the code in M02\_Lesson\_02.Rmd you may need additional packages.

* If necessary install ggplot2, cluster and amap packages.

install.packages("ggplot2");  
install.packages("cluster");  
install.packages("amap");  
install.packages("useful");

require(ggplot2)

## Loading required package: ggplot2

require(cluster)

## Loading required package: cluster

require(amap)

## Loading required package: amap

require(useful)

## Loading required package: useful

# Data

We'll be using GDP per capita, life expectancy, infant.mortality, and literacy data made availble by the WorldBank [data.worldbank.org](http://data.worldbank.org/)

GDP per capita (current US$)

GDP per capita is gross domestic product divided by midyear population. GDP is the sum of gross value added by all resident producers in the economy plus any product taxes and minus any subsidies not included in the value of the products. It is calculated without making deductions for depreciation of fabricated assets or for depletion and degradation of natural resources. Data are in current U.S. dollars.

Life expectancy at birth, total (years)

Life expectancy at birth indicates the number of years a newborn infant would live if prevailing patterns of mortality at the time of its birth were to stay the same throughout its life. Derived from male and female life expectancy at birth from sources such as: (1) United Nations Population Division. World Population Prospects, (2) United Nations Statistical Division. Population and Vital Statistics Report (various years), (3) Census reports and other statistical publications from national statistical offices, (4) Eurostat: Demographic Statistics, (5) Secretariat of the Pacific Community: Statistics and Demography Programme, and (6) U.S. Census Bureau: International Database.

Mortality rate, infant (per 1,000 live births)

Infant mortality rate is the number of infants dying before reaching one year of age, per 1,000 live births in a given year. Estimates developed by the UN Inter-agency Group for Child Mortality Estimation (UNICEF, WHO, World Bank, UN DESA Population Division) at [www.childmortality.org](http://www.childmortality.org/).

Literacy rate, adult total (% of people ages 15 and above)

Adult (15+) literacy rate (%). Total is the percentage of the population age 15 and above who can, with understanding, read and write a short, simple statement on their everyday life. Generally, ‘literacy’ also encompasses ‘numeracy’, the ability to make simple arithmetic calculations. This indicator is calculated by dividing the number of literates aged 15 years and over by the corresponding age group population and multiplying the result by 100.

We will also be using [Francis Galton's](https://en.wikipedia.org/wiki/Francis_Galton) analysis of the heights of sons and fathers. Heights of sons of both tall and short fathers appeared to “revert” or “regress” to the mean of the group.

We will also be using the [Wholesale customers Data Set](https://archive.ics.uci.edu/ml/datasets/Wholesale+customers) (Data used in Hierarchical Clustering). The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories.

Link: [<https://archive.ics.uci.edu/ml/machine-learning-databases/00292/>](https://archive.ics.uci.edu/ml/machine-learning-databases/00292/) Source: UCI Machine Learning Repository : Data Set: Wholesale customer data.csv

The data has been modified so that each row (expenses) has been simply identified as a different Region (Region 1 to 49).

# Load our data  
  
data\_url <- 'http://nikbearbrown.com/YouTube/MachineLearning/M02/Galton\_heights\_sons\_and\_fathers.csv'  
galton <- read.csv(url(data\_url))  
data\_url <- 'http://nikbearbrown.com/YouTube/MachineLearning/M04/AnnualSpending.csv'  
spend <- read.csv(url(data\_url))  
data\_url <- 'http://nikbearbrown.com/YouTube/MachineLearning/M04/data.worldbank.org.csv'  
dwb <- read.csv(url(data\_url),header=TRUE)

## Galton's data set

Francis Galton's analysis of the heights of sons and fathers .

head(galton)

## Family Father Mother Gender Height Kids  
## 1 1 78.5 67.0 M 73.2 4  
## 2 1 78.5 67.0 F 69.2 4  
## 3 1 78.5 67.0 F 69.0 4  
## 4 1 78.5 67.0 F 69.0 4  
## 5 2 75.5 66.5 M 73.5 4  
## 6 2 75.5 66.5 M 72.5 4

h.gend<-galton[c("Height","Gender")]  
head(h.gend)

## Height Gender  
## 1 73.2 M  
## 2 69.2 F  
## 3 69.0 F  
## 4 69.0 F  
## 5 73.5 M  
## 6 72.5 M

You can also

## data.worldbank.org

data.worldbank.org

head(dwb)

## Country Country.code Region  
## 1 Afghanistan AFG South Asia  
## 2 Albania ALB Europe & Central Asia  
## 3 Algeria DZA Middle East & North Africa  
## 4 American Samoa ASM East Asia & Pacific  
## 5 Andorra AND Europe & Central Asia  
## 6 Angola AGO Sub-Saharan Africa  
## IncomeGroup Life.expectancy Infant.mortality Per.capita.income  
## 1 Low income 60.9 70.2 575  
## 2 Upper middle income 77.5 13.3 4336  
## 3 Upper middle income 71.0 21.6 4887  
## 4 Upper middle income 44.9 2.7 37082  
## 5 High income 48.8 2.2 44095  
## 6 Upper middle income 51.9 101.6 5032  
## Literacy  
## 1 31.7  
## 2 98.7  
## 3 72.6  
## 4 97.3  
## 5 91.7  
## 6 70.6

wb<-dwb[c("Life.expectancy","Infant.mortality","Per.capita.income","Literacy")]  
head(wb)

## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 60.9 70.2 575 31.7  
## 2 77.5 13.3 4336 98.7  
## 3 71.0 21.6 4887 72.6  
## 4 44.9 2.7 37082 97.3  
## 5 48.8 2.2 44095 91.7  
## 6 51.9 101.6 5032 70.6

## Wholesale customers data set

Wholesale customers Data Set

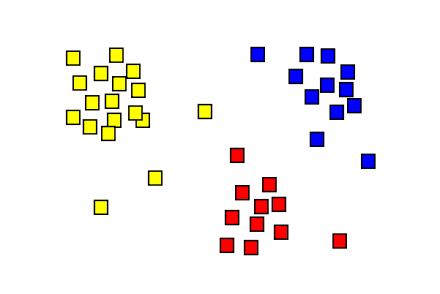
head(spend)

## Region Fresh Milk Grocery Frozen Detergents.Paper Delicassen  
## 1 1 12669 9656 7561 214 2674 1338  
## 2 2 7057 9810 9568 1762 3293 1776  
## 3 3 6353 8808 7684 2405 3516 7844  
## 4 4 13265 1196 4221 6404 507 1788  
## 5 5 22615 5410 7198 3915 1777 5185  
## 6 6 9413 8259 5126 666 1795 1451

# Clustering

What is [Clustering](https://en.wikipedia.org/wiki/Cluster_analysis)? Clustering is grouping like with like such that:

* Similar objects are close to one another within the same cluster.
* Dissimilar to the objects in other clusters.



*cluster analysis*

# Distance & Similarity Measures

There are two primary approaches to measure the "closeness" of data, distance and similarity. Distance measures are based in the notion of a [metric space](https://en.wikipedia.org/wiki/Metric_space).

[Similarity measures](https://en.wikipedia.org/wiki/Similarity_measure) s a real-valued function that quantifies the similarity between two objects. While no single definition of a similarity measure exists, they don't necessarily have the constraint of being a metric space. For example, [cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity) is a measure of the "closeness" of two vectors that is not a metric space.

## Metric Spaces

A *metric space* is an *ordered pair* where is a set and is a distance (i.e.metric) on , i.e., a function:

such that for any , the following holds:



### Examples of Metric Spaces

Examples of Metric Spaces are [Chebyshev distance](https://en.wikipedia.org/wiki/Chebyshev_distance), [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance), [Hamming distance](https://en.wikipedia.org/wiki/Hamming_distance), [Minkowski distance](https://en.wikipedia.org/wiki/Minkowski_distance) and many others.

#### Euclidean distance

Euclidean distance is the most common metric for measuring the distance between two vectors. The is the stadard Cartesian coordinates. That is, if and are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the Pythagorean formula:

## Similarity Measures

Similarity (or dimilarity measures) measure closeness without the constraints and benefits of being a formal metric space.

### Cosine similarity

The most common form of the similarity measure is the vector inner product (or [cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity)) Given vectors A and B, the vector inner product can be defined using the [Euclidean dot product](https://en.wikipedia.org/wiki/Euclidean_vector#Dot_product) formula:

This similarity measure can also be ranged normalized. Alternately, we can normalize this measure by dividing each vector component by the magnitude of the vector.

### Pearson correlation

Correlation based similarity is usually the [Pearson correlation](https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient). Pearson product-moment correlation coefficient commonly represented by the Greek letter (rho) and is defined as:

where is the [covariance](https://en.wikipedia.org/wiki/Covariance) and is the [standard deviation](https://en.wikipedia.org/wiki/Standard_deviation) of .

The formula for can be expressed in terms of mean and expectation.

# Clustering Data Structures

* Data matrix

Distance matrix

* Dissimilarity/Similarity matrix

The dissimilarity/similarity matrix is calculated by iterating over each element and calculating its dissimilarity/similarity to every other element. Let A be a Dissimilarity Matrix of size , and a set of elements. is the dissimilarity/similarity between elements and .

for i = 0 to N do  
 for j = 0 to N do  
 Aij = Dissimilarity(Bi,Bj) // or Similarity(Bi,Bj)  
 end-for  
 end-for

where the dissimilarity/similarity matrix is usually defined as follows:

# Types of Clustering

[Clustering](https://en.wikipedia.org/wiki/Cluster_analysis) (e.g., k-means, mixture models, hierarchical clustering). Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). There are various types of cluster analysis.

* Partitioning-based clustering (K-means and its variants)
* Hierarchical clustering
* Density-based clustering

## Partitioning-based clustering (K-means and its variants)

Partitioning algorithms: Construct various partitions and then evaluate them by some criterion Hierarchy algorithms: Create a hierarchical decomposition of the set of data (or objects) using some criterion Density-based: based on connectivity and density functions Grid-based: based on a multiple-level granularity structure Model-based: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

Partitioning method: Construct a partition of n documents into a set of K clusters Given: a set of documents and the number K Find: a partition of K clusters that optimizes the chosen partitioning criterion Globally optimal Intractable for many objective functions Ergo, exhaustively enumerate all partitions Effective heuristic methods: K-means and K-medoids algorithms

### K-means

The term "k-means" was first used by James MacQueen in 1967,[1] though the idea goes back to Hugo Steinhaus in 1957. Given a desired clusters and n data points :

Initialize centroids (usually randomly)

while (not coverged):

Step A (Assignment step):

Find the closest cluster to every point in

That is,

$$\underset{\mathbf{X}} {\operatorname{arg\,min}} \sum\_{i=1}^{k} \sum\_{\mathbf x \in X\_i} \left\| \mathbf x - \boldsymbol\mu\_i \right\|^2$$

where is the mean of points in .

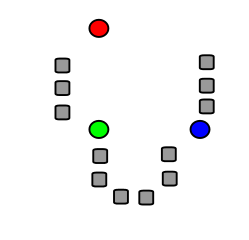
Step B (Update step):

Calculate the new means to be the centroids of the observations in the new clusters.

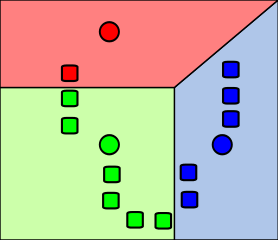
The [centroid](https://en.wikipedia.org/wiki/Centroid) of a finite set of {k} points

#### Demonstration of the k-means algorithm

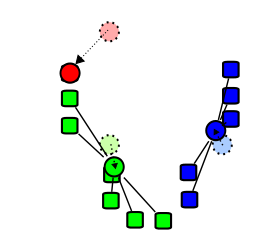
Demonstration of the standard algorithm (from [k-means algorithm](https://en.wikipedia.org/wiki/K-means_clustering#Standard_algorithm)



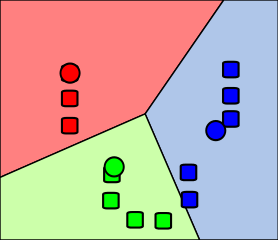
1. k initial "means" (in this case k=3) are randomly generated within the data domain (shown in color).



1. k clusters are created by associating every observation with the nearest mean. The partitions here represent the [Voronoi diagram](https://en.wikipedia.org/wiki/Voronoi_diagram) generated by the means.



1. The centroid of each of the k clusters becomes the new mean.



1. Steps 2 and 3 are repeated until convergence has been reached. The k-means algorithm converges when no points are assigned new clusters.

#### Problems with k-means clustering

For problems with k-means clustering see [K-means clustering is not a free lunch](http://varianceexplained.org/r/kmeans-free-lunch/)

### K-medoids (PAM)

The K-medoids or Partitioning Around Medoids (PAM) algorithm (Kaufman & Rousseeuw’87) is related to the k-means algorithm and but uses medoid shifts rather than reassigning points based on Euclidean distance. Each cluster is represented by one of the objects (i.e. points) in the cluster A medoid is a point in a cluster whose dissimilarity to all the points in the cluster is minimal. Medoids are similar in concept to means or centroids, but medoids are always members of the data set. That is, in 2D [Cartesian space](https://en.wikipedia.org/wiki/Cartesian_coordinate_system) a centroid can be any valid x.y coordinate. Whereas a medoid must be one of the data points. (The data point least dissimilar to the rest.)

Pseduocode for the k-medoid clustering (Partitioning Around Medoids (PAM)) algorithm:

Initialize: randomly select[citation needed] (without replacement) k of the n data points as the medoids  
  
Associate each data point to the closest medoid.  
  
While the cost of the configuration decreases:  
 For each medoid m, for each non-medoid data point o:  
 Swap m and o, recompute the cost  
  
 If the total cost of the configuration increased in the previous step, undo the swap.

## Hierarchical clustering

In [hierarchical clustering](https://en.wikipedia.org/wiki/Hierarchical_clustering) the idea is to group data objects (i.e. points) into a tree of clusters. That is, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters.

These trees (hierarchies) generally fall into two types:

### Agglomerative hierarchical clustering

Initially each data object (i.e. point) in its own cluster. Iteratively the clusters are merged together from the "bottom-up." The two most similar/closest objects are aggreated in to the same cluster/data object. Then the next two, until there is just one cluster/data object. This agglomerative approach result in “straggly” (long and thin) clusters due to a chaining effect. It is also sensitive to noise.

### Divisive hierarchical clustering

in divisive hierarchical clustering all data objects (i.e. points) are initially in one cluster. These clusters are successively divided recursivley in a "top-down" manner. The cluster is broken in to two clusters that are most dissimilar. Then each of those clusters is broken in to two cluster that are most dissimilar. This continues until each clsuter is a single data object (i.e. point).

### Linkage criteria

* Single Link: smallest distance between points
* Complete Link: largest distance between points
* Average Link: average distance between points
* Centroid: distance between centroids
  + where and are the centroids of clusters and , respectively.
* Minimum energy clustering: a statistical distance between probability distributions.
  + $\_{i,j=1}^{n,m} |a\_i- b\_j|*2 -* {i,j=1}^{n} |a\_i-a\_j|*2 -* {i,j=1}^{m} |b\_i-b\_j|\_2 $

## Density-based clustering

Density-based clustering is similar to k-means clustering, except that it uses the [expectation-maximization algorithm](https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm), to generate a likelihood that for each data objects (i.e. points) belong to a cluster. In the special case of a Gaussian mixture model, specifically, the limit of taking all covariances as diagonal, equal, and small. A k-means problem can be generalized into a Gaussian mixture model.

Density-based clustering uses a density estimator to learn a probalistic mapping from a set of attributes to a probability.

Input -> Density Estimator -> Probability

Given a data object x(i.e. point), a density estimator M can tell you how likely x belongs to cluster k.

Given a statistical model which generates a set of observed data (e.g. assuming it comes for a Gaussian distribution), a set of unobserved latent data or missing values , and a vector of unknown parameters $\boldsymbol\theta$, along with a likelihood function $L(\boldsymbol\theta; \mathbf{X}, \mathbf{Z}) = p(\mathbf{X}, \mathbf{Z}|\boldsymbol\theta),$ the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data $L(\boldsymbol\theta; \mathbf{X}) = p(\mathbf{X}|\boldsymbol\theta) = \sum\_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\boldsymbol\theta)$

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps:

Expectation step (E step): Calculate the expected value of the log likelihood function, with respect to the conditional distribution of given under the current estimate of the parameters

$$\boldsymbol\theta^{(t)}: Q(\boldsymbol\theta|\boldsymbol\theta^{(t)}) = \operatorname{E}\_{\mathbf{Z}|\mathbf{X},\boldsymbol\theta^{(t)}}\left[ \log L(\boldsymbol\theta;\mathbf{X},\mathbf{Z}) \right] $$

Maximization step (M step): Find the parameter that maximizes this quantity:

$$ \boldsymbol\theta^{(t+1)} = \underset{\boldsymbol\theta}{\operatorname{arg\,max}} Q(\boldsymbol\theta|\boldsymbol\theta^{(t)}) $$

# Good Clustering?

Evaluation of clustering results sometimes is referred to as cluster validation.

* high intra-class distance/similarity
* low inter-class distance/similarity

Cluster Cohesion: Measures how closely related are objects in a cluster.

Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters.

The quality of a clustering result depends on both the similarity measure used by the method and its implementation

## Visualization

There are a number of packages for plotting clusters:

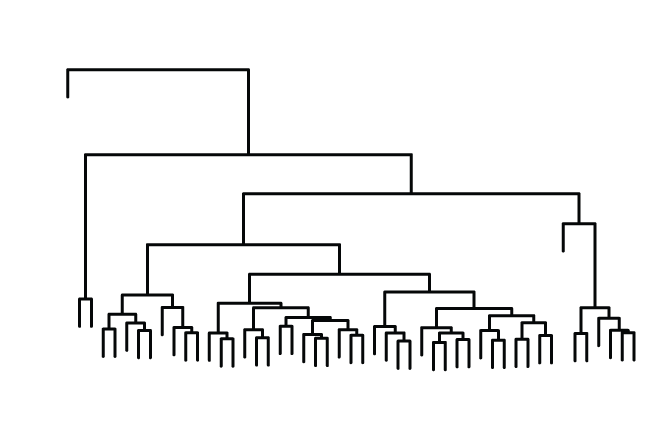
* R package [rggobi](https://cran.r-project.org/web/packages/rggobi/index.html) along with [GGobi.org](http://www.ggobi.org/)
* R package [clusterfly](http://had.co.nz/model-vis/)
* other R packages

We'll focus on dendrograms, multidimensional scaling (MDS) and plotting a confusion matrix.

### Dendrogram

A [dendrogram](https://en.wikipedia.org/wiki/Dendrogram) is a tree data structure which illustrates hierarchical (or taxonomic) relationships

* Each level shows clusters for that level.
* Leaf – individual clusters
* Root – one cluster
* A cluster at level is the union of its children clusters at level



*A dendrogram*

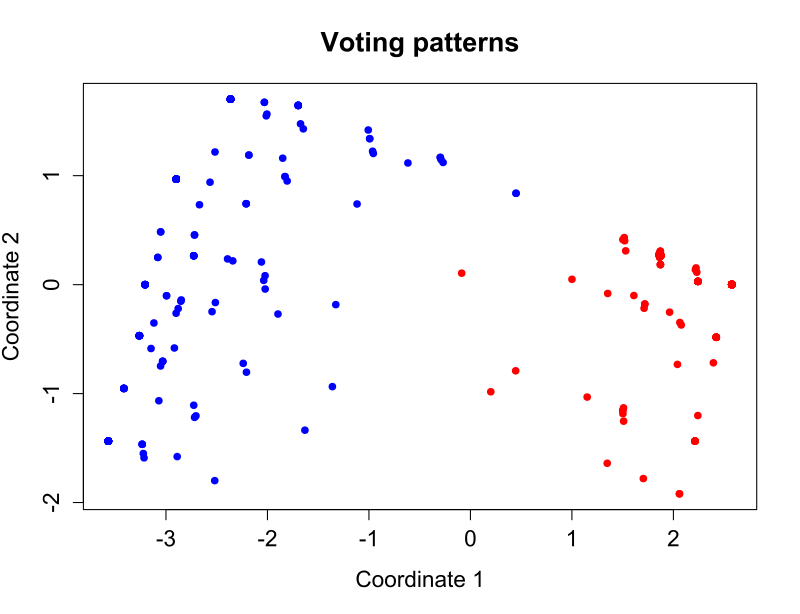
### Multidimensional scaling (MDS)

[Multidimensional scaling (MDS)](https://en.wikipedia.org/wiki/Multidimensional_scaling) is a means of visualizing the level of similarity of individual cases of a high-dimenional dataset.

Given a dataset of I objects (colors, faces, stocks, . . .) on which a distance function is defined, distance between and objects. These distances are the entries of the dissimilarity matrix

The goal of MDS is, given , to find I vectors such that for all , where is a vector norm. In classical MDS, this norm is the Euclidean distance, but, it may be a metric or arbitrary distance function.

In other words, MDS attempts to find an embedding from the I objects into such that distances are preserved.



*An example of classical multidimensional scaling applied to voting patterns in the United States House of Representatives*

### Confusion plots

confusion plot is a plot of the [confusion matrix](https://en.wikipedia.org/wiki/Confusion_matrix). A confusion matrix, also known as a contingency table or an error matrix.

A confusion matrix is a 2x2 table with counts of the following in each of its quadrents.

true positive (TP)  
eqv. with hit  
true negative (TN)  
eqv. with correct rejection  
false positive (FP)  
eqv. with false alarm, Type I error  
false negative (FN)  
eqv. with miss, Type II error

## Cluster evaluation metics

Cluster evaluation metics quantitate within cluster cohesion and between cluster Separation.

### Measuring Cluster Validity via Correlation

Compute can compute the correlation between the two matrices (a proximity patrix and “incidence” matrix). Since the matrices are symmetric, only the correlation between entries needs to be calculated. (ie. above or below the diagonal).

High correlation indicates that points that belong to the same cluster are close to each other. This is not a good measure for some density or contiguity based clusters.

*Proximity Matrix*

A proximity is a measurement of the similarity or dissimilarity, broadly defined, of a pair of objects. If measured for all pairs of objects in a set (e.g. driving distances among a set of U.S. cities), the proximities are represented by an object-by-object proximity matrix.

“Incidence” Matrix

An “incidence” matrix

* One row and one column for each data point
* An entry is 1 if the associated pair of points belong to the same cluster
* An entry is 0 if the associated pair of points belongs to different clusters

### Using Similarity Matrix for Cluster Validation

One can sort the similarity matrix with respect to cluster labels amd then plot.

### Davies–Bouldin index

The [Davies–Bouldin index](https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index) can be represented by the formula below:

where n is the number of clusters, is the centroid of cluster x, is the average distance of all elements in cluster x to centroid , and is the distance between centroids and .

Clusters with low intra-cluster distances (high intra-cluster similarity) and high inter-cluster distances (low inter-cluster similarity) will have a low Davies–Bouldin index.

### Dunn index

The [Dunn index](https://en.wikipedia.org/wiki/Dunn_index) aims to identify dense and well-separated clusters. It is defined as the ratio between the minimal inter-cluster distance to maximal intra-cluster distance. For each cluster partition, the Dunn index can be calculated by the following formula:

where represents the distance between clusters and , and measures the intra-cluster distance of cluster . The inter-cluster distance between two clusters may be any number of distance measures, such as the distance between the centroids of the clusters.

Clusters with high intra-cluster similarity and low inter-cluster similarity, produce clusters with high Dunn index (i.e. are more desirable).

### Silhouette coefficient

The silhouette coefficient contrasts the average distance to elements in the same cluster with the average distance to elements in other clusters. Objects with a high silhouette value are considered well clustered, objects with a low value may be outliers. This index works well with k-means clustering, and is also used to determine the optimal number of clusters.

### Purity (as an evaluation measure)

Simple measure: purity, the ratio between the dominant class in the cluster and the size of cluster. To compute purity , each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by . Formally:

Note that this is biased because having n clusters for n data objects maximizes purity.

### Entropy (as an evaluation measure)

The notion of entropy can be used to evauluate the "stickness" or mutual information between classes and clusters They can measure how much knowing one of these variables reduces uncertainty about the other. In a sense we want to measure how "sticky" within cluster data is and how "non-sticky" the between cluster data is. There are several ways to use entropy to measure "stickness."

#### Mutual Information

[Mutual information](https://en.wikipedia.org/wiki/Mutual_information) quantifies the mutual dependence of the two random variables. It is a measure of the “stickiness” between two items. It measures how much knowing one of these variables reduces uncertainty about the other. We can use mutual information to quantify the association between two tags. Mutual information is given by: $$ I(X;Y) = *{y Y}* {x X} p(x,y) , ,!

$$

where is the joint probability distribution function of and , and and are the marginal probability distribution functions of and respectively.

#### Kullback-Leibler divergence

[Kullback-Leibler divergence](https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence) is a non-symmetric measure of the difference between two probability distributions. The Kullback-Leibler measure goes by several names: relative entropy, discrimination information, Kullback-Leibler (KL) number, directed divergence, informational divergence, and cross entropy. Kullback-Leibler divergence is a measure of the difference between the observed entropy and its excepted entropy. We calculate the KL divergence by weighting one distribution (like an observed frequency distribution) by the log of probabilities of some other distribution. For discrete probability distributions P and Q, the Kullback–Leibler divergence of Q from P is defined to be:

In words, it is the expectation of the logarithmic difference between the probabilities P and Q, where the expectation is taken using the probabilities P.

#### Jaccard similarity coefficient

The [Jaccard index](https://en.wikipedia.org/wiki/Jaccard_index), also known as the Jaccard similarity coefficient (originally coined coefficient de communauté by Paul Jaccard), is a statistic used for comparing the similarity and diversity of sample sets. If ) are two vectors with all real , then their Jaccard similarity coefficient is defined as

and Jaccard distance

### Rand index

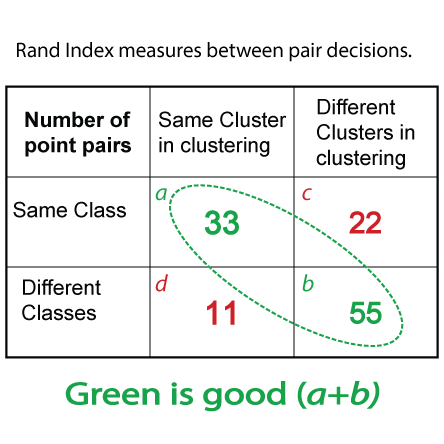
The [Rand index](https://en.wikipedia.org/wiki/Rand_index) or Rand measure (named after William M. Rand) in statistics, and in particular in data clustering, is a measure of the similarity between two data clusterings. A form of the Rand index may be defined that is adjusted for the chance grouping of elements, this is the adjusted Rand index.

Given a set of elements and two partitions of to compare, , a partition of into subsets, and , a partition of into subsets, define the following:

* , the number of pairs of elements in that are in the same set in and in the same set in (i.e. true positive (TP))
* , the number of pairs of elements in that are in different sets and in different sets in (i.e false positive (FP))
* , the number of pairs of elements in that are in the same set in and in different sets in (false negative (FN))
* , the number of pairs of elements in that are in different sets in and in the same set in (i.e. true negative (TN))

The Rand index, R, is:

Intuitively, can be considered as the number of agreements between and and as the number of disagreements between and .



The Rand index can be used to create a cluster F-measure.

Compare with standard Precision and Recall:

Compare with a confusion matrix:

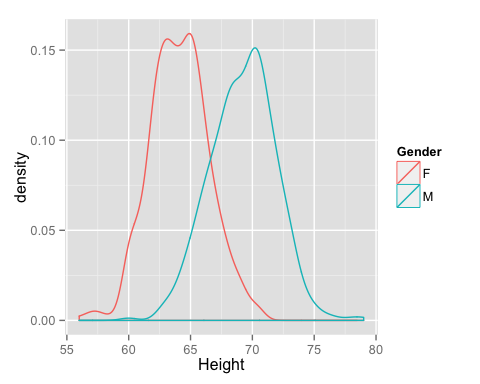
A confusion matrix is a 2x2 table with counts of the following in each of its quadrents.

true positive (TP)   
eqv. with hit  
true negative (TN)  
eqv. with correct rejection  
false positive (FP)  
eqv. with false alarm, Type I error  
false negative (FN)  
eqv. with miss, Type II error

# K-means clustering in R

K-means clustering in R. Can we cluster gender by height using the Galton regression data?

qplot(x=Height, data=h.gend, geom="density", group=Gender, color=Gender)



k<-2  
galton.2.clust<-kmeans(h.gend[,c("Height")],k)  
galton.2.clust

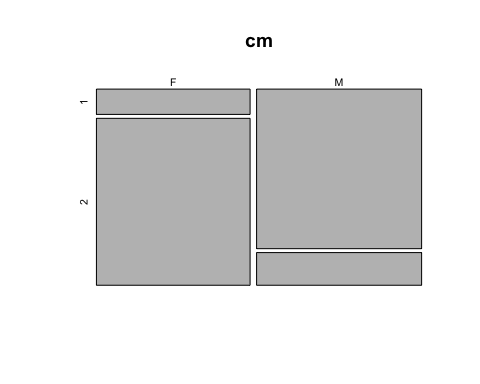
## K-means clustering with 2 clusters of sizes 443, 455  
##   
## Cluster means:  
## [,1]  
## 1 69.78420  
## 2 63.81692  
##   
## Clustering vector:  
## [1] 1 1 1 1 1 1 2 2 1 1 1 1 1 2 2 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 2 2 1 1  
## [36] 1 1 1 2 2 2 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2 2 2 2 1 1 1 1  
## [71] 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 2 2 1 1 2 1 1 1 1 2 2 1  
## [106] 1 2 2 1 1 1 2 2 2 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1  
## [141] 1 2 2 2 1 1 1 1 1 2 2 1 1 1 1 1 2 2 1 2 1 1 1 1 2 1 1 1 2 2 2 2 2 1 1  
## [176] 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 1 2 2 2 1 1 1 2 2 2 2 2 2 1 1 1 2 1 1 2  
## [211] 2 2 1 1 1 2 2 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1  
## [246] 2 2 2 1 1 1 1 2 1 1 2 2 2 2 2 1 1 1 1 1 2 2 2 2 1 1 1 2 2 2 2 2 1 2 2  
## [281] 2 2 1 1 1 2 2 2 1 1 1 1 1 2 2 1 1 2 1 1 1 1 1 2 2 2 2 1 1 1 1 2 2 1 1  
## [316] 1 2 2 1 1 2 2 2 1 1 1 1 1 1 2 2 2 1 2 2 2 1 1 1 1 1 1 1 2 2 1 1 2 2 2  
## [351] 2 2 2 1 1 1 2 2 1 1 1 2 1 1 2 2 1 2 2 2 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2  
## [386] 1 1 2 1 1 1 2 2 2 2 2 1 2 2 1 2 2 2 2 1 1 1 2 2 2 2 2 2 2 2 1 1 1 1 2  
## [421] 2 2 2 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 1 1 1 1 2 1 1 2 2 1 1 1 2 2 2 1 1  
## [456] 1 1 2 2 2 1 1 1 1 2 2 2 2 2 1 1 1 2 2 2 2 1 1 2 2 2 2 2 1 2 2 2 1 1 2  
## [491] 1 1 1 1 2 2 2 1 1 2 2 2 2 2 1 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 2 2 2 2 1  
## [526] 1 2 2 1 1 1 2 2 1 1 1 2 2 2 2 2 2 1 1 2 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1  
## [561] 1 1 1 1 2 1 2 2 2 1 1 1 2 2 2 2 1 1 1 2 1 1 2 2 2 2 2 2 1 1 1 1 2 2 2  
## [596] 2 2 2 2 2 2 2 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 1 1 1 2  
## [631] 1 1 1 2 2 1 1 2 2 2 2 2 2 1 1 2 2 2 2 1 1 1 1 1 2 2 1 1 2 2 1 1 2 2 2  
## [666] 2 2 2 2 2 2 2 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 1 2 2 1 1 1 2 2 2 2  
## [701] 1 1 2 2 2 1 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 2 2  
## [736] 1 1 2 1 1 1 1 2 1 1 1 1 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 1 2 2 1 1 2 2  
## [771] 2 2 1 1 2 2 2 2 2 2 1 1 1 2 2 1 2 2 1 1 2 2 2 2 1 1 2 2 2 2 2 1 1 1 2  
## [806] 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 2  
## [841] 1 2 1 1 1 1 2 2 1 1 2 2 2 2 1 2 2 2 2 1 1 1 2 1 1 1 1 2 1 1 2 2 2 2 2  
## [876] 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 1853.249 1669.160  
## (between\_SS / total\_SS = 69.4 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

Create confusion matrix and plot it. Note we can only do this becuase we know what the gender actually is. Note that runningh it twice can get different results.

cm<-table(h.gend$Gender,galton.2.clust$cluster) # Confustion Matrix  
cm

##   
## 1 2  
## F 57 376  
## M 386 79

plot(cm)



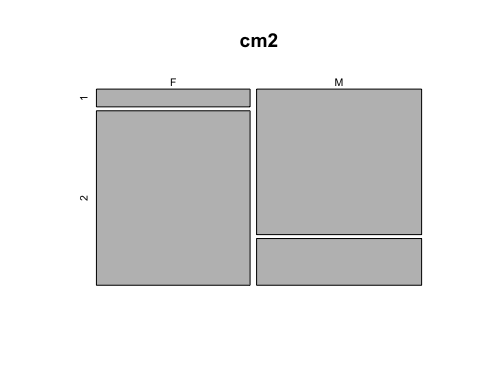
galton.2.clust<-kmeans(h.gend[,c("Height")],k)  
galton.2.clust

## K-means clustering with 2 clusters of sizes 392, 506  
##   
## Cluster means:  
## [,1]  
## 1 70.14643  
## 2 64.13775  
##   
## Clustering vector:  
## [1] 1 1 1 1 1 1 2 2 1 1 1 1 2 2 2 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 2 2 1 1  
## [36] 1 2 2 2 2 2 2 1 2 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2 2 2 2 1 1 1 1  
## [71] 1 1 1 2 1 1 1 1 1 2 1 1 1 2 2 2 2 2 2 2 1 1 1 2 2 1 2 2 1 2 1 1 2 2 1  
## [106] 1 2 2 1 1 1 2 2 2 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 2 2 2 1 1  
## [141] 1 2 2 2 1 2 1 1 1 2 2 1 1 1 1 1 2 2 1 2 1 1 1 1 2 1 1 1 2 2 2 2 2 1 1  
## [176] 1 2 1 1 1 1 1 1 2 2 2 2 1 2 1 2 2 2 2 1 1 1 2 2 2 2 2 2 1 1 1 2 1 1 2  
## [211] 2 2 1 1 1 2 2 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1  
## [246] 2 2 2 1 1 1 1 2 1 1 2 2 2 2 2 1 1 1 1 1 2 2 2 2 1 1 1 2 2 2 2 2 2 2 2  
## [281] 2 2 1 1 2 2 2 2 1 1 1 1 2 2 2 1 1 2 1 1 1 1 1 2 2 2 2 1 1 1 2 2 2 2 1  
## [316] 1 2 2 1 1 2 2 2 1 1 1 1 1 1 2 2 2 1 2 2 2 1 1 1 1 1 1 1 2 2 1 2 2 2 2  
## [351] 2 2 2 1 1 2 2 2 1 1 1 2 1 2 2 2 1 2 2 2 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2  
## [386] 1 1 2 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 1 1 2 2 2 2 2 2 2 2 1 1 1 1 2  
## [421] 2 2 2 1 1 1 1 1 1 1 2 2 1 1 1 2 2 2 1 1 1 1 2 1 1 2 2 1 1 1 2 2 2 1 1  
## [456] 1 1 2 2 2 1 1 1 1 2 2 2 2 2 1 1 2 2 2 2 2 1 1 2 2 2 2 2 1 2 2 2 1 1 2  
## [491] 1 1 1 1 2 2 2 1 2 2 2 2 2 2 1 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 2 2 2 2 1  
## [526] 1 2 2 1 1 1 2 2 1 1 1 2 2 2 2 2 2 1 1 2 1 2 2 2 1 1 2 1 1 2 1 1 1 1 1  
## [561] 1 1 1 1 2 1 2 2 2 1 1 1 2 2 2 2 1 1 1 2 1 1 2 2 2 2 2 2 1 1 1 2 2 2 2  
## [596] 2 2 2 2 2 2 2 1 1 1 1 2 2 1 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 1 1 1 2  
## [631] 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2 2 1 1 2 2 1 1 2 2 2  
## [666] 2 2 2 2 2 2 2 2 1 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 1 2 2 1 1 1 2 2 2 2  
## [701] 1 1 2 2 2 1 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 2 2 1 1 1 1 1 2 2 2  
## [736] 1 1 2 1 1 1 1 2 2 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2  
## [771] 2 2 1 1 2 2 2 2 2 2 1 1 2 2 2 1 2 2 1 2 2 2 2 2 1 1 2 2 2 2 2 1 1 2 2  
## [806] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 2  
## [841] 1 2 1 1 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 1 1 1 2 1 1 2 2 2 1 1 2 2 2 2 2  
## [876] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 1406.475 2133.809  
## (between\_SS / total\_SS = 69.3 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

cm2<-table(h.gend$Gender,galton.2.clust$cluster) # Confustion Matrix  
cm2

##   
## 1 2  
## F 40 393  
## M 352 113

plot(cm2)



Clustering the World Bank data.

head(wb)

## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 60.9 70.2 575 31.7  
## 2 77.5 13.3 4336 98.7  
## 3 71.0 21.6 4887 72.6  
## 4 44.9 2.7 37082 97.3  
## 5 48.8 2.2 44095 91.7  
## 6 51.9 101.6 5032 70.6

k<-2  
wb.2.clust<- kmeans(wb,k)   
wb.2.clust

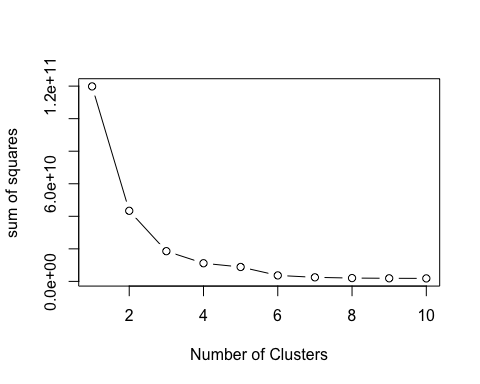
## K-means clustering with 2 clusters of sizes 37, 178  
##   
## Cluster means:  
## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 76.90000 3.421622 57846.000 94.80270  
## 2 68.72191 27.200562 7891.169 80.50955  
##   
## Clustering vector:  
## [1] 2 2 2 1 1 2 2 2 2 2 1 1 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 1  
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 2 2  
## [71] 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 1 2 2 1 2 1 2 2 2 2 2 2 2 1 2  
## [106] 2 2 2 2 2 2 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 1 1 2  
## [141] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2  
## [176] 2 2 2 2 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2  
## [211] 2 2 2 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 30877667926 12504335511  
## (between\_SS / total\_SS = 63.8 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

k<-3  
wb.3.clust<- kmeans(wb,k)   
wb.3.clust

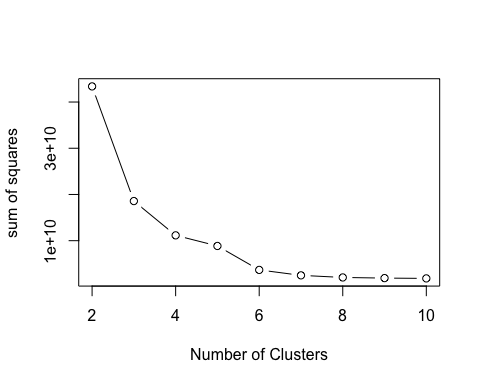
## K-means clustering with 3 clusters of sizes 56, 152, 7  
##   
## Cluster means:  
## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 77.04107 5.355357 36168.214 93.81786  
## 2 67.30329 30.578289 4992.513 78.45921  
## 3 76.20000 2.928571 108664.000 94.11429  
##   
## Clustering vector:  
## [1] 2 2 2 1 1 2 2 2 2 1 1 1 2 1 1 2 2 2 1 2 2 3 2 2 2 2 2 1 2 2 2 2 2 2 1  
## [36] 1 2 2 1 2 2 2 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 2 2  
## [71] 2 1 2 1 1 2 1 2 2 2 2 2 2 1 2 1 2 2 2 2 1 1 1 1 2 1 2 2 2 2 2 1 2 1 2  
## [106] 2 2 2 2 2 2 3 2 3 1 2 2 2 2 2 2 1 2 2 2 2 2 2 3 2 2 2 2 2 2 2 1 1 1 2  
## [141] 2 2 2 3 1 2 2 2 2 2 2 2 2 1 1 3 2 2 2 2 1 2 1 2 2 2 2 1 1 2 1 2 1 2 2  
## [176] 1 2 2 2 1 2 2 2 2 1 3 1 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 1 1 2 2 2 2 2  
## [211] 1 2 2 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 8725136622 3486673959 6362733207  
## (between\_SS / total\_SS = 84.5 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

Determining number of clusters. There is a number of methods for The issue of determining “the right number of clusters” including Hartigan’s rule, averaged Silhouette width and Gap statistic.

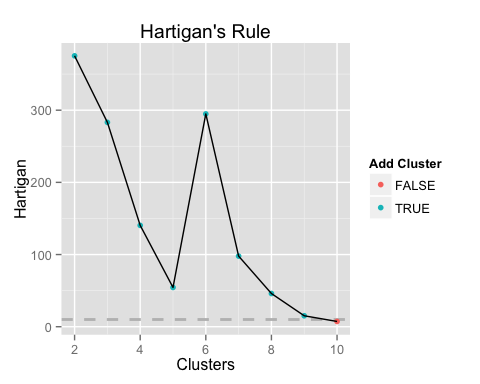
# Determining number of clusters   
sos <- (nrow(wb)-1)\*sum(apply(wb,2,var))  
for (i in 2:10) sos[i] <- sum(kmeans(wb, centers=i)$withinss)  
plot(1:10, sos, type="b", xlab="Number of Clusters", ylab="sum of squares")



plot(2:10, sos[c(2:10)], type="b", xlab="Number of Clusters", ylab="sum of squares")



# Hartigans's rule FitKMean (similarity)  
# require(useful)  
best<-FitKMeans(wb,max.clusters=10, seed=111)   
PlotHartigan(best)



A k of 6?

k<-6  
wb.6.clust<- kmeans(wb,k)   
wb.6.clust

## K-means clustering with 6 clusters of sizes 35, 33, 22, 2, 118, 5  
##   
## Cluster means:  
## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 76.91429 6.451429 27185.829 93.48286  
## 2 72.73333 12.687879 12420.576 96.22727  
## 3 77.46364 3.477273 49739.000 94.60909  
## 4 64.10000 2.150000 153558.500 95.95000  
## 5 65.66102 35.820339 2785.093 73.31186  
## 6 81.04000 3.240000 90706.200 93.38000  
##   
## Clustering vector:  
## [1] 5 5 5 1 3 5 2 2 5 1 3 3 5 1 1 5 2 5 3 5 5 6 5 5 5 5 2 1 5 5 5 5 5 5 3  
## [36] 1 5 5 1 2 5 5 5 5 5 2 5 2 5 1 1 2 3 5 5 5 5 5 5 2 5 2 5 3 5 3 3 1 2 5  
## [71] 5 3 5 1 1 2 1 5 5 5 5 5 5 1 2 3 5 5 5 5 3 1 1 1 5 1 5 2 5 5 5 1 5 3 5  
## [106] 5 2 2 5 5 5 4 2 6 3 5 5 5 2 5 5 1 5 5 2 2 5 5 4 5 5 5 5 5 5 5 3 1 1 5  
## [141] 5 5 1 6 1 5 2 2 5 5 5 5 2 1 1 6 2 2 5 5 3 5 1 5 5 2 5 3 1 2 1 5 1 5 5  
## [176] 1 5 2 5 3 5 5 2 5 3 6 1 1 5 5 5 5 5 5 2 5 2 5 1 5 5 5 3 3 3 2 5 5 2 5  
## [211] 1 5 5 5 5  
##   
## Within cluster sum of squares by cluster:  
## [1] 1046902366 326161381 1053132430 209613484 529085052 509674137  
## (between\_SS / total\_SS = 96.9 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

trails<-33  
wb.6.clust.33<- kmeans(wb,k, nstart = trails)   
wb.6.clust.33

## K-means clustering with 6 clusters of sizes 118, 33, 5, 35, 2, 22  
##   
## Cluster means:  
## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 65.66102 35.820339 2785.093 73.31186  
## 2 72.73333 12.687879 12420.576 96.22727  
## 3 81.04000 3.240000 90706.200 93.38000  
## 4 76.91429 6.451429 27185.829 93.48286  
## 5 64.10000 2.150000 153558.500 95.95000  
## 6 77.46364 3.477273 49739.000 94.60909  
##   
## Clustering vector:  
## [1] 1 1 1 4 6 1 2 2 1 4 6 6 1 4 4 1 2 1 6 1 1 3 1 1 1 1 2 4 1 1 1 1 1 1 6  
## [36] 4 1 1 4 2 1 1 1 1 1 2 1 2 1 4 4 2 6 1 1 1 1 1 1 2 1 2 1 6 1 6 6 4 2 1  
## [71] 1 6 1 4 4 2 4 1 1 1 1 1 1 4 2 6 1 1 1 1 6 4 4 4 1 4 1 2 1 1 1 4 1 6 1  
## [106] 1 2 2 1 1 1 5 2 3 6 1 1 1 2 1 1 4 1 1 2 2 1 1 5 1 1 1 1 1 1 1 6 4 4 1  
## [141] 1 1 4 3 4 1 2 2 1 1 1 1 2 4 4 3 2 2 1 1 6 1 4 1 1 2 1 6 4 2 4 1 4 1 1  
## [176] 4 1 2 1 6 1 1 2 1 6 3 4 4 1 1 1 1 1 1 2 1 2 1 4 1 1 1 6 6 6 2 1 1 2 1  
## [211] 4 1 1 1 1  
##   
## Within cluster sum of squares by cluster:  
## [1] 529085052 326161381 509674137 1046902366 209613484 1053132430  
## (between\_SS / total\_SS = 96.9 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

cm<-table(dwb$Country,wb.6.clust.33$cluster) # Confustion Matrix  
cm

##   
## 1 2 3 4 5 6  
## Afghanistan 1 0 0 0 0 0  
## Albania 1 0 0 0 0 0  
## Algeria 1 0 0 0 0 0  
## American Samoa 0 0 0 1 0 0  
## Andorra 0 0 0 0 0 1  
## Angola 1 0 0 0 0 0  
## Antigua and Barbuda 0 1 0 0 0 0  
## Argentina 0 1 0 0 0 0  
## Armenia 1 0 0 0 0 0  
## Aruba 0 0 0 1 0 0  
## Australia 0 0 0 0 0 1  
## Austria 0 0 0 0 0 1  
## Azerbaijan 1 0 0 0 0 0  
## Bahamas, The 0 0 0 1 0 0  
## Bahrain 0 0 0 1 0 0  
## Bangladesh 1 0 0 0 0 0  
## Barbados 0 1 0 0 0 0  
## Belarus 1 0 0 0 0 0  
## Belgium 0 0 0 0 0 1  
## Belize 1 0 0 0 0 0  
## Benin 1 0 0 0 0 0  
## Bermuda 0 0 1 0 0 0  
## Bhutan 1 0 0 0 0 0  
## Bolivia 1 0 0 0 0 0  
## Bosnia and Herzegovina 1 0 0 0 0 0  
## Botswana 1 0 0 0 0 0  
## Brazil 0 1 0 0 0 0  
## Brunei Darussalam 0 0 0 1 0 0  
## Bulgaria 1 0 0 0 0 0  
## Burkina Faso 1 0 0 0 0 0  
## Burundi 1 0 0 0 0 0  
## C\xf4te d'Ivoire 1 0 0 0 0 0  
## Cabo Verde 1 0 0 0 0 0  
## Cambodia 1 0 0 0 0 0  
## Cameroon 1 0 0 0 0 0  
## Canada 0 0 0 0 0 1  
## Cayman Islands 0 0 0 1 0 0  
## Central African Republic 1 0 0 0 0 0  
## Chad 1 0 0 0 0 0  
## Channel Islands 0 0 0 1 0 0  
## Chile 0 1 0 0 0 0  
## China 1 0 0 0 0 0  
## Colombia 1 0 0 0 0 0  
## Comoros 1 0 0 0 0 0  
## Congo, Dem. Rep. 1 0 0 0 0 0  
## Congo, Rep. 1 0 0 0 0 0  
## Costa Rica 0 1 0 0 0 0  
## Croatia 0 1 0 0 0 0  
## Cuba 1 0 0 0 0 0  
## Cura\xe7ao 0 0 0 1 0 0  
## Cyprus 0 0 0 1 0 0  
## Czech Republic 0 1 0 0 0 0  
## Denmark 0 0 0 0 0 1  
## Djibouti 1 0 0 0 0 0  
## Dominica 1 0 0 0 0 0  
## Dominican Republic 1 0 0 0 0 0  
## Ecuador 1 0 0 0 0 0  
## Egypt, Arab Rep. 1 0 0 0 0 0  
## El Salvador 1 0 0 0 0 0  
## Equatorial Guinea 0 1 0 0 0 0  
## Eritrea 1 0 0 0 0 0  
## Estonia 0 1 0 0 0 0  
## Ethiopia 1 0 0 0 0 0  
## Faeroe Islands 0 0 0 0 0 1  
## Fiji 1 0 0 0 0 0  
## Finland 0 0 0 0 0 1  
## France 0 0 0 0 0 1  
## French Polynesia 0 0 0 1 0 0  
## Gabon 0 1 0 0 0 0  
## Gambia, The 1 0 0 0 0 0  
## Georgia 1 0 0 0 0 0  
## Germany 0 0 0 0 0 1  
## Ghana 1 0 0 0 0 0  
## Greece 0 0 0 1 0 0  
## Greenland 0 0 0 1 0 0  
## Grenada 0 1 0 0 0 0  
## Guam 0 0 0 1 0 0  
## Guatemala 1 0 0 0 0 0  
## Guinea 1 0 0 0 0 0  
## Guinea-Bissau 1 0 0 0 0 0  
## Guyana 1 0 0 0 0 0  
## Haiti 1 0 0 0 0 0  
## Honduras 1 0 0 0 0 0  
## Hong Kong SAR, China 0 0 0 1 0 0  
## Hungary 0 1 0 0 0 0  
## Iceland 0 0 0 0 0 1  
## India 1 0 0 0 0 0  
## Indonesia 1 0 0 0 0 0  
## Iran, Islamic Rep. 1 0 0 0 0 0  
## Iraq 1 0 0 0 0 0  
## Ireland 0 0 0 0 0 1  
## Isle of Man 0 0 0 1 0 0  
## Israel 0 0 0 1 0 0  
## Italy 0 0 0 1 0 0  
## Jamaica 1 0 0 0 0 0  
## Japan 0 0 0 1 0 0  
## Jordan 1 0 0 0 0 0  
## Kazakhstan 0 1 0 0 0 0  
## Kenya 1 0 0 0 0 0  
## Kiribati 1 0 0 0 0 0  
## Korea, Dem. Rep. 1 0 0 0 0 0  
## Korea, Rep. 0 0 0 1 0 0  
## Kosovo 1 0 0 0 0 0  
## Kuwait 0 0 0 0 0 1  
## Kyrgyz Republic 1 0 0 0 0 0  
## Lao PDR 1 0 0 0 0 0  
## Latvia 0 1 0 0 0 0  
## Lebanon 0 1 0 0 0 0  
## Lesotho 1 0 0 0 0 0  
## Liberia 1 0 0 0 0 0  
## Libya 1 0 0 0 0 0  
## Liechtenstein 0 0 0 0 1 0  
## Lithuania 0 1 0 0 0 0  
## Luxembourg 0 0 1 0 0 0  
## Macao SAR, China 0 0 0 0 0 1  
## Macedonia, FYR 1 0 0 0 0 0  
## Madagascar 1 0 0 0 0 0  
## Malawi 1 0 0 0 0 0  
## Malaysia 0 1 0 0 0 0  
## Maldives 1 0 0 0 0 0  
## Mali 1 0 0 0 0 0  
## Malta 0 0 0 1 0 0  
## Marshall Islands 1 0 0 0 0 0  
## Mauritania 1 0 0 0 0 0  
## Mauritius 0 1 0 0 0 0  
## Mexico 0 1 0 0 0 0  
## Micronesia, Fed. Sts. 1 0 0 0 0 0  
## Moldova 1 0 0 0 0 0  
## Monaco 0 0 0 0 1 0  
## Mongolia 1 0 0 0 0 0  
## Montenegro 1 0 0 0 0 0  
## Morocco 1 0 0 0 0 0  
## Mozambique 1 0 0 0 0 0  
## Myanmar 1 0 0 0 0 0  
## Namibia 1 0 0 0 0 0  
## Nepal 1 0 0 0 0 0  
## Netherlands 0 0 0 0 0 1  
## New Caledonia 0 0 0 1 0 0  
## New Zealand 0 0 0 1 0 0  
## Nicaragua 1 0 0 0 0 0  
## Niger 1 0 0 0 0 0  
## Nigeria 1 0 0 0 0 0  
## Northern Mariana Islands 0 0 0 1 0 0  
## Norway 0 0 1 0 0 0  
## Oman 0 0 0 1 0 0  
## Pakistan 1 0 0 0 0 0  
## Palau 0 1 0 0 0 0  
## Panama 0 1 0 0 0 0  
## Papua New Guinea 1 0 0 0 0 0  
## Paraguay 1 0 0 0 0 0  
## Peru 1 0 0 0 0 0  
## Philippines 1 0 0 0 0 0  
## Poland 0 1 0 0 0 0  
## Portugal 0 0 0 1 0 0  
## Puerto Rico 0 0 0 1 0 0  
## Qatar 0 0 1 0 0 0  
## Romania 0 1 0 0 0 0  
## Russian Federation 0 1 0 0 0 0  
## Rwanda 1 0 0 0 0 0  
## S\xe3o Tom\xe9 and Principe 1 0 0 0 0 0  
## Samoa 1 0 0 0 0 0  
## San Marino 0 0 0 0 0 1  
## Saudi Arabia 0 0 0 1 0 0  
## Senegal 1 0 0 0 0 0  
## Serbia 1 0 0 0 0 0  
## Seychelles 0 1 0 0 0 0  
## Sierra Leone 1 0 0 0 0 0  
## Singapore 0 0 0 0 0 1  
## Sint Maarten (Dutch part) 0 0 0 1 0 0  
## Slovak Republic 0 1 0 0 0 0  
## Slovenia 0 0 0 1 0 0  
## Solomon Islands 1 0 0 0 0 0  
## Somalia 0 0 0 1 0 0  
## South Africa 1 0 0 0 0 0  
## South Sudan 1 0 0 0 0 0  
## Spain 0 0 0 1 0 0  
## Sri Lanka 1 0 0 0 0 0  
## St. Kitts and Nevis 0 1 0 0 0 0  
## St. Lucia 1 0 0 0 0 0  
## St. Martin (French part) 0 0 0 0 0 1  
## St. Vincent and the Grenadines 1 0 0 0 0 0  
## Sudan 1 0 0 0 0 0  
## Suriname 0 1 0 0 0 0  
## Swaziland 1 0 0 0 0 0  
## Sweden 0 0 0 0 0 1  
## Switzerland 0 0 1 0 0 0  
## Syrian Arab Republic 0 0 0 1 0 0  
## Taiwan, China 0 0 0 1 0 0  
## Tajikistan 1 0 0 0 0 0  
## Tanzania 1 0 0 0 0 0  
## Thailand 1 0 0 0 0 0  
## Timor-Leste 1 0 0 0 0 0  
## Togo 1 0 0 0 0 0  
## Tonga 1 0 0 0 0 0  
## Trinidad and Tobago 0 1 0 0 0 0  
## Tunisia 1 0 0 0 0 0  
## Turkey 0 1 0 0 0 0  
## Turkmenistan 1 0 0 0 0 0  
## Turks and Caicos Islands 0 0 0 1 0 0  
## Tuvalu 1 0 0 0 0 0  
## Uganda 1 0 0 0 0 0  
## Ukraine 1 0 0 0 0 0  
## United Arab Emirates 0 0 0 0 0 1  
## United Kingdom 0 0 0 0 0 1  
## United States 0 0 0 0 0 1  
## Uruguay 0 1 0 0 0 0  
## Uzbekistan 1 0 0 0 0 0  
## Vanuatu 1 0 0 0 0 0  
## Venezuela, RB 0 1 0 0 0 0  
## Vietnam 1 0 0 0 0 0  
## Virgin Islands (U.S.) 0 0 0 1 0 0  
## West Bank and Gaza 1 0 0 0 0 0  
## Yemen, Rep. 1 0 0 0 0 0  
## Zambia 1 0 0 0 0 0  
## Zimbabwe 1 0 0 0 0 0

Multidimensional scaling (MDS)

# require useful mds  
# require(useful)  
# plot(wb.2.clust)

## Evaluating model performance

# Evaluating model performance   
# look at the size of the clusters  
wb.6.clust$size

## [1] 35 33 22 2 118 5

# look at the cluster centers  
wb.6.clust$centers

## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 76.91429 6.451429 27185.829 93.48286  
## 2 72.73333 12.687879 12420.576 96.22727  
## 3 77.46364 3.477273 49739.000 94.60909  
## 4 64.10000 2.150000 153558.500 95.95000  
## 5 65.66102 35.820339 2785.093 73.31186  
## 6 81.04000 3.240000 90706.200 93.38000

names(wb)

## [1] "Life.expectancy" "Infant.mortality" "Per.capita.income"  
## [4] "Literacy"

# mean of 'Per.capita.income' by cluster   
pci<-aggregate(data = wb, Per.capita.income ~ wb.6.clust$cluster, mean)  
pci

## wb.6.clust$cluster Per.capita.income  
## 1 1 27185.829  
## 2 2 12420.576  
## 3 3 49739.000  
## 4 4 153558.500  
## 5 5 2785.093  
## 6 6 90706.200

# mean of 'Literacy' by cluster   
l<-aggregate(data = wb, Literacy ~ wb.6.clust$cluster, mean)  
l

## wb.6.clust$cluster Literacy  
## 1 1 93.48286  
## 2 2 96.22727  
## 3 3 94.60909  
## 4 4 95.95000  
## 5 5 73.31186  
## 6 6 93.38000

# mean of 'Infant.mortality' by cluster   
im<-aggregate(data = wb, Infant.mortality ~ wb.6.clust$cluster, mean)  
im

## wb.6.clust$cluster Infant.mortality  
## 1 1 6.451429  
## 2 2 12.687879  
## 3 3 3.477273  
## 4 4 2.150000  
## 5 5 35.820339  
## 6 6 3.240000

# mean 'Life.expectancy' by cluster   
le<-aggregate(data = wb, Life.expectancy ~ wb.6.clust$cluster, mean)  
le

## wb.6.clust$cluster Life.expectancy  
## 1 1 76.91429  
## 2 2 72.73333  
## 3 3 77.46364  
## 4 4 64.10000  
## 5 5 65.66102  
## 6 6 81.04000

cm.3=table(wb$Per.capita.income,wb.6.clust$cluster)  
cm.3

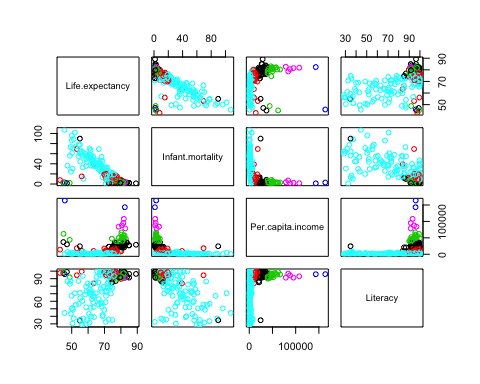
##   
## 1 2 3 4 5 6  
## 237 0 0 0 0 1 0  
## 253 0 0 0 0 1 0  
## 366 0 0 0 0 1 0  
## 379 0 0 0 0 1 0  
## 381 0 0 0 0 1 0  
## 392 0 0 0 0 1 0  
## 422 0 0 0 0 1 0  
## 423 0 0 0 0 1 0  
## 434 0 0 0 0 1 0  
## 445 0 0 0 0 1 0  
## 477 0 0 0 0 1 0  
## 530 0 0 0 0 1 0  
## 571 0 0 0 0 2 0  
## 575 0 0 0 0 1 0  
## 576 0 0 0 0 1 0  
## 577 0 0 0 0 1 0  
## 581 0 0 0 0 1 0  
## 619 0 0 0 0 1 0  
## 641 0 0 0 0 1 0  
## 704 0 0 0 0 1 0  
## 743 0 0 0 0 1 0  
## 753 0 0 0 0 1 0  
## 760 0 0 0 0 1 0  
## 794 0 0 0 0 1 0  
## 815 0 0 0 0 1 0  
## 832 0 0 0 0 1 0  
## 863 0 0 0 0 1 0  
## 882 0 0 0 0 1 0  
## 961 0 0 0 0 1 0  
## 994 0 0 0 0 1 0  
## 995 0 0 0 0 1 0  
## 1004 0 0 0 0 1 0  
## 1048 0 0 0 0 1 0  
## 1082 0 0 0 0 2 0  
## 1113 0 0 0 0 1 0  
## 1158 0 0 0 0 1 0  
## 1198 0 0 0 0 1 0  
## 1236 0 0 0 0 1 0  
## 1247 0 0 0 0 1 0  
## 1283 0 0 0 0 1 0  
## 1344 0 0 0 0 1 0  
## 1365 0 0 0 0 1 0  
## 1387 0 0 0 0 1 0  
## 1400 0 0 0 0 1 0  
## 1462 0 0 0 0 1 0  
## 1505 0 0 0 0 1 0  
## 1537 0 0 0 0 1 0  
## 1551 0 0 0 0 1 0  
## 1570 0 0 0 0 1 0  
## 1573 0 0 0 0 1 0  
## 1581 0 0 0 0 1 0  
## 1618 0 0 0 0 1 0  
## 1651 0 0 0 0 1 0  
## 1656 0 0 0 0 1 0  
## 1907 0 0 0 0 1 0  
## 2165 0 0 0 0 1 0  
## 2217 0 0 0 0 1 0  
## 2309 0 0 0 0 1 0  
## 2326 0 0 0 0 1 0  
## 2350 0 0 0 0 1 0  
## 2510 0 0 0 0 1 0  
## 2682 0 0 0 0 1 0  
## 2734 0 0 0 0 1 0  
## 2869 0 0 0 0 1 0  
## 2914 0 0 0 0 1 0  
## 2956 0 0 0 0 1 0  
## 2986 0 0 0 0 1 0  
## 3031 0 0 0 0 1 0  
## 3082 0 0 0 0 1 0  
## 3141 0 0 0 0 1 0  
## 3161 0 0 0 0 1 0  
## 3173 0 0 0 0 1 0  
## 3221 0 0 0 0 1 0  
## 3231 0 0 0 0 1 0  
## 3324 0 0 0 0 1 0  
## 3412 0 0 0 0 1 0  
## 3502 0 0 0 0 1 0  
## 3544 0 0 0 0 1 0  
## 3567 0 0 0 0 1 0  
## 3625 0 0 0 0 1 0  
## 3647 0 0 0 0 1 0  
## 3822 0 0 0 0 1 0  
## 3823 0 0 0 0 1 0  
## 4102 0 0 0 0 1 0  
## 4256 0 0 0 0 1 0  
## 4336 0 0 0 0 1 0  
## 4621 0 0 0 0 1 0  
## 4626 0 0 0 0 1 0  
## 4681 0 0 0 0 1 0  
## 4824 0 0 0 0 1 0  
## 4887 0 0 0 0 1 0  
## 4987 0 0 0 0 1 0  
## 5032 0 0 0 0 1 0  
## 5112 0 0 0 0 1 0  
## 5145 0 0 0 0 1 0  
## 5183 0 0 0 0 1 0  
## 5293 0 0 0 0 1 0  
## 5454 0 0 0 0 1 0  
## 5510 0 0 0 0 1 0  
## 5511 0 0 0 0 1 0  
## 5565 0 0 0 0 1 0  
## 5965 0 0 0 0 1 0  
## 6007 0 0 0 0 1 0  
## 6075 0 0 0 0 1 0  
## 6365 0 0 0 0 1 0  
## 6478 0 0 0 0 1 0  
## 6575 0 0 0 0 1 0  
## 6594 0 0 0 0 1 0  
## 6664 0 0 0 0 1 0  
## 6735 0 0 0 0 1 0  
## 6837 0 0 0 0 1 0  
## 6987 0 0 0 0 1 0  
## 7069 0 0 0 0 1 0  
## 7096 0 0 0 0 1 0  
## 7205 0 0 0 0 1 0  
## 7436 0 0 0 0 1 0  
## 7699 0 1 0 0 0 0  
## 8363 0 1 0 0 0 0  
## 8431 0 1 0 0 0 0  
## 8597 0 1 0 0 0 0  
## 8985 0 1 0 0 0 0  
## 9022 0 1 0 0 0 0  
## 9099 0 1 0 0 0 0  
## 9478 0 1 0 0 0 0  
## 9499 0 1 0 0 0 0  
## 10020 0 1 0 0 0 0  
## 10142 0 1 0 0 0 0  
## 10277 0 1 0 0 0 0  
## 10588 0 1 0 0 0 0  
## 11105 0 1 0 0 0 0  
## 12214 0 1 0 0 0 0  
## 12517 0 1 0 0 0 0  
## 12532 0 1 0 0 0 0  
## 12644 0 1 0 0 0 0  
## 13277 0 1 0 0 0 0  
## 13340 0 1 0 0 0 0  
## 13349 0 1 0 0 0 0  
## 13507 0 1 0 0 0 0  
## 13660 0 1 0 0 0 0  
## 13776 0 1 0 0 0 0  
## 14029 0 1 0 0 0 0  
## 14164 0 1 0 0 0 0  
## 14229 0 1 0 0 0 0  
## 15199 0 1 0 0 0 0  
## 17192 0 1 0 0 0 0  
## 17401 0 1 0 0 0 0  
## 17601 0 1 0 0 0 0  
## 18389 0 1 0 0 0 0  
## 19554 0 1 0 0 0 0  
## 20342 1 0 0 0 0 0  
## 20734 1 0 0 0 0 0  
## 20832 1 0 0 0 0 0  
## 21000 1 0 0 0 0 0  
## 21088 1 0 0 0 0 0  
## 21683 1 0 0 0 0 0  
## 22081 1 0 0 0 0 0  
## 22246 1 0 0 0 0 0  
## 22335 1 0 0 0 0 0  
## 22648 1 0 0 0 0 0  
## 22682 1 0 0 0 0 0  
## 22843 1 0 0 0 0 0  
## 23363 1 0 0 0 0 0  
## 23760 1 0 0 0 0 0  
## 23963 1 0 0 0 0 0  
## 24427 1 0 0 0 0 0  
## 25458 1 0 0 0 0 0  
## 26695 1 0 0 0 0 0  
## 26714 1 0 0 0 0 0  
## 26793 1 0 0 0 0 0  
## 27194 1 0 0 0 0 0  
## 27659 1 0 0 0 0 0  
## 27964 1 0 0 0 0 0  
## 28920 1 0 0 0 0 0  
## 30097 1 0 0 0 0 0  
## 30262 1 0 0 0 0 0  
## 32325 1 0 0 0 0 0  
## 34364 1 0 0 0 0 0  
## 34960 1 0 0 0 0 0  
## 35021 1 0 0 0 0 0  
## 35374 1 0 0 0 0 0  
## 35727 1 0 0 0 0 0  
## 36194 1 0 0 0 0 0  
## 36674 1 0 0 0 0 0  
## 37082 1 0 0 0 0 0  
## 39726 0 0 1 0 0 0  
## 39798 0 0 1 0 0 0  
## 41430 0 0 1 0 0 0  
## 42541 0 0 1 0 0 0  
## 44095 0 0 1 0 0 0  
## 44673 0 0 1 0 0 0  
## 46426 0 0 1 0 0 0  
## 46784 0 0 1 0 0 0  
## 47553 0 0 1 0 0 0  
## 48471 0 0 1 0 0 0  
## 48861 0 0 1 0 0 0  
## 49110 0 0 1 0 0 0  
## 49262 0 0 1 0 0 0  
## 49536 0 0 1 0 0 0  
## 50375 0 0 1 0 0 0  
## 51590 0 0 1 0 0 0  
## 52070 0 0 1 0 0 0  
## 55716 0 0 1 0 0 0  
## 57594 0 0 1 0 0 0  
## 59874 0 0 1 0 0 0  
## 62191 0 0 1 0 0 0  
## 66582 0 0 1 0 0 0  
## 78683 0 0 0 0 0 1  
## 83922 0 0 0 0 0 1  
## 87909 0 0 0 0 0 1  
## 95268 0 0 0 0 0 1  
## 107749 0 0 0 0 0 1  
## 143321 0 0 0 1 0 0  
## 163796 0 0 0 1 0 0

print(wb.6.clust)

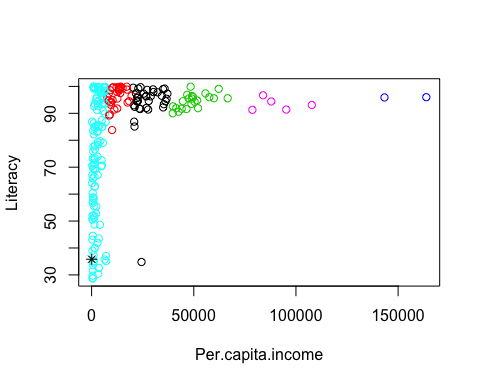
## K-means clustering with 6 clusters of sizes 35, 33, 22, 2, 118, 5  
##   
## Cluster means:  
## Life.expectancy Infant.mortality Per.capita.income Literacy  
## 1 76.91429 6.451429 27185.829 93.48286  
## 2 72.73333 12.687879 12420.576 96.22727  
## 3 77.46364 3.477273 49739.000 94.60909  
## 4 64.10000 2.150000 153558.500 95.95000  
## 5 65.66102 35.820339 2785.093 73.31186  
## 6 81.04000 3.240000 90706.200 93.38000  
##   
## Clustering vector:  
## [1] 5 5 5 1 3 5 2 2 5 1 3 3 5 1 1 5 2 5 3 5 5 6 5 5 5 5 2 1 5 5 5 5 5 5 3  
## [36] 1 5 5 1 2 5 5 5 5 5 2 5 2 5 1 1 2 3 5 5 5 5 5 5 2 5 2 5 3 5 3 3 1 2 5  
## [71] 5 3 5 1 1 2 1 5 5 5 5 5 5 1 2 3 5 5 5 5 3 1 1 1 5 1 5 2 5 5 5 1 5 3 5  
## [106] 5 2 2 5 5 5 4 2 6 3 5 5 5 2 5 5 1 5 5 2 2 5 5 4 5 5 5 5 5 5 5 3 1 1 5  
## [141] 5 5 1 6 1 5 2 2 5 5 5 5 2 1 1 6 2 2 5 5 3 5 1 5 5 2 5 3 1 2 1 5 1 5 5  
## [176] 1 5 2 5 3 5 5 2 5 3 6 1 1 5 5 5 5 5 5 2 5 2 5 1 5 5 5 3 3 3 2 5 5 2 5  
## [211] 1 5 5 5 5  
##   
## Within cluster sum of squares by cluster:  
## [1] 1046902366 326161381 1053132430 209613484 529085052 509674137  
## (between\_SS / total\_SS = 96.9 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

### Plotting Clusters

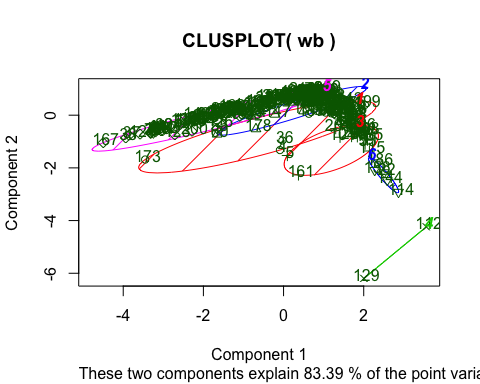
plot(wb,col=wb.6.clust$cluster) # Plot Clusters



plot(wb[c("Per.capita.income","Literacy")],col=wb.6.clust$cluster)   
points(wb.6.clust$centers, col = 1:2, pch = 8)



# Centroid Plot against 1st two discriminant functions  
clusplot(wb, wb.6.clust$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)



# library(fpc)  
# plotcluster(wb,wb.6.clust$cluster)

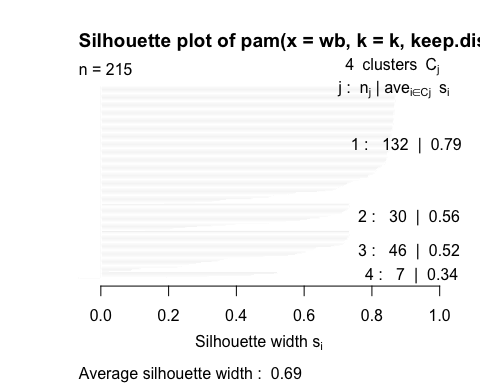
# K-medoids clustering in R

K-medoids clustering in R. PAM can handle catagerical data and is robust to outliers

# PAM  
k<-4  
wb.pam.4.clust<- pam(wb,k, keep.diss = TRUE, keep.data = TRUE)  
wb.pam.4.clust

## Medoids:  
## ID Life.expectancy Infant.mortality Per.capita.income Literacy  
## [1,] 58 71.1 18.6 2914 73.9  
## [2,] 19 80.4 3.5 46426 91.6  
## [3,] 122 80.7 5.3 21088 92.4  
## [4,] 144 81.5 2.3 95268 91.4  
## Clustering vector:  
## [1] 1 1 1 2 2 1 3 3 1 3 2 2 1 3 3 1 3 1 2 1 1 4 1 1 1 1 1 2 1 1 1 1 1 1 2  
## [36] 3 1 1 3 3 1 1 1 1 1 1 1 3 1 3 3 3 2 1 1 1 1 1 1 3 1 3 1 2 1 2 2 2 1 1  
## [71] 1 2 1 3 3 1 3 1 1 1 1 1 1 2 3 2 1 1 1 1 2 3 3 2 1 2 1 1 1 1 1 3 1 2 1  
## [106] 1 3 1 1 1 1 4 3 4 2 1 1 1 1 1 1 3 1 1 1 1 1 1 4 1 1 1 1 1 1 1 2 2 2 1  
## [141] 1 1 3 4 3 1 1 1 1 1 1 1 3 3 3 4 1 3 1 1 2 1 3 1 1 3 1 2 3 3 3 1 3 1 1  
## [176] 3 1 3 1 2 1 1 1 1 2 4 3 3 1 1 1 1 1 1 3 1 1 1 3 1 1 1 2 2 2 3 1 1 3 1  
## [211] 3 1 1 1 1  
## Objective function:  
## build swap   
## 4699.241 4209.529   
##   
## Available components:  
## [1] "medoids" "id.med" "clustering" "objective" "isolation"   
## [6] "clusinfo" "silinfo" "diss" "call" "data"

plot(wb.pam.4.clust, which.plots = 2)



# long lines good - means greater within cluster similarity

## Gap statistic

Gap statistic

clusGap() calculates a goodness of clustering measure, the “gap” statistic. For each number of clusters k, it compares with where the latter is defined via bootstrapping, i.e. simulating from a reference distribution.

maxSE(f, SE.f) determines the location of the maximum of f, taking a “1-SE rule” into account for the *SE* methods. The default method "firstSEmax" looks for the smallest k such that its value f(k) is not more than 1 standard error away from the first local maximum. This is similar but not the same as "Tibs2001SEmax", Tibshirani et al's recommendation of determining the number of clusters from the gap statistics and their standard deviations.

See [clusGap](https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/clusGap.html)

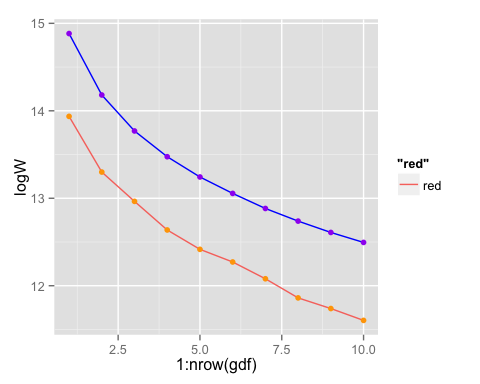
gap<-clusGap(wb,FUNcluster=pam,K.max=10) # Bootstrapping  
gap$Tab

## logW E.logW gap SE.sim  
## [1,] 13.93632 14.88329 0.9469673 0.03157772  
## [2,] 13.29982 14.18070 0.8808819 0.03261127  
## [3,] 12.96461 13.76945 0.8048481 0.03098882  
## [4,] 12.63747 13.47494 0.8374659 0.03221813  
## [5,] 12.41600 13.24335 0.8273458 0.03145102  
## [6,] 12.27150 13.05556 0.7840653 0.03666262  
## [7,] 12.07968 12.88335 0.8036681 0.03349781  
## [8,] 11.86096 12.73889 0.8779310 0.02760994  
## [9,] 11.73915 12.60920 0.8700535 0.02723799  
## [10,] 11.60367 12.49516 0.8914850 0.02619031

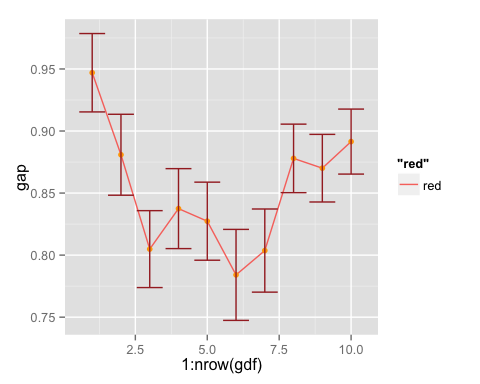
gdf<-as.data.frame(gap$Tab)  
head(gdf)

## logW E.logW gap SE.sim  
## 1 13.93632 14.88329 0.9469673 0.03157772  
## 2 13.29982 14.18070 0.8808819 0.03261127  
## 3 12.96461 13.76945 0.8048481 0.03098882  
## 4 12.63747 13.47494 0.8374659 0.03221813  
## 5 12.41600 13.24335 0.8273458 0.03145102  
## 6 12.27150 13.05556 0.7840653 0.03666262

qplot(x=1:nrow(gdf),y=logW,data = gdf,geom="line",color="red")+geom\_point(aes(y=logW),color="orange")+geom\_line(aes(y=E.logW),color="blue")+geom\_point(aes(y=E.logW),color="purple")



# Gap statistic  
qplot(x=1:nrow(gdf),y=gap,data = gdf,geom="line",color="red")+geom\_point(aes(y=gap),color="orange")+geom\_errorbar(aes(ymin=gap-SE.sim,ymax=gap+SE.sim),color="brown")

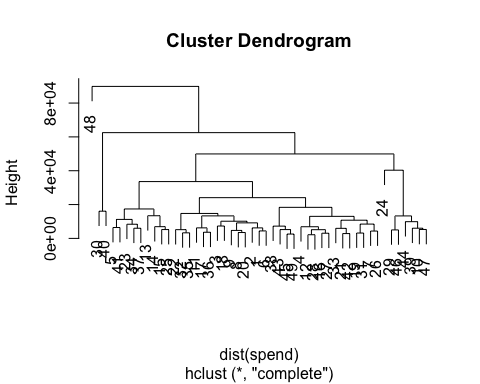


# Hierarchical clustering in R

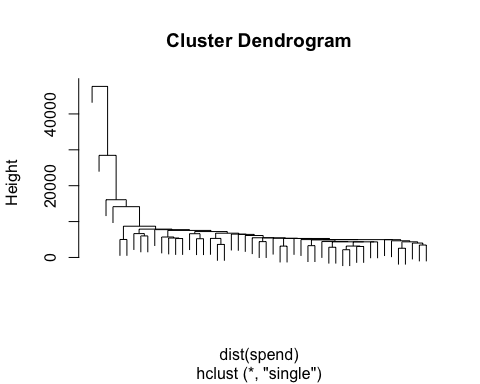
Hierarchical clustering in R

Hierarchical Clustering for the wholesale customers data set.

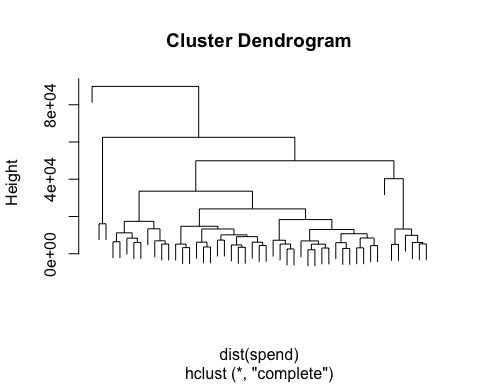
spend.h.clust<- hclust(d=dist(spend))  
plot(spend.h.clust)



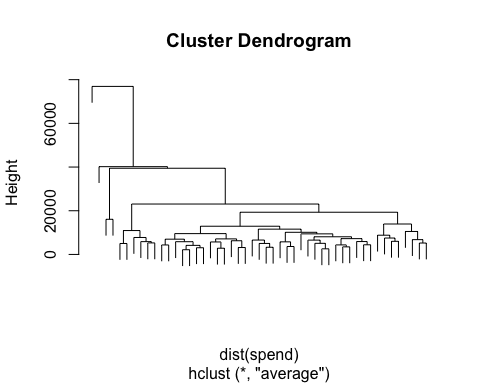
spend.h.clust.si<- hclust(dist(spend), method = "single")  
spend.h.clust.co<- hclust(dist(spend), method = "complete")  
spend.h.clust.av<- hclust(dist(spend), method = "average")  
spend.h.clust.ce<- hclust(dist(spend), method = "centroid")  
plot(spend.h.clust.si, labels = FALSE)



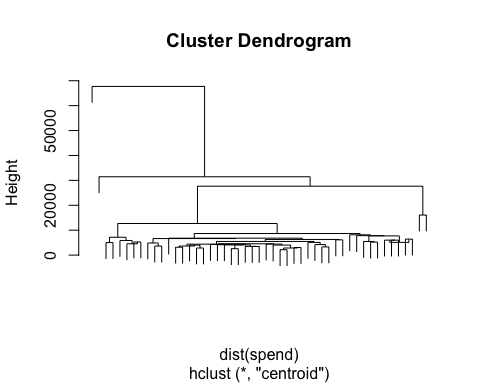
plot(spend.h.clust.co, labels = FALSE)



plot(spend.h.clust.av, labels = FALSE)

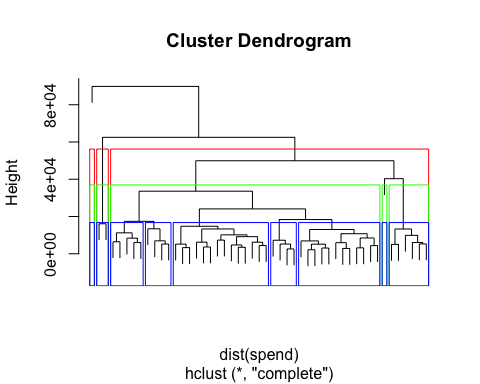


plot(spend.h.clust.ce, labels = FALSE)



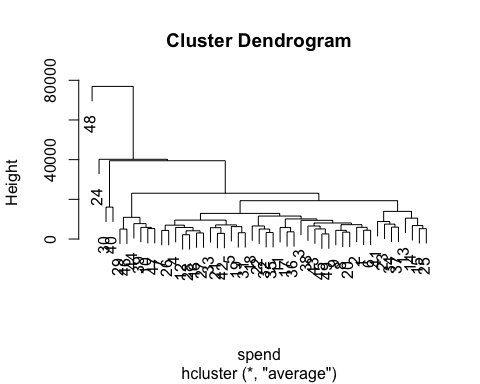
Plotting to deterimine the cluster level.

plot(spend.h.clust, labels = FALSE)  
rect.hclust(spend.h.clust, k=3, border="red")  
rect.hclust(spend.h.clust, k=5, border="green")  
rect.hclust(spend.h.clust, k=9, border="blue")

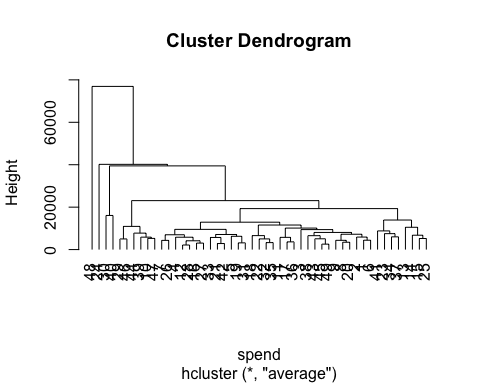


Hierarchical clustering using centroid clustering and squared Euclidean distance

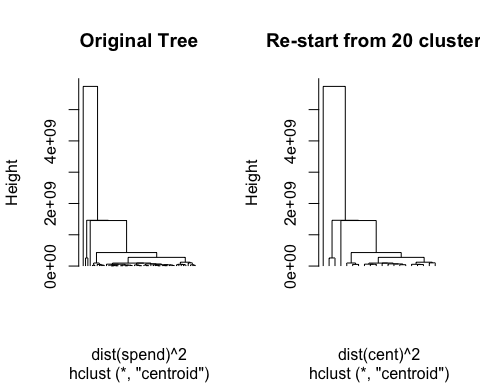
h\_c <- hcluster(spend,link = "ave") # require(amap)  
plot(h\_c)



plot(h\_c, hang = -1)



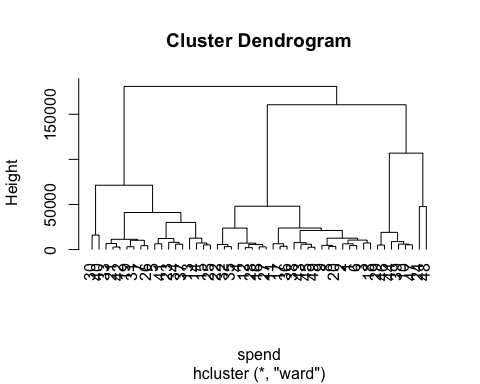
### centroid clustering and squared Euclidean distance  
h\_c<- hclust(dist(spend)^2, "cen")  
  
### Cutting the tree into 20 clusters and reconstruct upper part of the tree from cluster center  
memb <- cutree(h\_c, k = 20)  
cent <- NULL  
for(k in 1:20){  
 cent <- rbind(cent, colMeans(spend[,-1][memb == k, , drop = FALSE]))  
}  
h\_c1 <- hclust(dist(cent)^2, method = "cen", members = table(memb))  
opar <- par(mfrow = c(1, 2))  
plot(h\_c, labels = FALSE, hang = -1, main = "Original Tree")  
plot(h\_c1, labels = FALSE, hang = -1, main = "Re-start from 20 clusters")



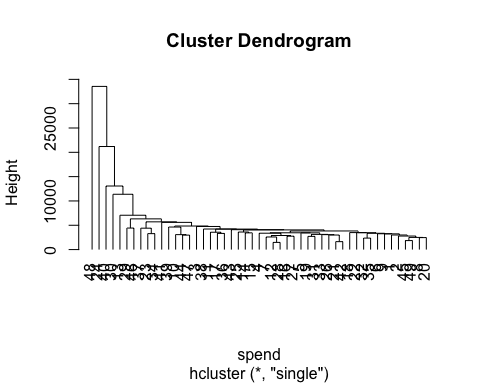
par(opar)

Other combinations

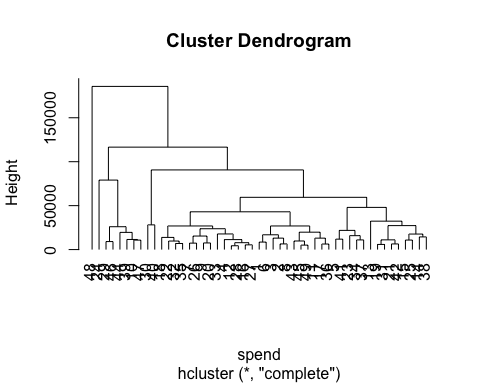
## other combinations   
  
h\_c <- hcluster(spend,method = "euc",link = "ward", nbproc= 1,  
 doubleprecision = TRUE)  
plot(h\_c, hang = -1)



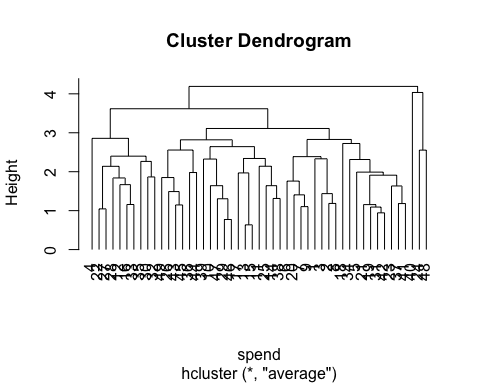
h\_c <- hcluster(spend,method = "max",link = "single", nbproc= 2,   
 doubleprecision = TRUE)  
plot(h\_c, hang = -1)



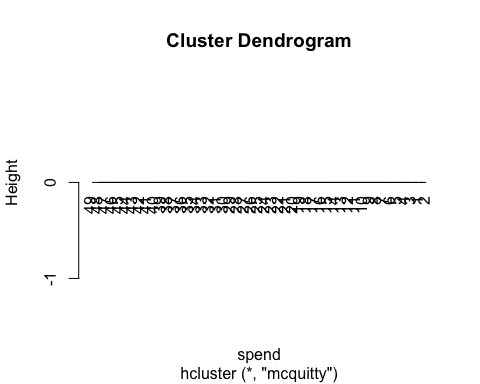
h\_c <- hcluster(spend,method = "man",link = "complete", nbproc= 1,  
 doubleprecision = TRUE)  
plot(h\_c, hang = -1)



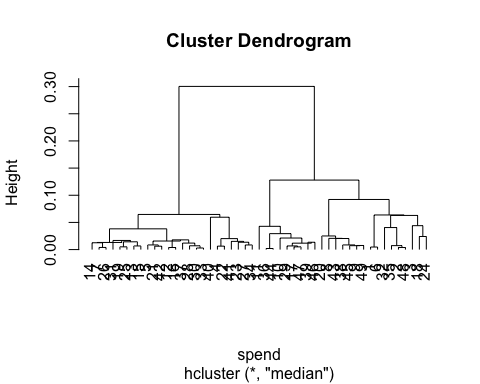
h\_c <- hcluster(spend,method = "can",link = "average", nbproc= 2,  
 doubleprecision = TRUE)  
plot(h\_c, hang = -1)



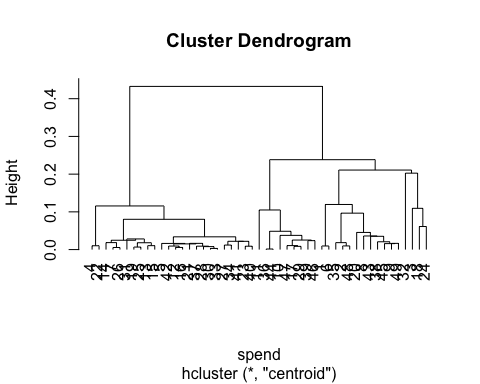
h\_c <- hcluster(spend,method = "bin",link = "mcquitty", nbproc= 1,  
 doubleprecision = FALSE)  
plot(h\_c, hang = -1)



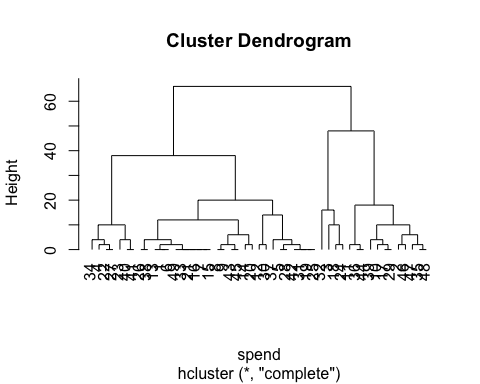
h\_c <- hcluster(spend,method = "pea",link = "median", nbproc= 2,  
 doubleprecision = FALSE)  
plot(h\_c, hang = -1)



h\_c <- hcluster(spend,method = "cor",link = "centroid", nbproc= 1,  
 doubleprecision = FALSE)  
plot(h\_c, hang = -1)



h\_c <- hcluster(spend,method = "spe",link = "complete", nbproc= 2,  
 doubleprecision = FALSE)  
plot(h\_c, hang = -1)



# Readings

An Introduction to Statistical Learning with Applications in R (2013) Authors: Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani Free online via SpringerLink (<http://link.Springer.com/>) <http://link.springer.com/book/10.1007/978-1-4614-7138-7>

* Chapter 14 Unsupervised Learning
* Chapter 13 Prototype Methods and Nearest-Neighbors  
  # Resources

[UC Irvine Machine Learning Repository](https://archive.ics.uci.edu/ml/)

[K-means clustering is not a free lunch](http://varianceexplained.org/r/kmeans-free-lunch/)

# Assingment

* Go to the [UC Irvine Machine Learning Repository](https://archive.ics.uci.edu/ml/) and find a dataset for clustering. Every student MUST use a different dataset so you MUST get approved for which you can going to use. You will use this dataset for this modeule on unsupervised learning and the next on supervised learning.
* Cluster some of your data using k-means, PAM and hierarchical clustering. Answer the following questions:
  + How did you choose a k for k-means?
  + Evaluating model performance.How do the clustering appaoches compare on the same data?
  + Generate and plot confusion matrices for the k-means and PAM. What do they tell you?
  + Generate centroid plots against 1st two discriminant functions for k-means and PAM. What do they tell you?
  + Generate siloutte plots for PAM. What do they tell you?
  + For the hierarchical clustering use all linkage methods (Single Link, Complete Link, Average Link, Centroid and Minimum energy clustering) and generate dendograms. How they compare on the same data?
  + For the hierarchical clustering use both agglomerative and divisive clustering with a linkage method of your choice and generate dendograms. How they compare on the same data?
  + For the hierarchical clustering use centroid clustering and squared Euclidean distance and generate dendograms. How they compare on the same data?