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 are techniques to help choose the number of clusters as well as how to evaluating cluster performance.

Additional packages needed

To run the code 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

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.

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 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 (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/ 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://54.198.163.24/YouTube/MachineLearning/M04/Galton_heights_sons_a
nd_fathers.csv'
```

```
galton <- read.csv(url(data_url))
data_url <-
'http://54.198.163.24/YouTube/MachineLearning/M04/AnnualSpending.csv'
spend <- read.csv(url(data_url))
data_url <-
'http://54.198.163.24/YouTube/MachineLearning/M04/data.worldbank.org.csv'
dwb <- read.csv(url(data_url),header=TRUE,na.strings=c("NA","..", "?"))
```

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          ADO      Europe & Central Asia
## 6         Angola          AGO         Sub-Saharan Africa
##          Income.Group Per.capita.income Literacy Life.expectancy
## 1          Low income          590.2695       NA        60.37446
```

```
## 2 Upper middle income         3965.0168       NA        77.83046
## 3 Upper middle income         4206.0312       NA        74.80810
## 4 Upper middle income               NA       NA              NA
## 5         High income               NA       NA              NA
## 6 Upper middle income         4102.1186 70.77841        52.26688
##   Infant.mortality
## 1             66.3
## 2             12.5
## 3             21.9
## 4               NA
## 5              2.1
## 6             96.0
```

```
nrow(dwb) - nrow(na.omit(dwb)) # There are some rows with at neast one
NA
```

```
## [1] 172
```

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

```
##          Country Life.expectancy Infant.mortality Per.capita.income
## 1    Afghanistan        60.37446             66.3          590.2695
## 2        Albania        77.83046             12.5         3965.0168
## 3        Algeria        74.80810             21.9         4206.0312
## 4 American Samoa              NA               NA               NA
## 5        Andorra              NA              2.1               NA
## 6         Angola        52.26688             96.0         4102.1186
##   Literacy
## 1       NA
## 2       NA
## 3       NA
## 4       NA
## 5       NA
## 6 70.77841
```

```
wb<-na.omit(wb) # Clustering won't work with NA or Inf values
# We remove the NA's here to prevent an NA in a column of no interest
removing a row that has otherwise good data
wb.Country<-wb$Country # We need the country names for formating and
confusion matrices
wb[["Country"]] <- NULL
head(wb)
```

```
##    Life.expectancy Infant.mortality Per.capita.income Literacy
## 6         52.26688             96.0          4102.119 70.77841
## 7         75.93763              5.8         14128.879 98.95000
## 13        70.76322             27.9          5496.345 99.78936
## 16        71.62590             30.7          1211.702 59.72154
```

```
## 25        76.43324               5.1          4197.807 98.26413
## 26        64.42924              34.8          6360.645 87.32057
```

```
nrow(wb) # Make sure there are enough records after removing NA
```

```
## [1] 45
```

Wholesale customers data set

Wholesale customers Data Set

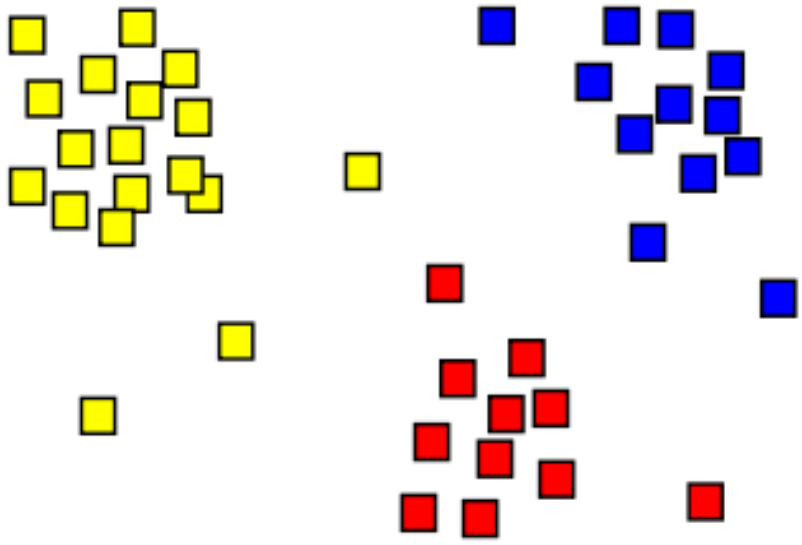
```
head(spend)
```

```
##   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? 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

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.

Similarity measures 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 is a measure of the "closeness" of two vectors that is not a metric space.

Metric Spaces

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

$d: M \times M \to \mathbb{R}$

such that for any $x, y, z \in M$, the following holds:

- $d(x, y) \geq 0 \quad (non - negative)$,

- $d(x,y) = 0, \Leftrightarrow x = y \quad (identity \quad of \quad indiscernibles)$,
- $d(x,y) = d(y,x), \quad (symmetry)$.
- $d(x,z) \leq d(x,y) + d(y,z) \quad (triangle \quad inequality)$

Examples of Metric Spaces

Examples of Metric Spaces are Chebyshev distance, Euclidean distance, Hamming distance, 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 $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ 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:

$$\begin{aligned} \mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) \quad &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned}$$

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) Given vectors A and B, the vector inner product can be defined using the Euclidean dot product formula:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos\theta$$

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. Pearson product-moment correlation coefficient commonly represented by the Greek letter ρ (rho) and is defined as:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}$$

where cov is the covariance and σ_X is the standard deviation of X.

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

$$\operatorname{cov}(X,Y) = E[(X - \mu_X)(Y - \mu_Y)]$$

Clustering Data Structures

- Data matrix

$$M = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,n} \\ .. & .. & .. & \dots & .. \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,n} \end{bmatrix}$$

Distance matrix

$$M = \begin{bmatrix} x_{1,1} & & & & \\ x_{2,1} & 0 & & & \\ x_{3,1} & x_{3,2} & 0 & & \\ .. & .. & .. & \dots & .. \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & 0 \end{bmatrix}$$

- 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 NxN, and B a set of N elements. $A_{i,j}$ is the dissimilarity/similarity between elements B_i and B_j.

```
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:

$$M = \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ .. & .. & .. & \dots & .. \\ d(n,1) & d(n,2) & d(n,3) & \dots & 0 \end{bmatrix}$$

Types of Clustering

Clustering (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 k clusters and n data points $X = x_1, x_2, \dots, x_n$:

Initialize centroids $\mu_1, \mu_1, \dots \mu_k \quad \in \quad \mathbb{R}^n$ (usually randomly)

while (not coverged):

Step A (Assignment step):

Find the closest cluster to every point in $X = x_1, x_2, \dots, x_n$

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 μ_i is the mean of points in X_i.

$$S_i^{(t)} = \{x_p : \| x_p - m_i^{(t)} \|^2 \leq \| x_p - m_j^{(t)} \|^2 \ \ \forall j, 1 \leq j \leq k\}$$

Step B (Update step):

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

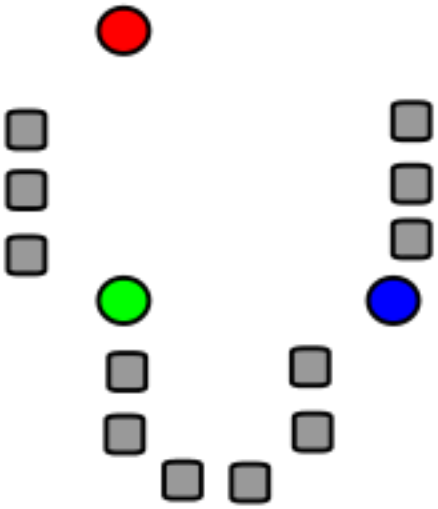
$$m_i^{(t+1)} = \frac{1}{|X_i^{(t)}|} \sum_{x_j \in X_i^{(t)}} x_j$$

The centroid of a finite set of {k} points

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k in \mathbb{R}^n \quad is \quad \mathbf{C} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_k}{k}$$

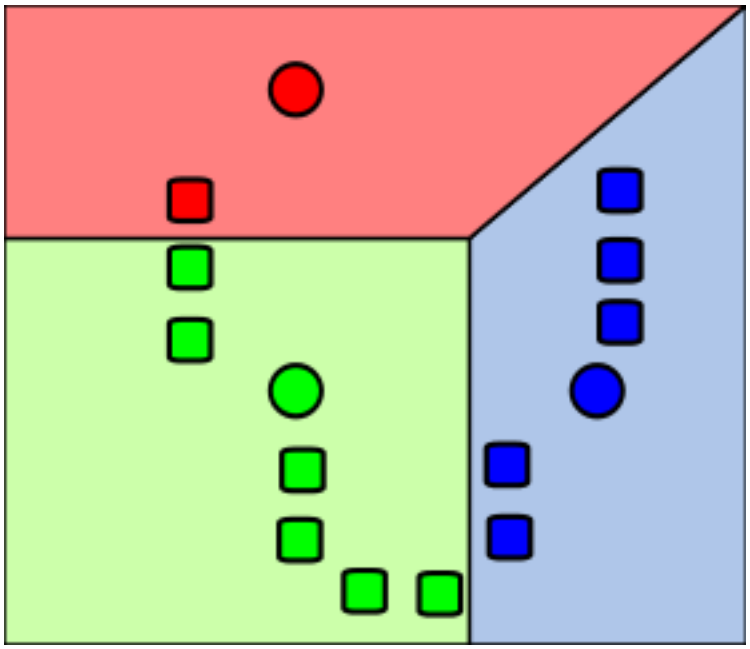
Demonstration of the k-means algorithm

Demonstration of the standard algorithm (from k-means algorithm



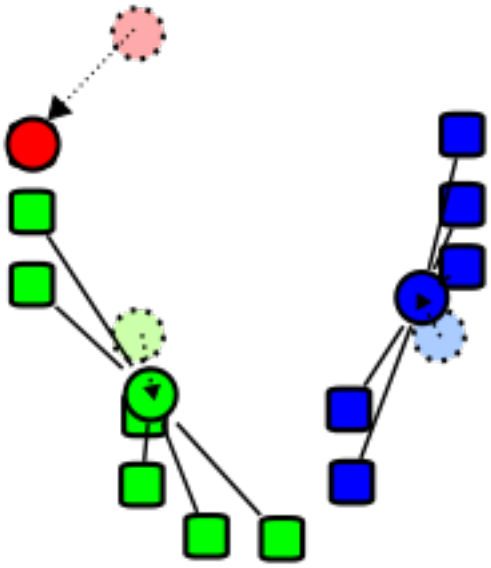
k initial controids

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



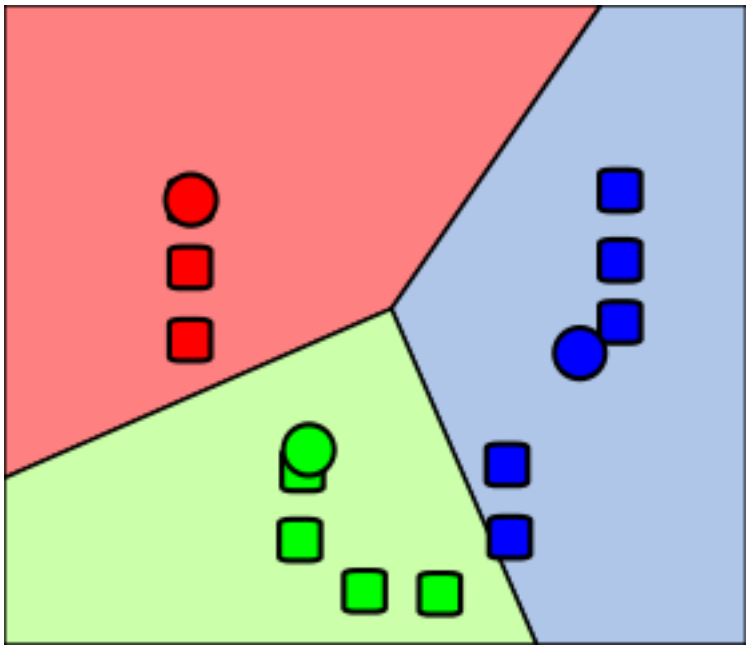
find nearest cluster

2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



new centroid

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



k-means algorithm converges

4. 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

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 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 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
  - $\min \{ d(a,b) : a \in A,\ b \in B \}$
- Complete Link: largest distance between points
  - $\max \{ d(a,b) : a \in A,\ b \in B \}$
- Average Link: average distance between points
  - $\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d\,(a,b)$
- Centroid: distance between centroids
  - $||c_s - c_t||$ where c_s and c_t are the centroids of clusters s and t, 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, 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 **X** of observed data (e.g. assuming it comes for a Gaussian distribution), a set of unobserved latent data or missing values **Z**, 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 **Z** given **X** 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:

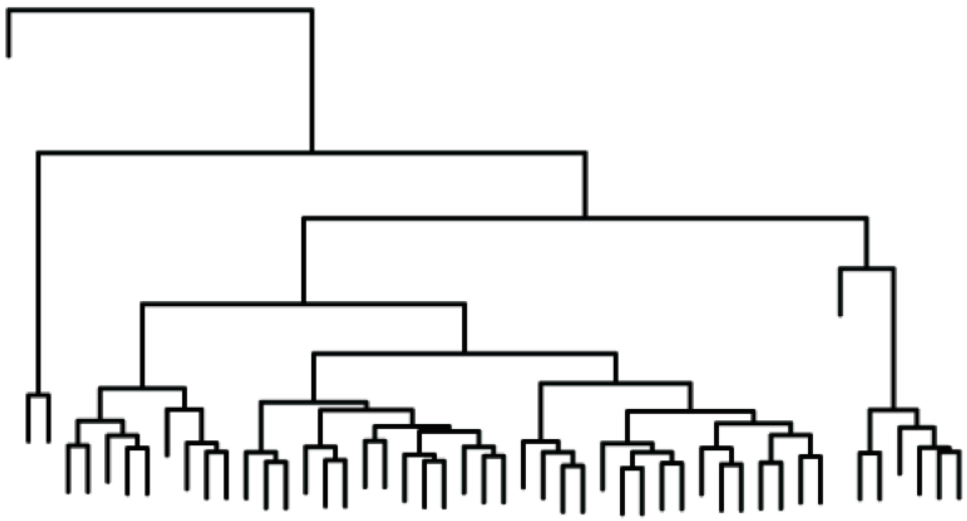
- R package rggobi along with GGobi.org
- R package clusterfly
- other R packages

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

Dendrogram

A 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 i is the union of its children clusters at level $i + 1$



A dendrogram

A dendrogram

Multidimensional scaling (MDS)

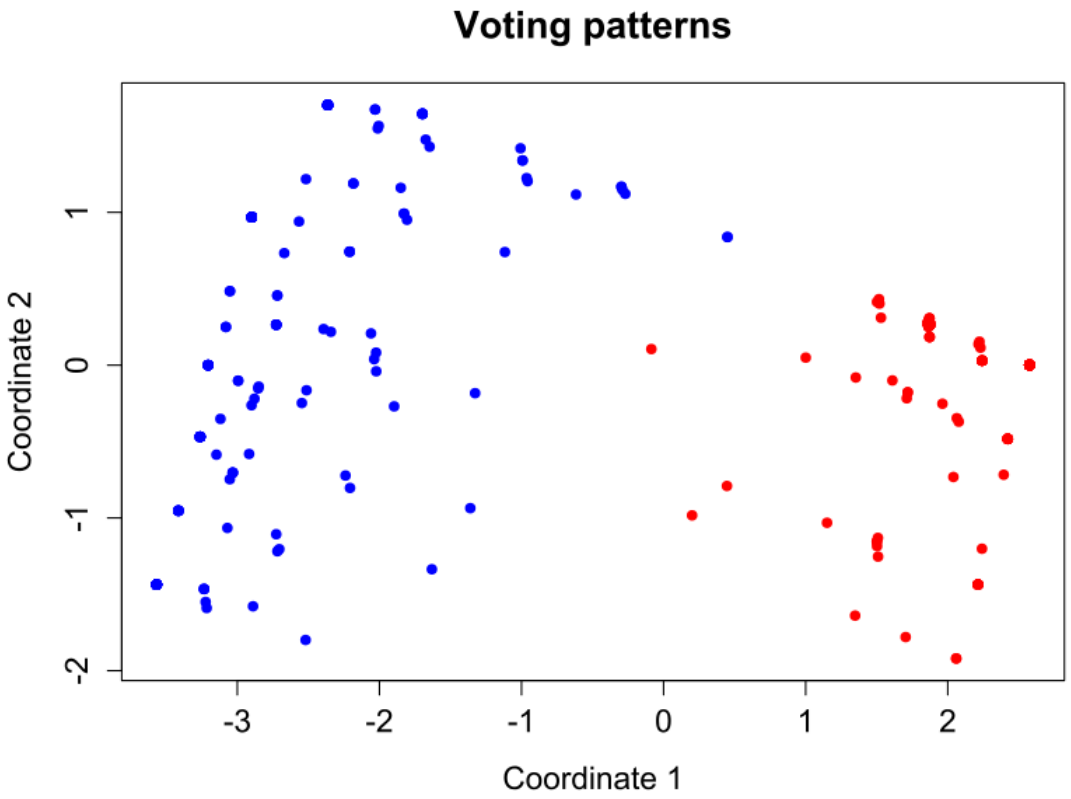
Multidimensional scaling (MDS) is a means of visualizing the level of similarity of individual cases of a high-dimenional dataset.

Given a dataset of I objects (colors, faces, stocks, . . .) on which a distance function is defined, $\delta_{i,j} :=$ distance between $i-th$ and $j-th$ objects. These distances are the entries of the dissimilarity matrix

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \cdots & \delta_{I,I} \end{pmatrix}.$$

The goal of MDS is, given Δ, to find I vectors $x_1, \ldots, x_I \in \mathbb{R}^N$ such that $\| x_i - x_j \| \approx \delta_{i,j}$ for all $i, j \in 1, \ldots, I$, where $\|\cdot\|$ 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 $\mathbb{R}^N$ such that distances are preserved.



multidimensional scaling

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. 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 $fracn(n-1)2$ 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 can be represented by the formula below:

$$DB = \frac{1}{n}\sum_{i=1}^{n} \max_{j \neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

where n is the number of clusters,c_x is the centroid of cluster x, σ_x is the average distance of all elements in cluster x to centroid c_x, and $d(c_i, c_j)$ is the distance between centroids c_i and c_j.

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 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:

$$D = \frac{\min\limits_{1 \le i < j \le n} d(i,j)}{\max\limits_{1 \le k \le n} d'(k)}$$

where $d(i,j)$ represents the distance between clusters i and j, and $d'(k)$ measures the intra-cluster distance of cluster k. The inter-cluster distance $d(i,j)$ 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 c_i and the size of clusterc_i. 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 N. Formally:

$$\text{purity}(\mathbb{C}_i) = \frac{1}{n_i} \sum_k \max_j (n_{i,j}) \quad j \in \mathbb{C}$$

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 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 $p(x, y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively.

Kullback-Leibler divergence

Kullback-Leibler 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:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{i} P\,(i) \ln \frac{P(i)}{Q(i)}$$

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, 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 $\mathbf{x} = (x_1, x_2, \ldots, x_n) and \mathbf{y} = (y_1, y_2, \ldots, y_n)$ are two vectors with all real $x_i, y_i \geq 0$, then their Jaccard similarity coefficient is defined as

$$J(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)},$$

and Jaccard distance

$$d_J(\mathbf{x}, \mathbf{y}) = 1 - J(\mathbf{x}, \mathbf{y}).$$

Rand index

The 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 n elements $S = \{o_1, \dots, o_n\}$ and two partitions of S to compare, $X = \{X_1, \dots, X_r\}$, a partition of S into r subsets, and $Y = \{Y_1, \dots, Y_s\}$, a partition of S into s subsets, define the following:

- a, the number of pairs of elements in S that are in the same set in X and in the same set in Y (i.e. true positive (TP))
- b, the number of pairs of elements in S that are in different sets X and in different sets in Y (i.e false positive (FP))
- c, the number of pairs of elements in S that are in the same set in X and in different sets in Y (false negative (FN))
- d, the number of pairs of elements in S that are in different sets in X and in the same set in Y (i.e. true negative (TN))

The Rand index, R, is:

$$RI = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

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

Rand Index measures between pair decisions.

Number of point pairs	Same Cluster in clustering	Different Clusters in clustering
Same Class	*a* **33**	*c* **22**
Different Classes	*d* **11**	*b* **55**

Green is good (*a*+*b*)

Rand Index

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

$$RI = \frac{a + b}{a + b + c + d}$$

Compare with standard Precision and Recall:

$$Precision = \frac{a}{a + c}$$

$$Recall = \frac{a}{a + d}$$

Compare with a confusion matrix:

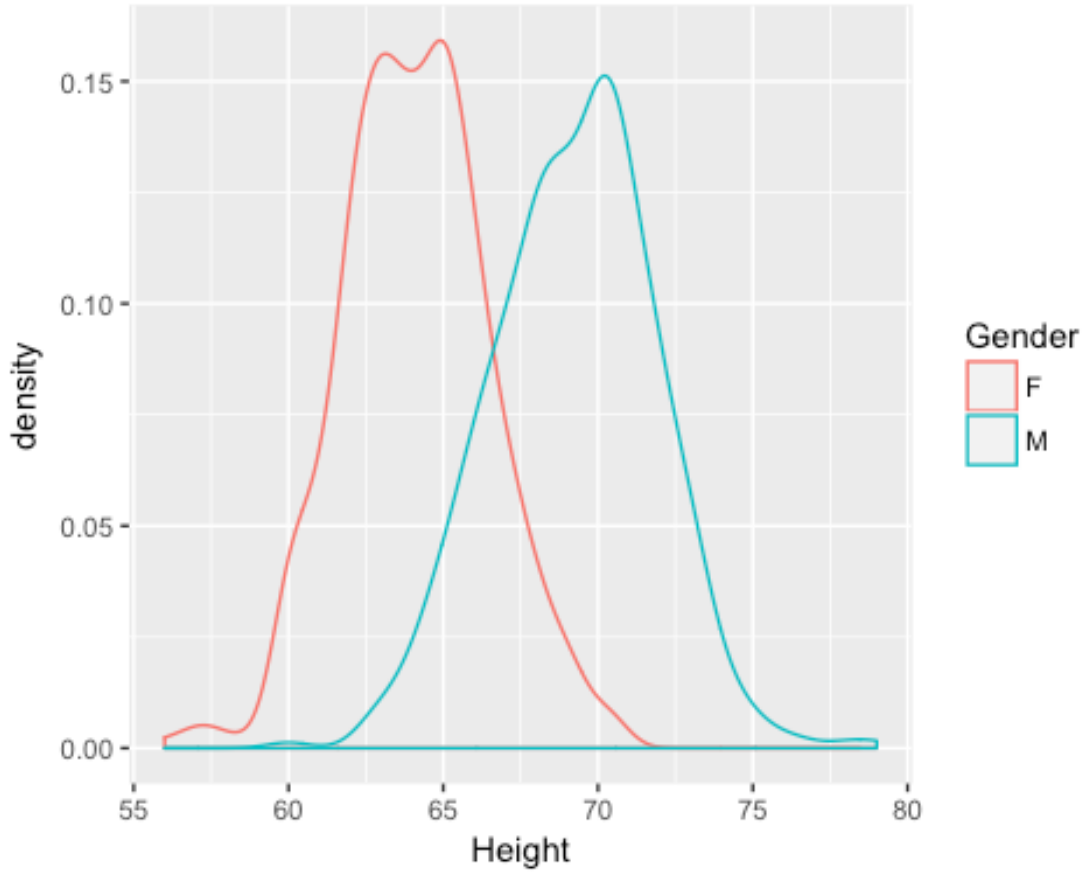
A confusion matrix is a 2x2 table with counts of the following in each of its quadrents.

```
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)
```



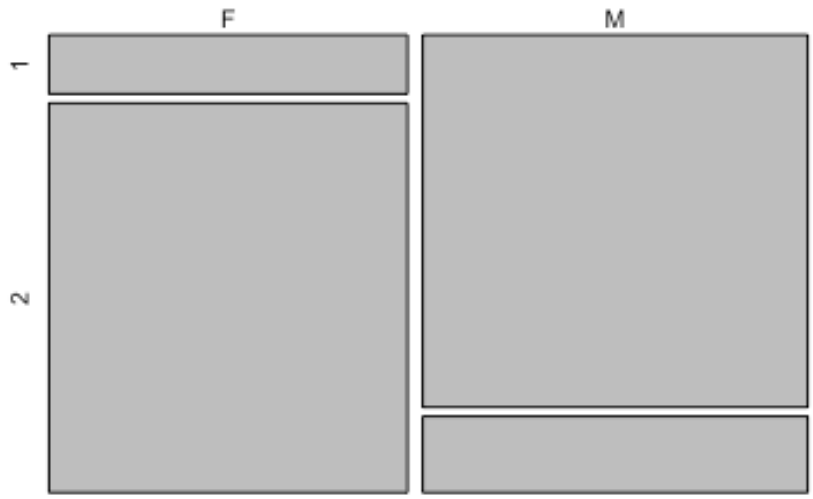
```r
k<-2
galton.2.clust<-kmeans(h.gend[,c("Height")],k)
galton.2.clust
```

```
## 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 1 2 2 2 2 1 2 1 1 2 2 2 2 1 1 1 2 2 2 2 2 2 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 2 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 2 1 1 1 1 1 1 2 2 2 1 2 2 2 2 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 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 2 1 1 1 2 2 2 2 2 2 2 2 1
1 1 1 2
## [421] 2 2 2 2 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 2 1 1 1 2 1 1 2 2 1 1 1 2
2 2 1 1
## [456] 1 1 2 2 2 2 1 1 1 1 2 2 2 2 2 1 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 2 1 1 2 2 2 2 2 2 1 2 2 2 2 1 1 1 1 1 1 1 2 2 1 1
2 2 2 2 1
## [526] 1 2 2 2 1 1 1 2 2 2 1 1 1 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 1 1 2 1
1 1 1 1 1
## [561] 1 1 1 1 2 1 2 2 2 2 1 1 1 2 2 2 2 2 1 1 1 2 1 1 2 2 2 2 2 2 2 1
1 1 2 2 2
## [596] 2 2 2 2 2 2 2 2 1 1 1 1 2 2 1 1 2 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 2 1 1 2 2 2 2 2 2 1 1 2 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 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 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.

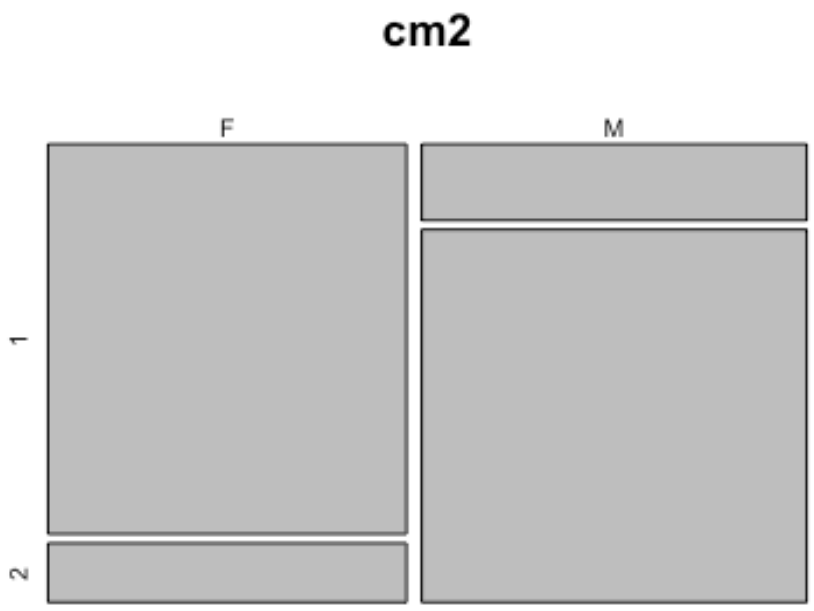
```
cm<-table(h.gend$Gender,galton.2.clust$cluster) # Confustion Matrix
cm

##
##       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 455, 443
##
## Cluster means:
##       [,1]
## 1 63.81692
## 2 69.78420
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 1 1 2 2 2 2 2 1 1 2 2 2 1 1 1 2 2 2 2 2 2 1 2 2 1
1 1 2 2
##  [36] 2 2 2 1 1 1 1 2 2 2 2 1 2 2 2 2 2 2 1 1 1 2 2 2 1 1 1 1 1 1 1
2 2 2 2
##  [71] 2 2 2 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 2 2 1 1 2 2 1 2 2 2
2 1 1 2
## [106] 2 1 1 2 2 2 1 1 1 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
2 1 2 2
## [141] 2 1 1 1 2 2 2 2 2 1 1 2 2 2 2 2 2 1 1 2 1 2 2 2 2 1 2 2 2 1 1 1
1 1 2 2
## [176] 2 2 2 2 2 2 2 2 2 1 1 1 1 2 1 2 2 1 1 1 2 2 2 1 1 1 1 1 1 2 2 2
1 2 2 1
```

```
## [211] 1 1 2 2 2 1 1 2 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 2 2 2 2 2 1 1
1 2 2 2
## [246] 1 1 1 2 2 2 2 1 2 2 1 1 1 1 1 2 2 2 2 2 1 1 1 1 2 2 2 1 1 1 1
1 2 1 1
## [281] 1 1 2 2 2 1 1 1 2 2 2 2 2 1 1 2 2 1 2 2 2 2 2 1 1 1 1 2 2 2 2
1 1 2 2
## [316] 2 1 1 2 2 1 1 1 2 2 2 2 2 2 1 1 1 2 1 1 1 2 2 2 2 2 2 2 1 1 2
2 1 1 1
## [351] 1 1 1 2 2 2 1 1 2 2 2 1 2 2 1 1 2 1 1 1 2 2 2 2 2 1 1 1 2 2 2
1 1 1 1
## [386] 2 2 1 2 2 2 1 1 1 1 1 2 1 1 2 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 2
2 2 2 1
## [421] 1 1 1 2 2 2 2 2 2 2 2 1 2 2 2 1 1 1 2 2 2 2 1 2 2 1 1 2 2 2 1
1 1 2 2
## [456] 2 2 1 1 1 2 2 2 2 1 1 1 1 1 2 2 2 1 1 1 1 2 2 1 1 1 1 1 2 1 1
1 2 2 1
## [491] 2 2 2 2 1 1 1 2 2 1 1 1 1 1 2 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 1
1 1 1 2
## [526] 2 1 1 2 2 2 1 1 2 2 2 1 1 1 1 1 1 2 2 1 2 1 1 1 2 2 1 2 2 2 2
2 2 2 2
## [561] 2 2 2 2 1 2 1 1 1 2 2 2 1 1 1 1 2 2 2 1 2 2 1 1 1 1 1 1 2 2 2
2 1 1 1
## [596] 1 1 1 1 1 1 1 2 2 2 2 1 1 2 2 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1
2 2 2 1
## [631] 2 2 2 1 1 2 2 1 1 1 1 1 1 2 2 1 1 1 1 2 2 2 2 2 1 1 2 2 1 1 2
2 1 1 1
## [666] 1 1 1 1 1 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 2 2 2
1 1 1 1
## [701] 2 2 1 1 1 2 2 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
2 2 1 1
## [736] 2 2 1 2 2 2 2 1 2 2 2 2 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1
2 2 1 1
## [771] 1 1 2 2 1 1 1 1 1 1 2 2 2 1 1 2 1 1 2 2 1 1 1 1 2 2 1 1 1 1 1
2 2 2 1
## [806] 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1
1 1 1 1
## [841] 2 1 2 2 2 2 1 1 2 2 1 1 1 1 2 1 1 1 1 2 2 2 1 2 2 2 2 1 2 2 1
1 1 1 1
## [876] 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 1669.160 1853.249
##  (between_SS / total_SS =  69.4 %)
##
## 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 376  57
##   M  79 386

plot(cm2)
```



Clustering the World Bank data.

```
head(wb)

##    Life.expectancy Infant.mortality Per.capita.income Literacy
## 6         52.26688              96.0          4102.119 70.77841
## 7         75.93763               5.8         14128.879 98.95000
## 13        70.76322              27.9          5496.345 99.78936
## 16        71.62590              30.7          1211.702 59.72154
## 25        76.43324               5.1          4197.807 98.26413
## 26        64.42924              34.8          6360.645 87.32057
```

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

```
## K-means clustering with 2 clusters of sizes 5, 40
##
## Cluster means:
##   Life.expectancy Infant.mortality Per.capita.income Literacy
## 1        80.31929             4.52         42443.843 97.35069
## 2        70.23347            26.62          6657.215 87.17252
##
## Clustering vector:
##   6   7  13  16  25  26  27  39  57  58  59  60  61  72  76  80  82
85
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2
##  87  92  96  97 106 113 118 128 130 136 156 159 166 167 170 171 174
179
##   2   2   1   2   1   2   2   2   2   2   2   1   2   2   2   1   2
1
## 185 191 197 199 200 204 208 209 214
##   2   2   2   2   2   2   2   2   2
##
## Within cluster sum of squares by cluster:
## [1] 1763237061 1352488790
##  (between_SS / total_SS =  64.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

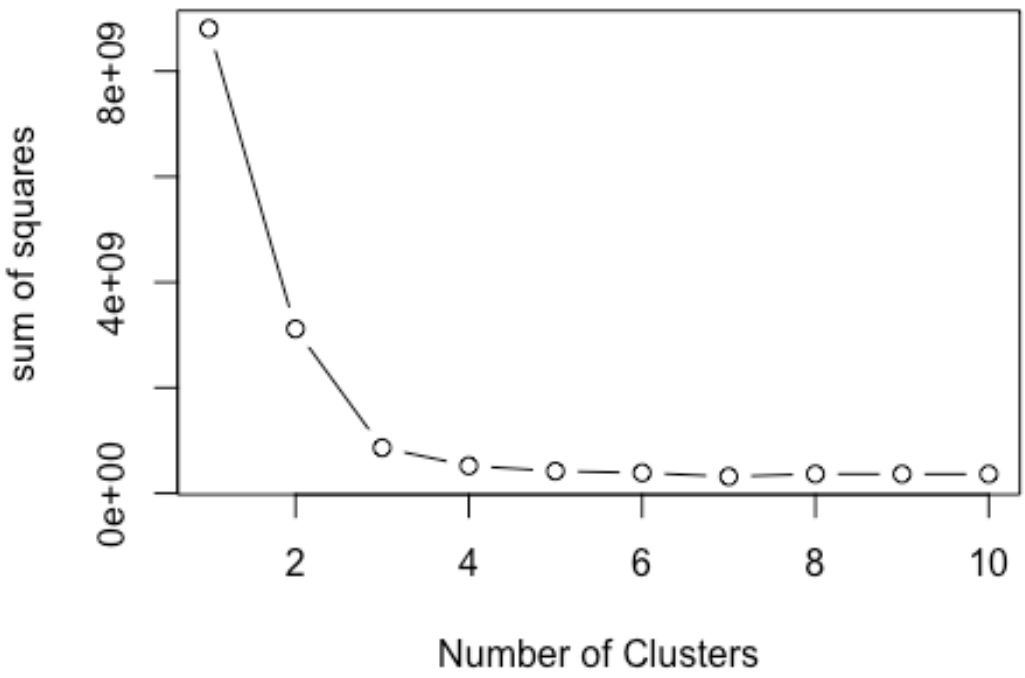
```r
k<-3
wb.3.clust<- kmeans(wb,k)
wb.3.clust
```

```
## K-means clustering with 3 clusters of sizes 2, 11, 32
##
## Cluster means:
##   Life.expectancy Infant.mortality Per.capita.income Literacy
## 1        80.62157         4.450000          63777.97 97.00900
## 2        77.54424         6.763636          19890.37 98.12095
## 3        68.64705        31.378125           4129.93 84.38456
##
## Clustering vector:
##   6   7  13  16  25  26  27  39  57  58  59  60  61  72  76  80  82
85
##   3   2   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3
3
```

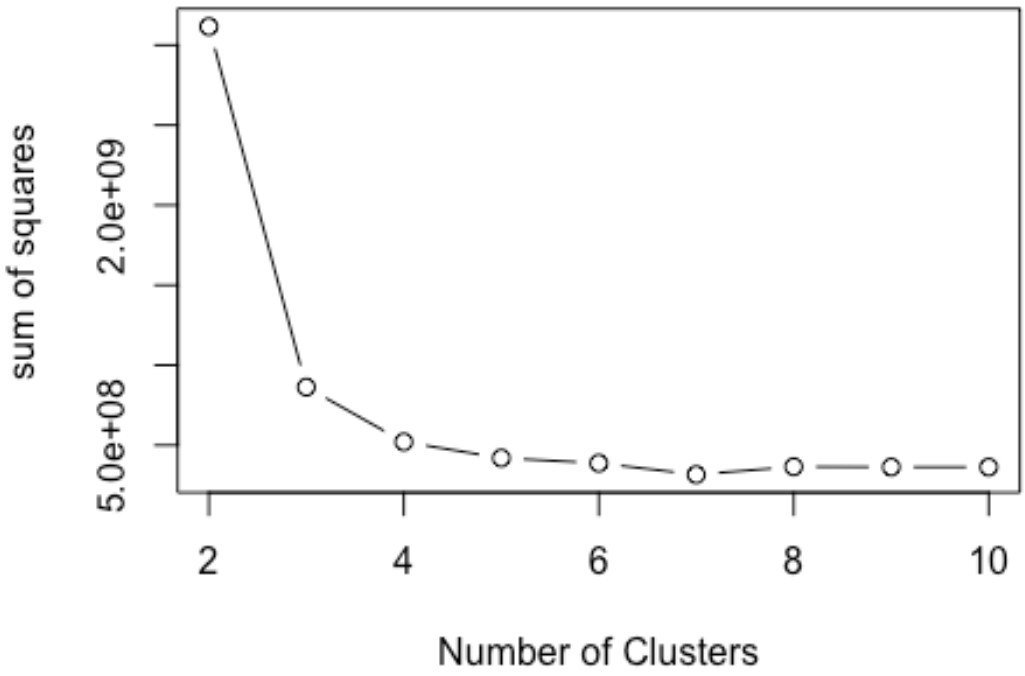
```
##  87  92  96  97 106 113 118 128 130 136 156 159 166 167 170 171 174
179
##   2   3   2   3   2   3   3   3   3   3   2   1   2   3   3   1   2
2
## 185 191 197 199 200 204 208 209 214
##   3   3   2   3   3   3   2   3   3
##
## Within cluster sum of squares by cluster:
## [1] 237150514 386675339 239076979
##  (between_SS / total_SS =  90.2 %)
##
## 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.

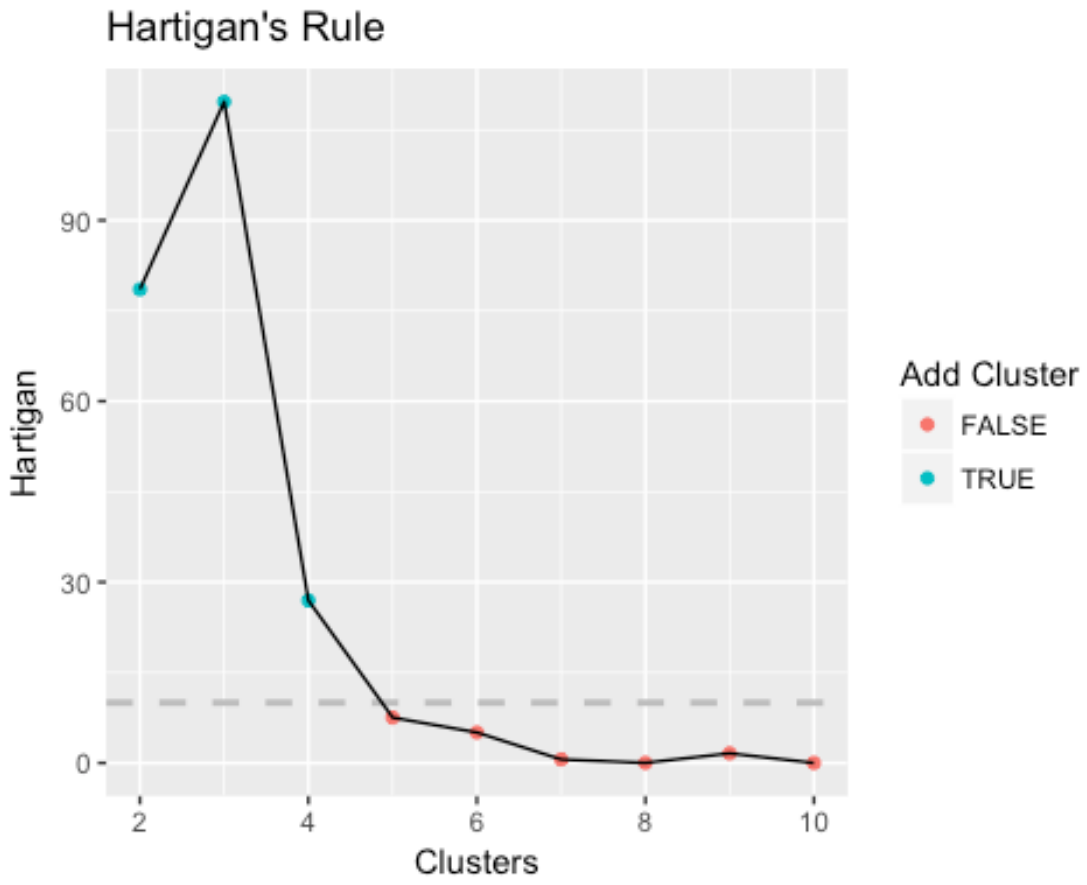
```
# Determining number of clusters
sos <- (nrow(wb)-1)*sum(apply(wb,2,var))
for (i in 2:10) sos[i] <- sum(kmeans(wb, centers=i)$withinss)
plot(1:10, sos, type="b", xlab="Number of Clusters", ylab="sum of
squares")
```




```
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?

```
head(wb)
```

```
##    Life.expectancy Infant.mortality Per.capita.income Literacy
## 6         52.26688             96.0          4102.119 70.77841
## 7         75.93763              5.8         14128.879 98.95000
## 13        70.76322             27.9          5496.345 99.78936
## 16        71.62590             30.7          1211.702 59.72154
## 25        76.43324              5.1          4197.807 98.26413
## 26        64.42924             34.8          6360.645 87.32057
```

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

```
## K-means clustering with 6 clusters of sizes 6, 15, 4, 2, 5, 13
##
## Cluster means:
##   Life.expectancy Infant.mortality Per.capita.income Literacy
## 1        77.60843          7.75000         24383.696 97.62204
## 2        70.74951         25.16000          4968.196 88.90804
## 3        70.98357         26.42500          9449.684 93.86932
## 4        80.62157          4.45000         63777.971 97.00900
```

```
## 5        77.46720          5.58000         14498.383 98.71964
## 6        65.50221         40.07692          1525.854 76.24678
##
## Clustering vector:
##   6   7  13  16  25  26  27  39  57  58  59  60  61  72  76  80  82
85
##   2   5   2   6   2   2   3   6   2   2   2   2   3   2   5   2   6
6
##  87  92  96  97 106 113 118 128 130 136 156 159 166 167 170 171 174
179
##   5   2   1   2   1   2   2   3   6   6   5   4   1   6   6   4   1
1
## 185 191 197 199 200 204 208 209 214
##   6   6   1   3   2   6   5   6   6
##
## Within cluster sum of squares by cluster:
## [1]  97332955  16265503   3921650 237150514  22834895   6991191
##  (between_SS / total_SS =  95.6 %)
##
## 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 7, 13, 15, 3, 2, 5
##
## Cluster means:
##   Life.expectancy Infant.mortality Per.capita.income Literacy
## 1        73.28554        17.328571         10954.456 96.17687
## 2        65.50221        40.076923          1525.854 76.24678
## 3        70.74951        25.160000          4968.196 88.90804
## 4        80.11777         4.566667         28221.092 97.57849
## 5        80.62157         4.450000         63777.971 97.00900
## 6        76.71376         9.020000         19049.671 97.76685
##
## Clustering vector:
##   6   7  13  16  25  26  27  39  57  58  59  60  61  72  76  80  82
85
##   3   1   3   2   3   3   1   2   3   3   3   3   1   3   6   3   2
2
##  87  92  96  97 106 113 118 128 130 136 156 159 166 167 170 171 174
179
##   1   3   4   3   4   3   3   1   2   2   1   5   6   2   2   5   6
4
## 185 191 197 199 200 204 208 209 214
```

```
##   2   2   6   1   3   2   6   2   2
##
## Within cluster sum of squares by cluster:
## [1]  27130607   6991191  16265503   8936570 237150514  19871802
##  (between_SS / total_SS =  96.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
length(wb.Country)
```

```
## [1] 45
```

```
length(wb.6.clust.33$cluster)
```

```
## [1] 45
```

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

```
##
## wb.Country                       1 2 3 4 5 6
##   Afghanistan                    0 0 0 0 0 0
##   Albania                        0 0 0 0 0 0
##   Algeria                        0 0 0 0 0 0
##   American Samoa                 0 0 0 0 0 0
##   Andorra                        0 0 0 0 0 0
##   Angola                         0 0 1 0 0 0
##   Antigua and Barbuda            1 0 0 0 0 0
##   Argentina                      0 0 0 0 0 0
##   Armenia                        0 0 0 0 0 0
##   Aruba                          0 0 0 0 0 0
##   Australia                      0 0 0 0 0 0
##   Austria                        0 0 0 0 0 0
##   Azerbaijan                     0 0 1 0 0 0
##   Bahamas, The                   0 0 0 0 0 0
##   Bahrain                        0 0 0 0 0 0
##   Bangladesh                     0 1 0 0 0 0
##   Barbados                       0 0 0 0 0 0
##   Belarus                        0 0 0 0 0 0
##   Belgium                        0 0 0 0 0 0
##   Belize                         0 0 0 0 0 0
##   Benin                          0 0 0 0 0 0
##   Bermuda                        0 0 0 0 0 0
##   Bhutan                         0 0 0 0 0 0
##   Bolivia                        0 0 0 0 0 0
##   Bosnia and Herzegovina         0 0 1 0 0 0
##   Botswana                       0 0 1 0 0 0
```

```
##   Brazil                          1 0 0 0 0 0
##   British Virgin Islands          0 0 0 0 0 0
##   Brunei Darussalam               0 0 0 0 0 0
##   Bulgaria                        0 0 0 0 0 0
##   Burkina Faso                    0 0 0 0 0 0
##   Burundi                         0 0 0 0 0 0
##   Cabo Verde                      0 0 0 0 0 0
##   Cambodia                        0 0 0 0 0 0
##   Cameroon                        0 0 0 0 0 0
##   Canada                          0 0 0 0 0 0
##   Cayman Islands                  0 0 0 0 0 0
##   Central African Republic        0 0 0 0 0 0
##   Chad                            0 1 0 0 0 0
##   Channel Islands                 0 0 0 0 0 0
##   Chile                           0 0 0 0 0 0
##   China                           0 0 0 0 0 0
##   Colombia                        0 0 0 0 0 0
##   Comoros                         0 0 0 0 0 0
##   Congo, Dem. Rep.                0 0 0 0 0 0
##   Congo, Rep.                     0 0 0 0 0 0
##   Costa Rica                      0 0 0 0 0 0
##   Cote d'Ivoire                   0 0 0 0 0 0
##   Croatia                         0 0 0 0 0 0
##   Cuba                            0 0 0 0 0 0
##   Curacao                         0 0 0 0 0 0
##   Cyprus                          0 0 0 0 0 0
##   Czech Republic                  0 0 0 0 0 0
##   Denmark                         0 0 0 0 0 0
##   Djibouti                        0 0 0 0 0 0
##   Dominica                        0 0 0 0 0 0
##   Dominican Republic              0 0 1 0 0 0
##   Ecuador                         0 0 1 0 0 0
##   Egypt, Arab Rep.                0 0 1 0 0 0
##   El Salvador                     0 0 1 0 0 0
##   Equatorial Guinea               1 0 0 0 0 0
##   Eritrea                         0 0 0 0 0 0
##   Estonia                         0 0 0 0 0 0
##   Ethiopia                        0 0 0 0 0 0
##   Faroe Islands                   0 0 0 0 0 0
##   Fiji                            0 0 0 0 0 0
##   Finland                         0 0 0 0 0 0
##   France                          0 0 0 0 0 0
##   French Polynesia                0 0 0 0 0 0
##   Gabon                           0 0 0 0 0 0
##   Gambia, The                     0 0 0 0 0 0
##   Georgia                         0 0 1 0 0 0
##   Germany                         0 0 0 0 0 0
##   Ghana                           0 0 0 0 0 0
##   Gibraltar                       0 0 0 0 0 0
##   Greece                          0 0 0 0 0 1
```

```
##   Greenland                        0 0 0 0 0 0
##   Grenada                          0 0 0 0 0 0
##   Guam                             0 0 0 0 0 0
##   Guatemala                        0 0 1 0 0 0
##   Guinea                           0 0 0 0 0 0
##   Guinea-Bissau                    0 1 0 0 0 0
##   Guyana                           0 0 0 0 0 0
##   Haiti                            0 0 0 0 0 0
##   Honduras                         0 1 0 0 0 0
##   Hong Kong SAR, China             0 0 0 0 0 0
##   Hungary                          1 0 0 0 0 0
##   Iceland                          0 0 0 0 0 0
##   India                            0 0 0 0 0 0
##   Indonesia                        0 0 0 0 0 0
##   Iran, Islamic Rep.               0 0 0 0 0 0
##   Iraq                             0 0 1 0 0 0
##   Ireland                          0 0 0 0 0 0
##   Isle of Man                      0 0 0 0 0 0
##   Israel                           0 0 0 0 0 0
##   Italy                            0 0 0 1 0 0
##   Jamaica                          0 0 1 0 0 0
##   Japan                            0 0 0 0 0 0
##   Jordan                           0 0 0 0 0 0
##   Kazakhstan                       0 0 0 0 0 0
##   Kenya                            0 0 0 0 0 0
##   Kiribati                         0 0 0 0 0 0
##   Korea, Dem. People\x89۪s Rep.   0 0 0 0 0 0
##   Korea, Rep.                      0 0 0 0 0 0
##   Kosovo                           0 0 0 0 0 0
##   Kuwait                           0 0 0 1 0 0
##   Kyrgyz Republic                  0 0 0 0 0 0
##   Lao PDR                          0 0 0 0 0 0
##   Latvia                           0 0 0 0 0 0
##   Lebanon                          0 0 0 0 0 0
##   Lesotho                          0 0 0 0 0 0
##   Liberia                          0 0 0 0 0 0
##   Libya                            0 0 1 0 0 0
##   Liechtenstein                    0 0 0 0 0 0
##   Lithuania                        0 0 0 0 0 0
##   Luxembourg                       0 0 0 0 0 0
##   Macao SAR, China                 0 0 0 0 0 0
##   Macedonia, FYR                   0 0 1 0 0 0
##   Madagascar                       0 0 0 0 0 0
##   Malawi                           0 0 0 0 0 0
##   Malaysia                         0 0 0 0 0 0
##   Maldives                         0 0 0 0 0 0
##   Mali                             0 0 0 0 0 0
##   Malta                            0 0 0 0 0 0
##   Marshall Islands                 0 0 0 0 0 0
##   Mauritania                       0 0 0 0 0 0
```

```
##   Mauritius                      0 0 0 0 0 0
##   Mexico                         1 0 0 0 0 0
##   Micronesia, Fed. Sts.          0 0 0 0 0 0
##   Moldova                        0 1 0 0 0 0
##   Monaco                         0 0 0 0 0 0
##   Mongolia                       0 0 0 0 0 0
##   Montenegro                     0 0 0 0 0 0
##   Morocco                        0 0 0 0 0 0
##   Mozambique                     0 0 0 0 0 0
##   Myanmar                        0 1 0 0 0 0
##   Namibia                        0 0 0 0 0 0
##   Nauru                          0 0 0 0 0 0
##   Nepal                          0 0 0 0 0 0
##   Netherlands                    0 0 0 0 0 0
##   New Caledonia                  0 0 0 0 0 0
##   New Zealand                    0 0 0 0 0 0
##   Nicaragua                      0 0 0 0 0 0
##   Niger                          0 0 0 0 0 0
##   Nigeria                        0 0 0 0 0 0
##   Northern Mariana Islands       0 0 0 0 0 0
##   Norway                         0 0 0 0 0 0
##   Oman                           0 0 0 0 0 0
##   Pakistan                       0 0 0 0 0 0
##   Palau                          0 0 0 0 0 0
##   Panama                         0 0 0 0 0 0
##   Papua New Guinea               0 0 0 0 0 0
##   Paraguay                       0 0 0 0 0 0
##   Peru                           0 0 0 0 0 0
##   Philippines                    0 0 0 0 0 0
##   Poland                         1 0 0 0 0 0
##   Portugal                       0 0 0 0 0 0
##   Puerto Rico                    0 0 0 0 0 0
##   Qatar                          0 0 0 0 1 0
##   Romania                        0 0 0 0 0 0
##   Russian Federation             0 0 0 0 0 0
##   Rwanda                         0 0 0 0 0 0
##   Samoa                          0 0 0 0 0 0
##   San Marino                     0 0 0 0 0 0
##   Sao Tome and Principe          0 0 0 0 0 0
##   Saudi Arabia                   0 0 0 0 0 1
##   Senegal                        0 1 0 0 0 0
##   Serbia                         0 0 0 0 0 0
##   Seychelles                     0 0 0 0 0 0
##   Sierra Leone                   0 1 0 0 0 0
##   Singapore                      0 0 0 0 1 0
##   Sint Maarten (Dutch part)      0 0 0 0 0 0
##   Slovak Republic                0 0 0 0 0 0
##   Slovenia                       0 0 0 0 0 1
##   Solomon Islands                0 0 0 0 0 0
##   Somalia                        0 0 0 0 0 0
```

```
##   South Africa                    0 0 0 0 0 0
##   South Sudan                     0 0 0 0 0 0
##   Spain                           0 0 0 1 0 0
##   Sri Lanka                       0 0 0 0 0 0
##   St. Kitts and Nevis             0 0 0 0 0 0
##   St. Lucia                       0 0 0 0 0 0
##   St. Martin (French part)        0 0 0 0 0 0
##   St. Vincent and the Grenadines 0 0 0 0 0 0
##   Sudan                           0 1 0 0 0 0
##   Suriname                        0 0 0 0 0 0
##   Swaziland                       0 0 0 0 0 0
##   Sweden                          0 0 0 0 0 0
##   Switzerland                     0 0 0 0 0 0
##   Syrian Arab Republic            0 0 0 0 0 0
##   Tajikistan                      0 1 0 0 0 0
##   Tanzania                        0 0 0 0 0 0
##   Thailand                        0 0 0 0 0 0
##   Timor-Leste                     0 0 0 0 0 0
##   Togo                            0 0 0 0 0 0
##   Tonga                           0 0 0 0 0 0
##   Trinidad and Tobago             0 0 0 0 0 1
##   Tunisia                         0 0 0 0 0 0
##   Turkey                          1 0 0 0 0 0
##   Turkmenistan                    0 0 1 0 0 0
##   Turks and Caicos Islands        0 0 0 0 0 0
##   Tuvalu                          0 0 0 0 0 0
##   Uganda                          0 0 0 0 0 0
##   Ukraine                         0 1 0 0 0 0
##   United Arab Emirates            0 0 0 0 0 0
##   United Kingdom                  0 0 0 0 0 0
##   United States                   0 0 0 0 0 0
##   Uruguay                         0 0 0 0 0 1
##   Uzbekistan                      0 1 0 0 0 0
##   Vanuatu                         0 0 0 0 0 0
##   Venezuela, RB                   0 0 0 0 0 0
##   Vietnam                         0 0 0 0 0 0
##   Virgin Islands (U.S.)           0 0 0 0 0 0
##   West Bank and Gaza              0 1 0 0 0 0
##   Yemen, Rep.                     0 0 0 0 0 0
##   Zambia                          0 0 0 0 0 0
##   Zimbabwe                        0 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]  6 15  4  2  5 13
```

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

```
##   Life.expectancy Infant.mortality Per.capita.income Literacy
## 1        77.60843          7.75000         24383.696 97.62204
## 2        70.74951         25.16000          4968.196 88.90804
## 3        70.98357         26.42500          9449.684 93.86932
## 4        80.62157          4.45000         63777.971 97.00900
## 5        77.46720          5.58000         14498.383 98.71964
## 6        65.50221         40.07692          1525.854 76.24678
```

```
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         24383.696
## 2                  2          4968.196
## 3                  3          9449.684
## 4                  4         63777.971
## 5                  5         14498.383
## 6                  6          1525.854
```

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

```
##   wb.6.clust$cluster Literacy
## 1                  1 97.62204
## 2                  2 88.90804
## 3                  3 93.86933
## 4                  4 97.00900
## 5                  5 98.71964
## 6                  6 76.24678
```

```
# 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          7.75000
## 2                  2         25.16000
## 3                  3         26.42500
## 4                  4          4.45000
## 5                  5          5.58000
## 6                  6         40.07692
```

```
# 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        77.60843
## 2                  2        70.74951
## 3                  3        70.98357
## 4                  4        80.62157
## 5                  5        77.46720
## 6                  6        65.50221
```

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

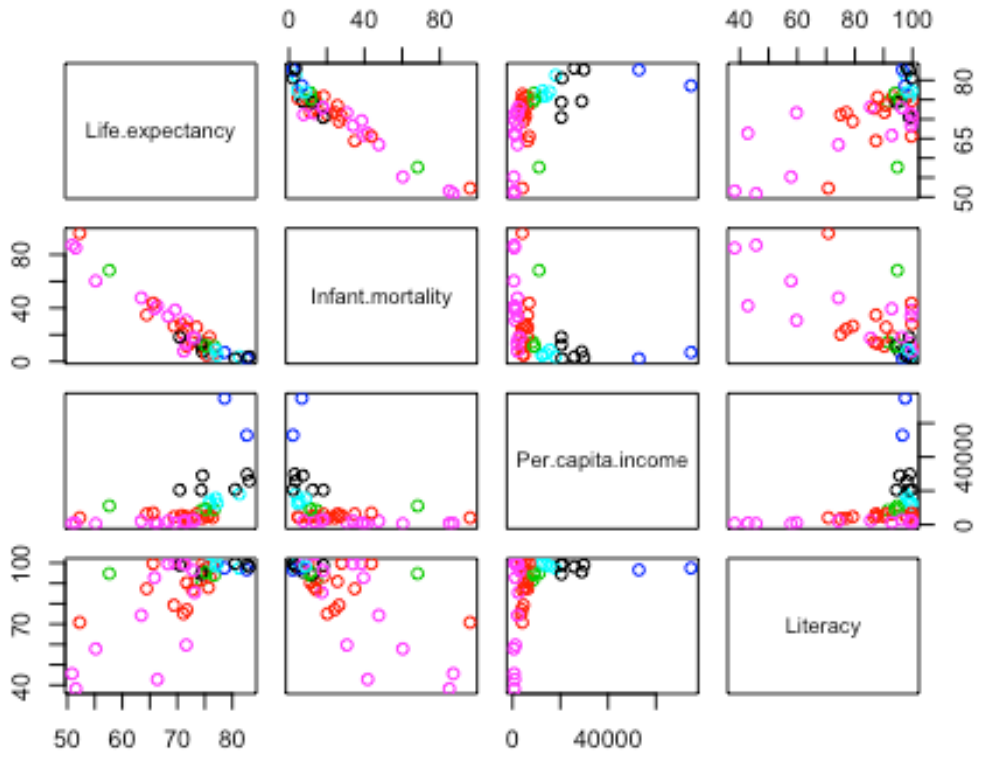
```
##
##               1 2 3 4 5 6
##   573.0286189 0 0 0 0 0 1
##   693.4080457 0 0 0 0 0 1
##   775.6950905 0 0 0 0 0 1
##   910.788688  0 0 0 0 0 1
##   925.9118877 0 0 0 0 0 1
##   1203.505387 0 0 0 0 0 1
##   1211.701531 0 0 0 0 0 1
##   1843.242802 0 0 0 0 0 1
##   2089.400212 0 0 0 0 0 1
##   2114.954716 0 0 0 0 0 1
##   2132.072442 0 0 0 0 0 1
##   2495.590498 0 0 0 0 0 1
##   2866.800101 0 0 0 0 0 1
##   3614.746766 0 1 0 0 0 0
##   3795.973308 0 1 0 0 0 0
##   3903.490842 0 1 0 0 0 0
##   4102.11859  0 1 0 0 0 0
##   4197.807304 0 1 0 0 0 0
##   4219.35033  0 1 0 0 0 0
##   4629.076634 0 1 0 0 0 0
##   4643.305762 0 1 0 0 0 0
##   4852.657847 0 1 0 0 0 0
##   5137.91553  0 1 0 0 0 0
##   5496.34464  0 1 0 0 0 0
##   6248.110873 0 1 0 0 0 0
##   6360.644776 0 1 0 0 0 0
```

```
##   6373.553553 0 1 0 0 0 0
##   6947.840023 0 1 0 0 0 0
##   8538.589975 0 0 1 0 0 0
##   9009.261163 0 0 1 0 0 0
##   9130.026065 0 0 1 0 0 0
##   11120.85799 0 0 1 0 0 0
##   12259.11503 0 0 0 0 1 0
##   12494.46619 0 0 0 0 1 0
##   14128.87855 0 0 0 0 1 0
##   15573.90092 0 0 0 0 1 0
##   18035.55432 0 0 0 0 1 0
##   20444.07859 1 0 0 0 0 0
##   20481.74532 1 0 0 0 0 0
##   20713.07475 1 0 0 0 0 0
##   25831.58231 1 0 0 0 0 0
##   28984.64339 1 0 0 0 0 0
##   29847.04979 1 0 0 0 0 0
##   52888.74467 0 0 0 1 0 0
##   74667.19707 0 0 0 1 0 0

print(wb.6.clust)

## K-means clustering with 6 clusters of sizes 6, 15, 4, 2, 5, 13
##
## Cluster means:
##   Life.expectancy Infant.mortality Per.capita.income Literacy
## 1        77.60843          7.75000         24383.696 97.62204
## 2        70.74951         25.16000          4968.196 88.90804
## 3        70.98357         26.42500          9449.684 93.86932
## 4        80.62157          4.45000         63777.971 97.00900
## 5        77.46720          5.58000         14498.383 98.71964
## 6        65.50221         40.07692          1525.854 76.24678
##
## Clustering vector:
##   6   7  13  16  25  26  27  39  57  58  59  60  61  72  76  80  82
85
##   2   5   2   6   2   2   3   6   2   2   2   2   3   2   5   2   6
6
##  87  92  96  97 106 113 118 128 130 136 156 159 166 167 170 171 174
179
##   5   2   1   2   1   2   2   3   6   6   5   4   1   6   6   4   1
1
## 185 191 197 199 200 204 208 209 214
##   6   6   1   3   2   6   5   6   6
##
## Within cluster sum of squares by cluster:
## [1]  97332955  16265503   3921650 237150514  22834895   6991191
##  (between_SS / total_SS =  95.6 %)
##
## Available components:
```

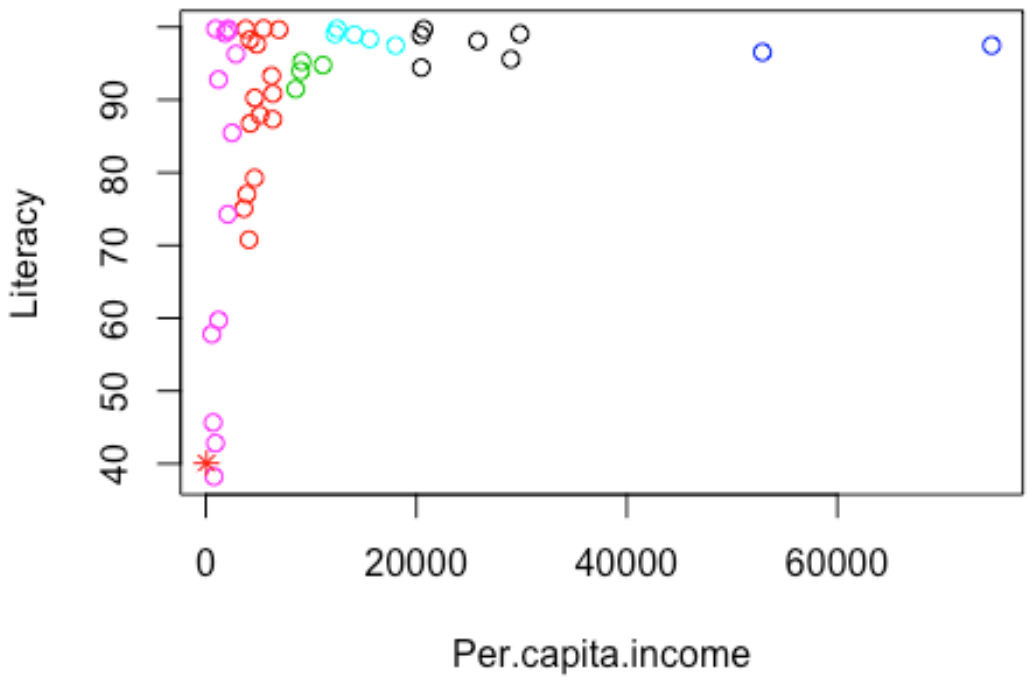
```
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Plotting Clusters

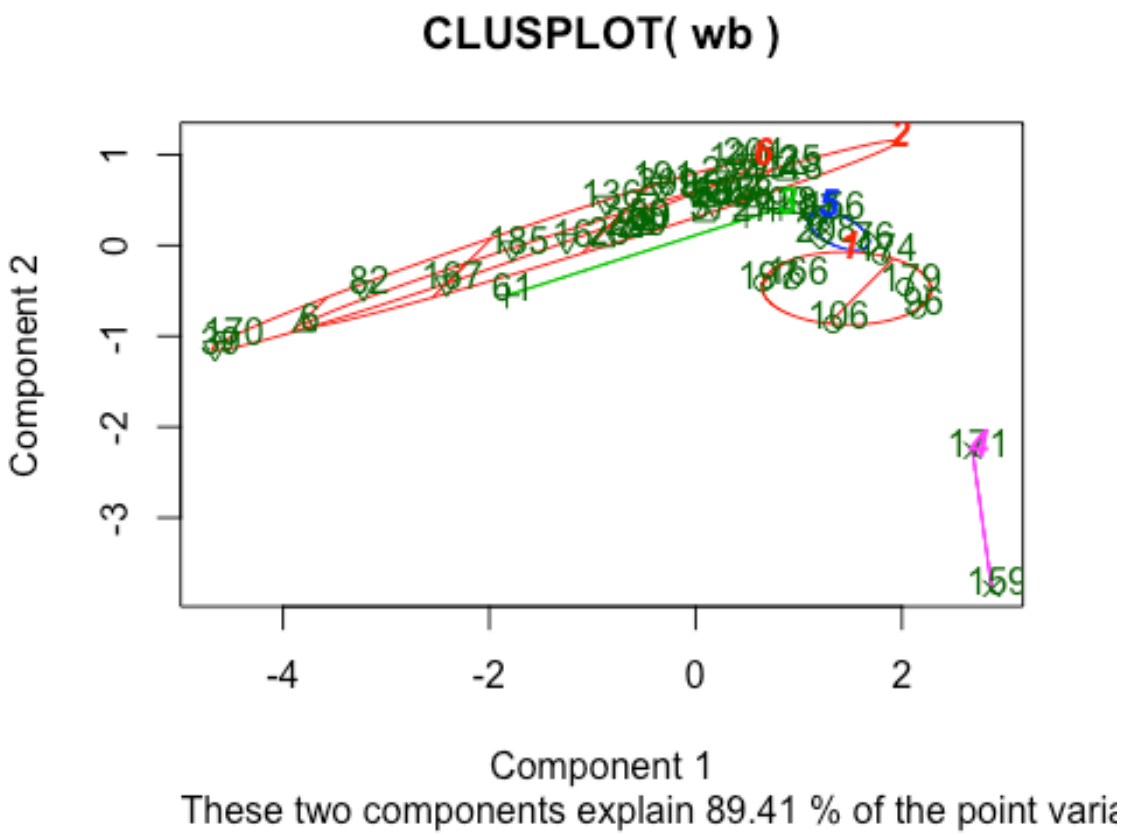
```
plot(wb,col=wb.6.clust$cluster)      # Plot Clusters
```




```
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)
```

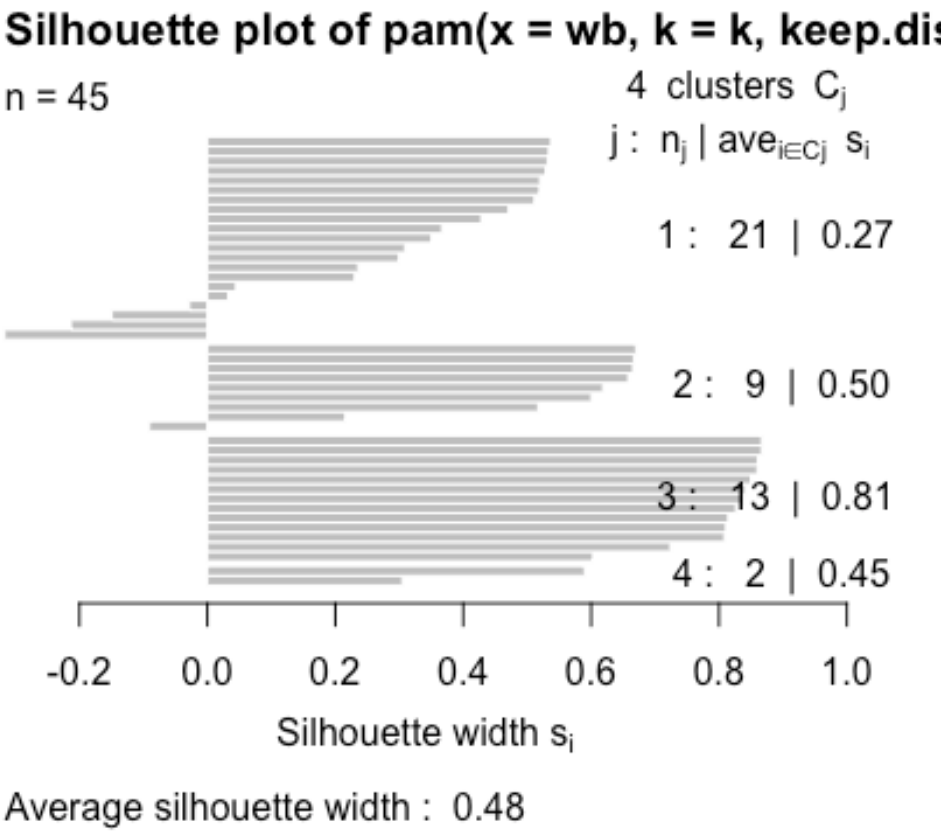
K-medoids clustering in R

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
## 13   3        70.76322             27.9          5496.345 99.78936
## 166 31        74.33722             12.5         20481.745 94.42635
## 16   4        71.62590             30.7          1211.702 59.72154
## 171 34        82.64634              2.1         52888.745 96.54015
## Clustering vector:
##   6   7  13  16  25  26  27  39  57  58  59  60  61  72  76  80  82
85
##   1   2   1   3   1   1   1   3   1   1   1   1   1   1   2   1   3
```

```
3
##  87  92  96  97 106 113 118 128 130 136 156 159 166 167 170 171 174
179
##   1   1   2   1   2   1   1   1   3   3   1   4   2   3   3   4   2
2
## 185 191 197 199 200 204 208 209 214
##   3   3   2   1   1   3   2   3   3
## Objective function:
##    build     swap
## 2521.326 2508.203
##
## 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 $\log(W(k))$ with $E^*[\log(W(k))]$ 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

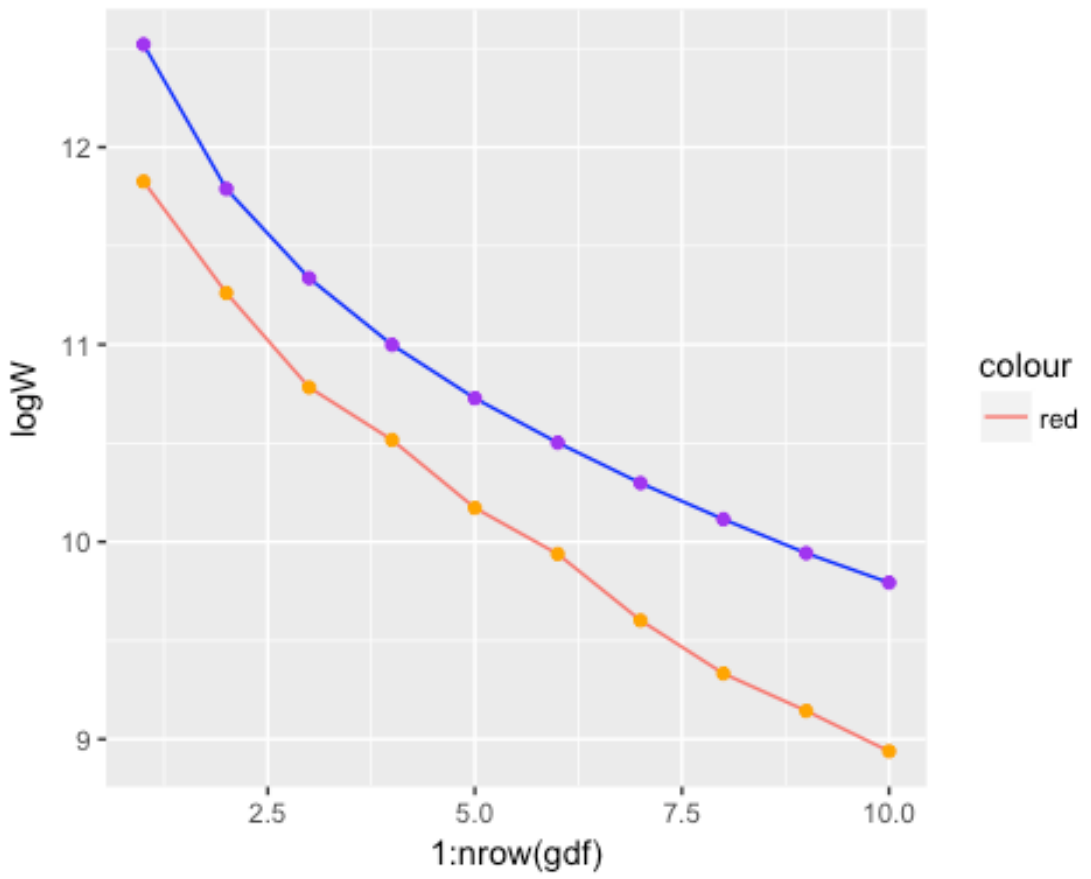
```
gap<-clusGap(wb,FUNcluster=pam,K.max=10) # Bootstrapping
gap$Tab
```

```
##             logW    E.logW       gap     SE.sim
##  [1,] 11.826708 12.521009 0.6943005 0.07153167
##  [2,] 11.261341 11.788692 0.5273507 0.06028033
##  [3,] 10.782713 11.336035 0.5533223 0.07707577
##  [4,] 10.516580 10.998472 0.4818913 0.07692385
##  [5,] 10.171902 10.727707 0.5558048 0.09042793
##  [6,]  9.938525 10.502693 0.5641680 0.09130502
##  [7,]  9.601981 10.297558 0.6955772 0.09311483
##  [8,]  9.332552 10.114634 0.7820818 0.09625402
##  [9,]  9.143441  9.942525 0.7990832 0.10137723
## [10,]  8.938730  9.793015 0.8542853 0.10666050
```

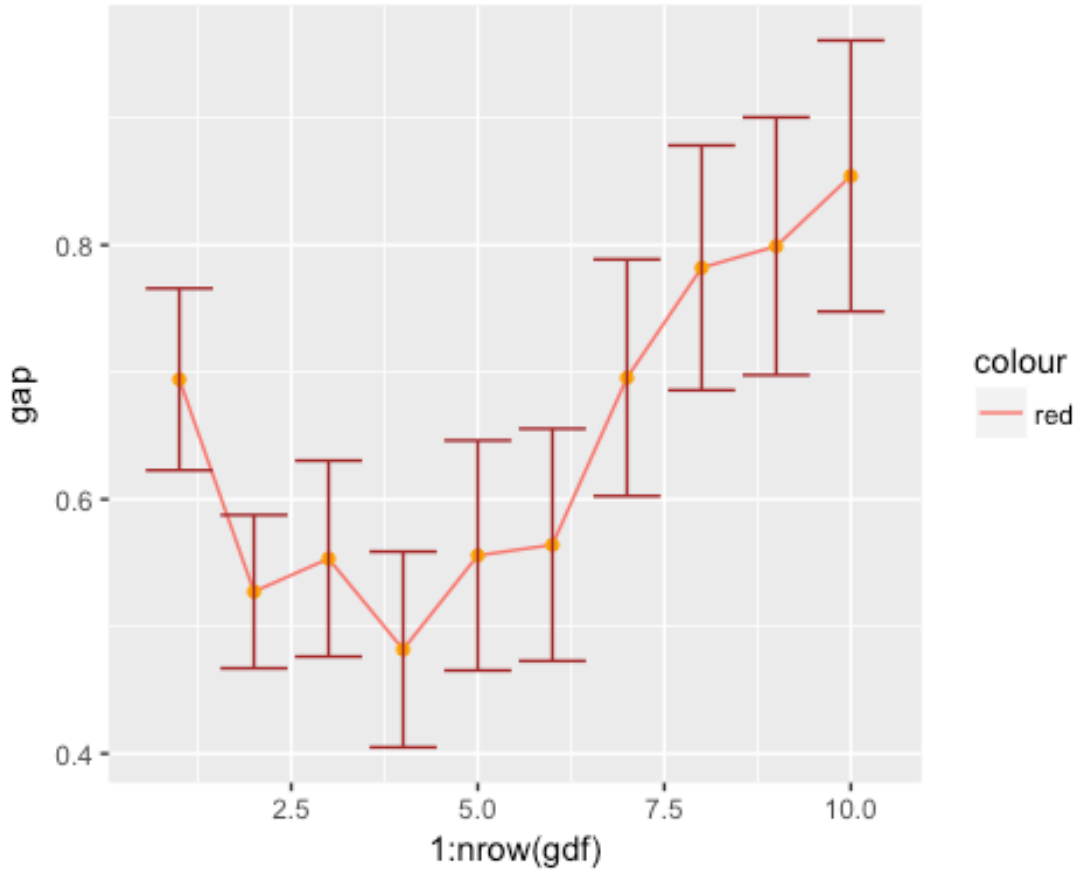
```
gdf<-as.data.frame(gap$Tab)
head(gdf)
```

```
##        logW   E.logW       gap     SE.sim
## 1 11.826708 12.52101 0.6943005 0.07153167
## 2 11.261341 11.78869 0.5273507 0.06028033
## 3 10.782713 11.33604 0.5533223 0.07707577
## 4 10.516580 10.99847 0.4818913 0.07692385
## 5 10.171902 10.72771 0.5558048 0.09042793
## 6  9.938525 10.50269 0.5641680 0.09130502
```

```
qplot(x=1:nrow(gdf),y=logW,data =
gdf,geom="line",color="red")+geom_point(aes(y=logW),color="orange")+geo
m_line(aes(y=E.logW),color="blue")+geom_point(aes(y=E.logW),color="purp
le")
```




```
# 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")
```

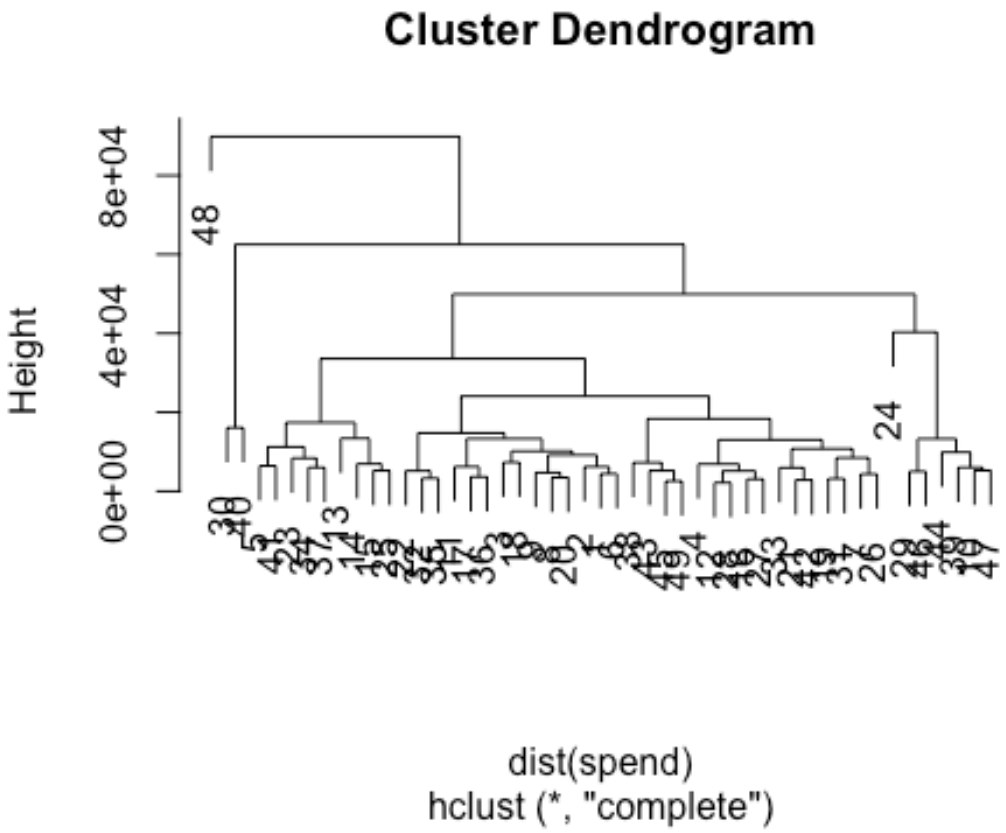


Hierarchical clustering in R

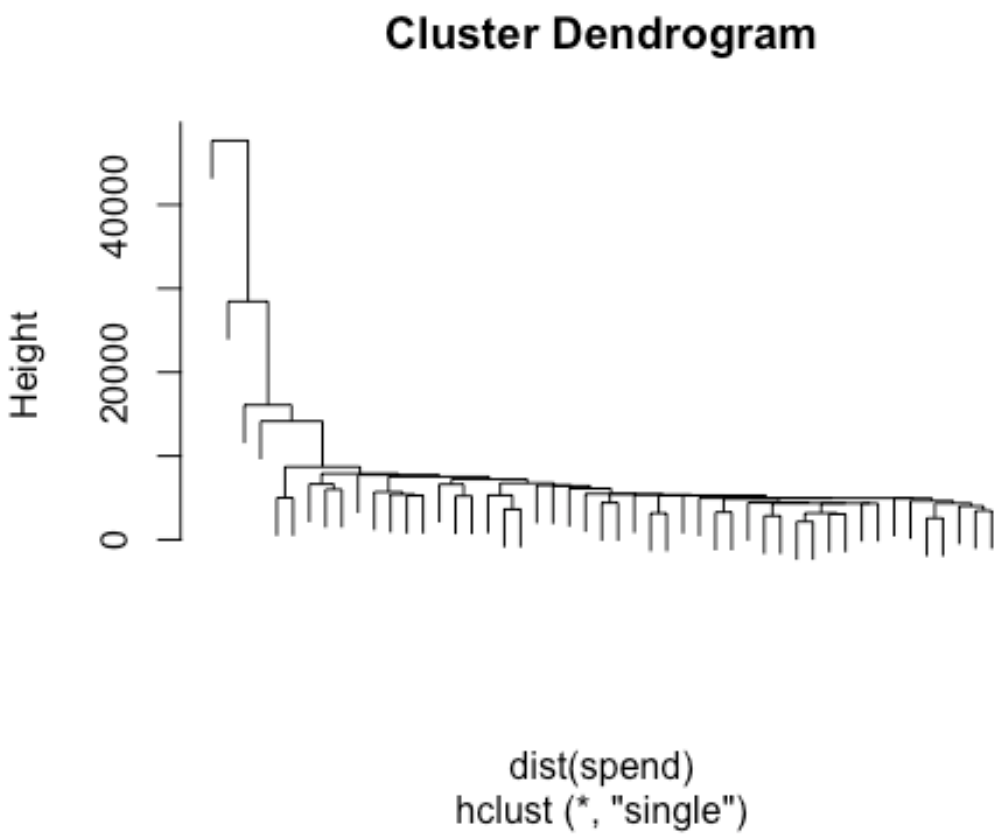
Hierarchical clustering in R

Hierarchical Clustering for the wholesale customers data set.

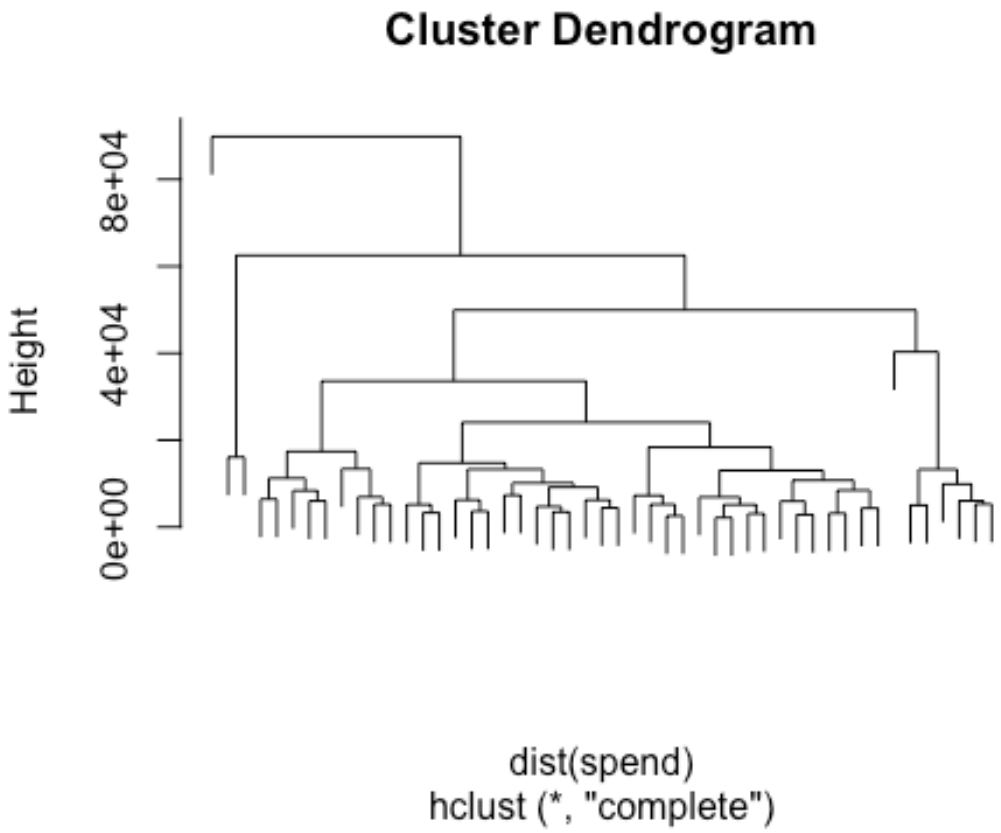
```
spend.h.clust<- hclust(d=dist(spend))
plot(spend.h.clust)
```



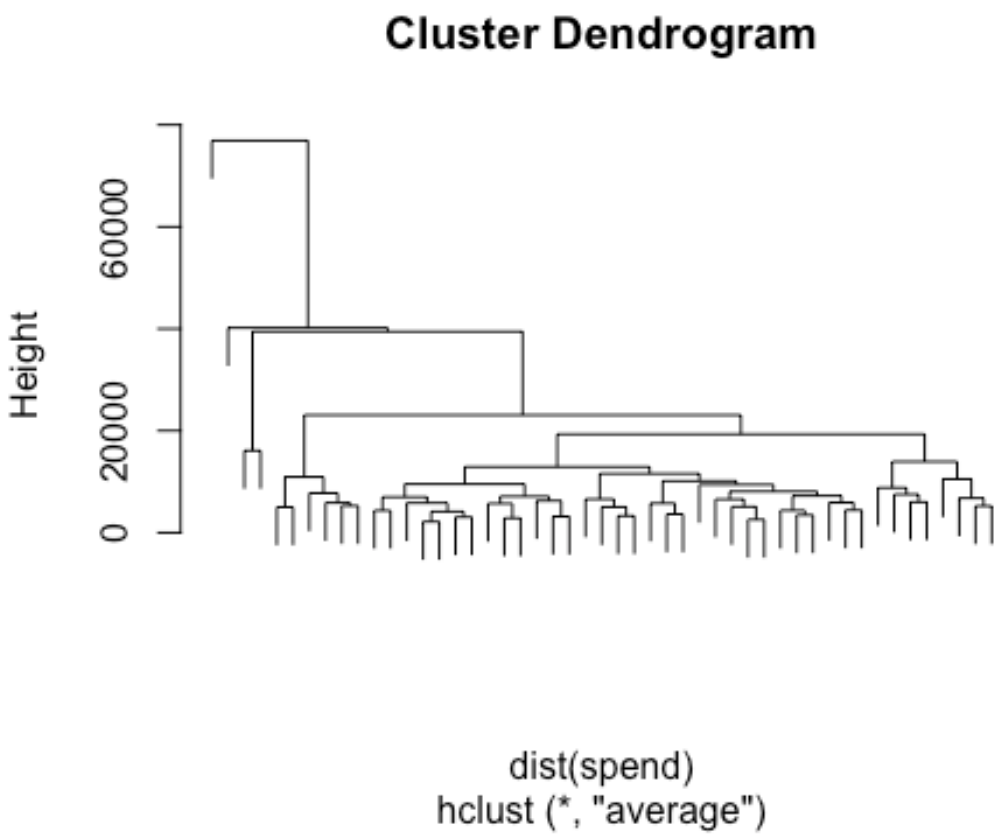

```r
spend.h.clust.si<- hclust(dist(spend), method = "single")
spend.h.clust.co<- hclust(dist(spend), method = "complete")
spend.h.clust.av<- hclust(dist(spend), method = "average")
spend.h.clust.ce<- hclust(dist(spend), method = "centroid")
plot(spend.h.clust.si, labels = FALSE)
```



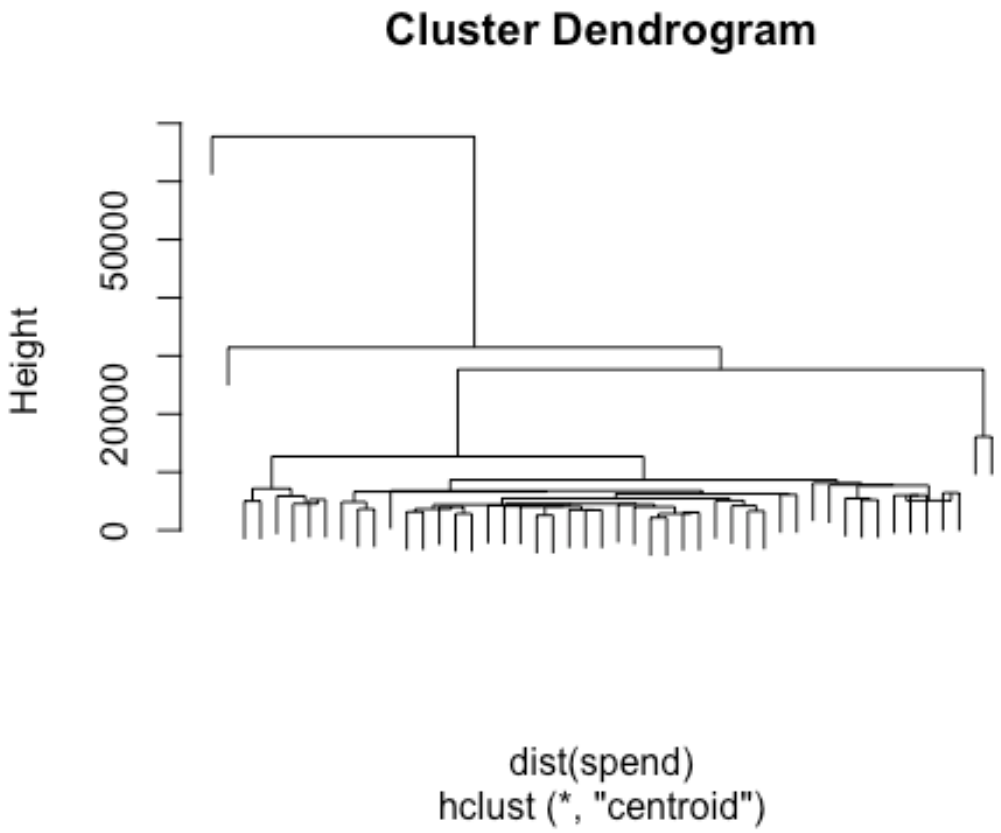

```r
plot(spend.h.clust.co, labels = FALSE)
```




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

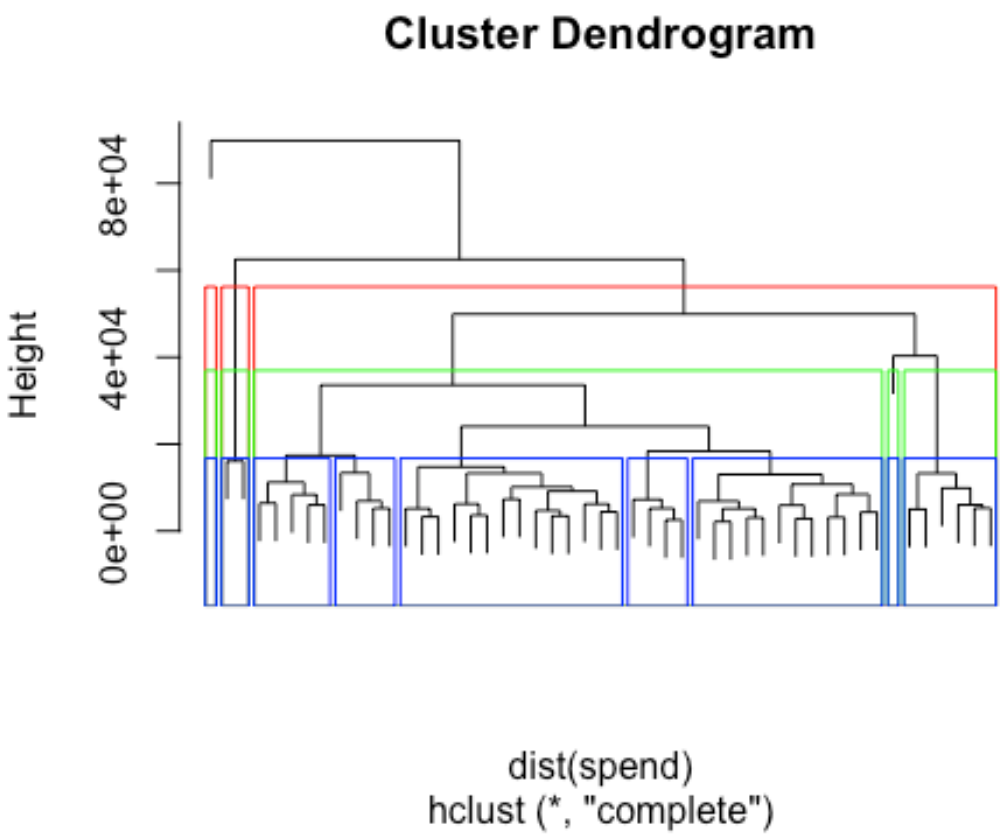



```r
plot(spend.h.clust.ce, labels = FALSE)
```



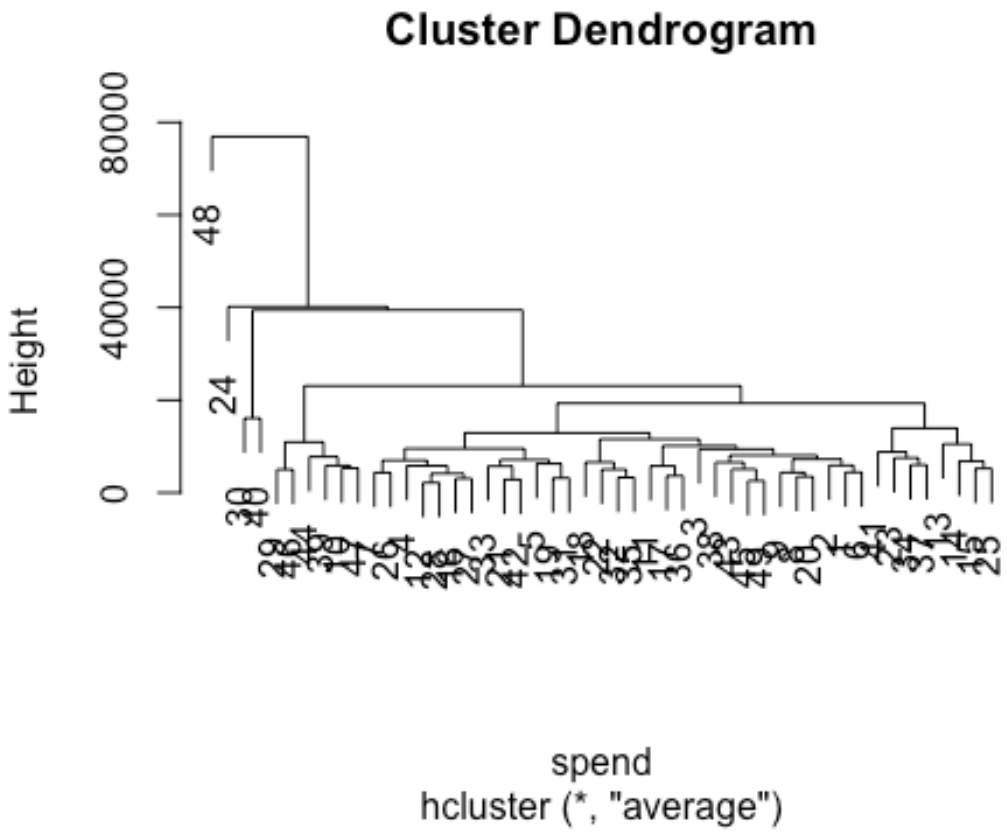
Plotting to deterimine the cluster level.

```
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")
```

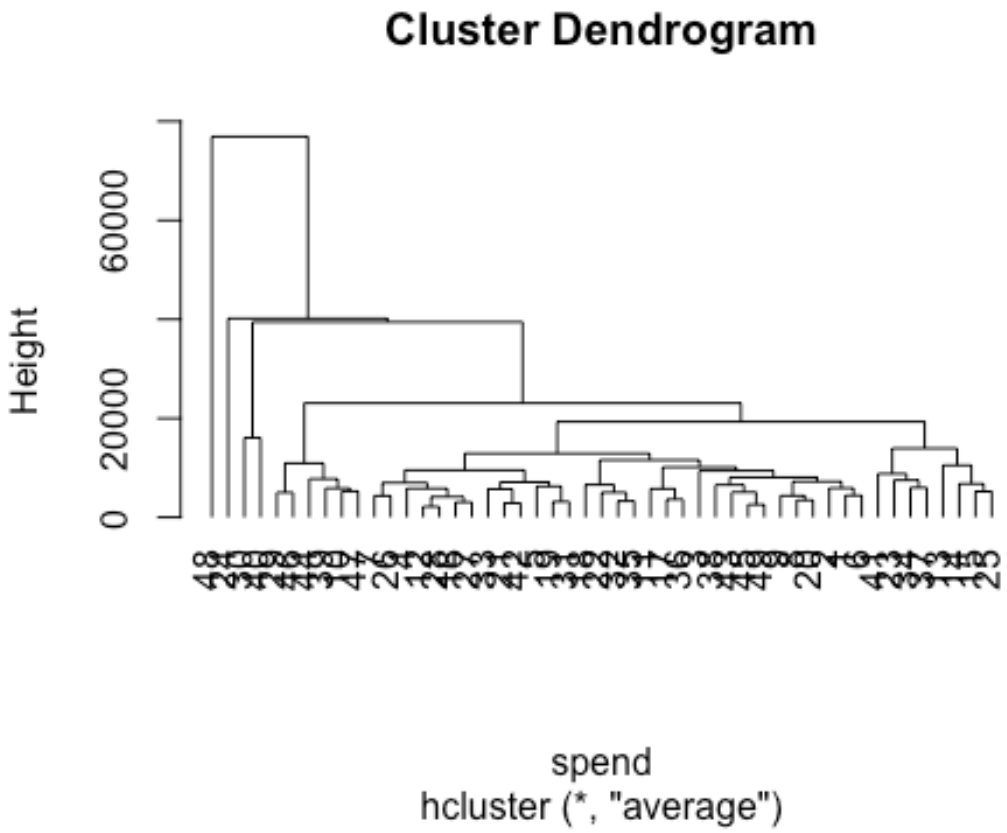


Hierarchical clustering using centroid clustering and squared Euclidean distance

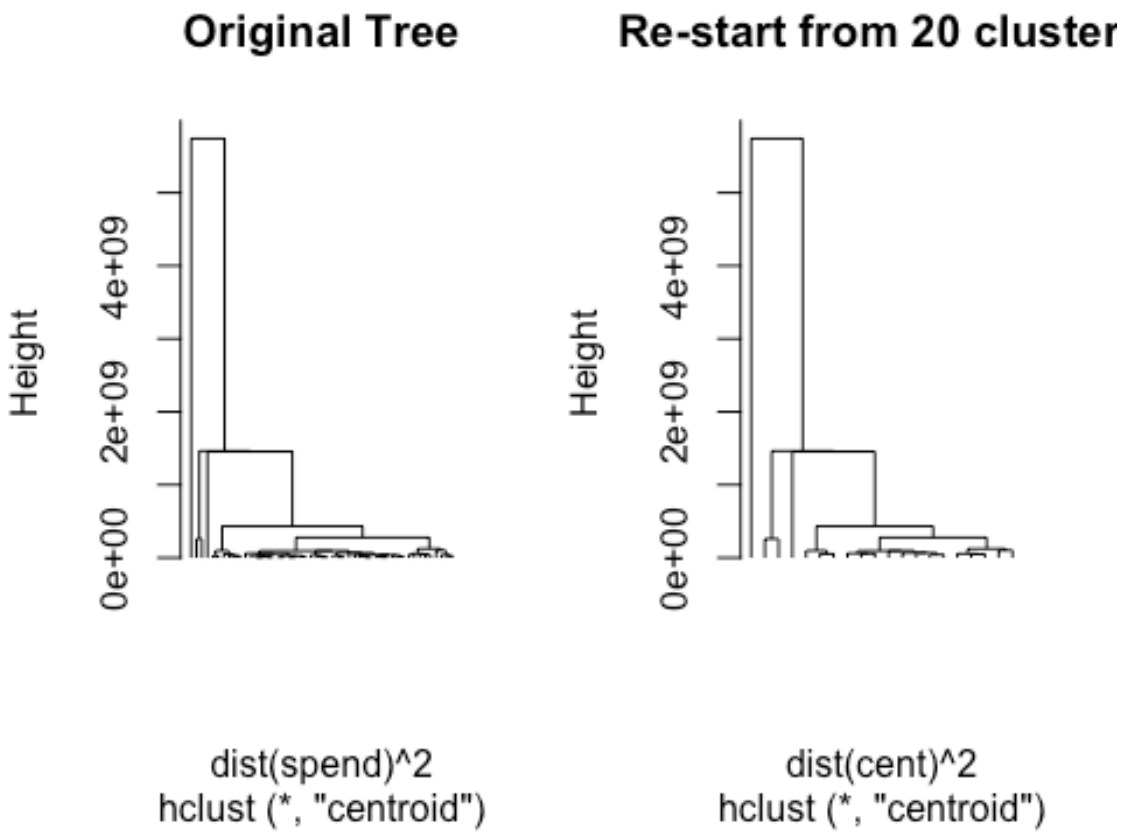
```
h_c <- hcluster(spend,link = "ave") # require(amap)
plot(h_c)
```




```
plot(h_c, hang = -1)
```



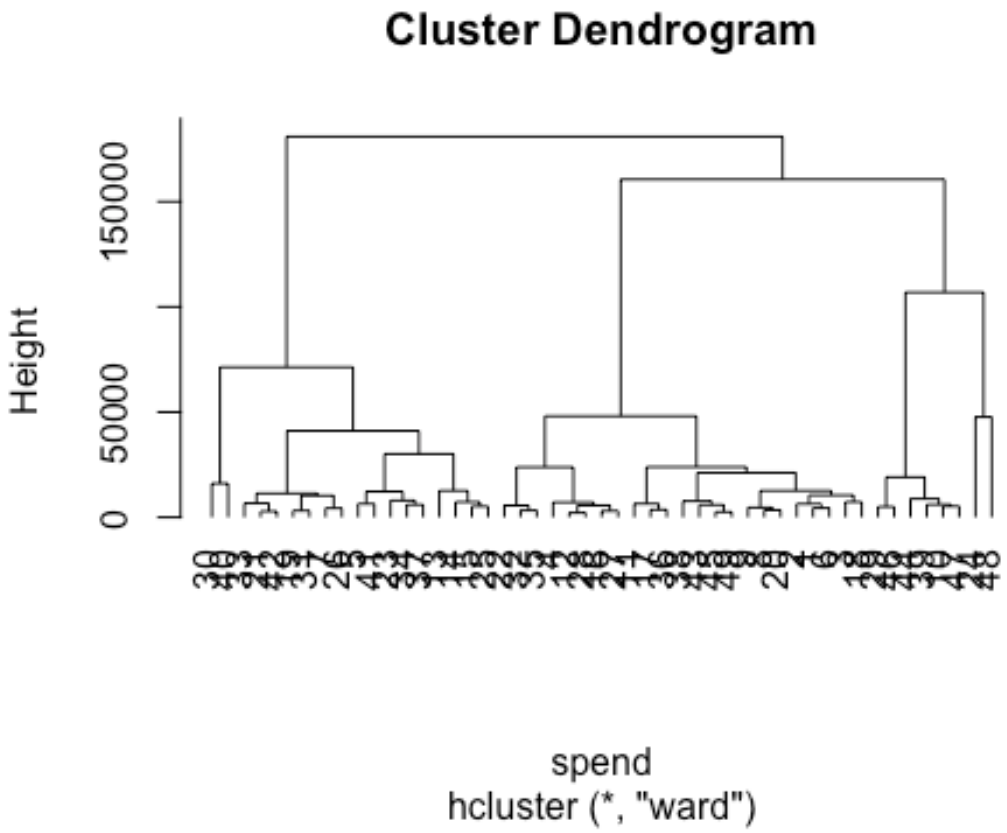

```
### centroid clustering and squared Euclidean distance
h_c<- hclust(dist(spend)^2, "cen")

### 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")
```



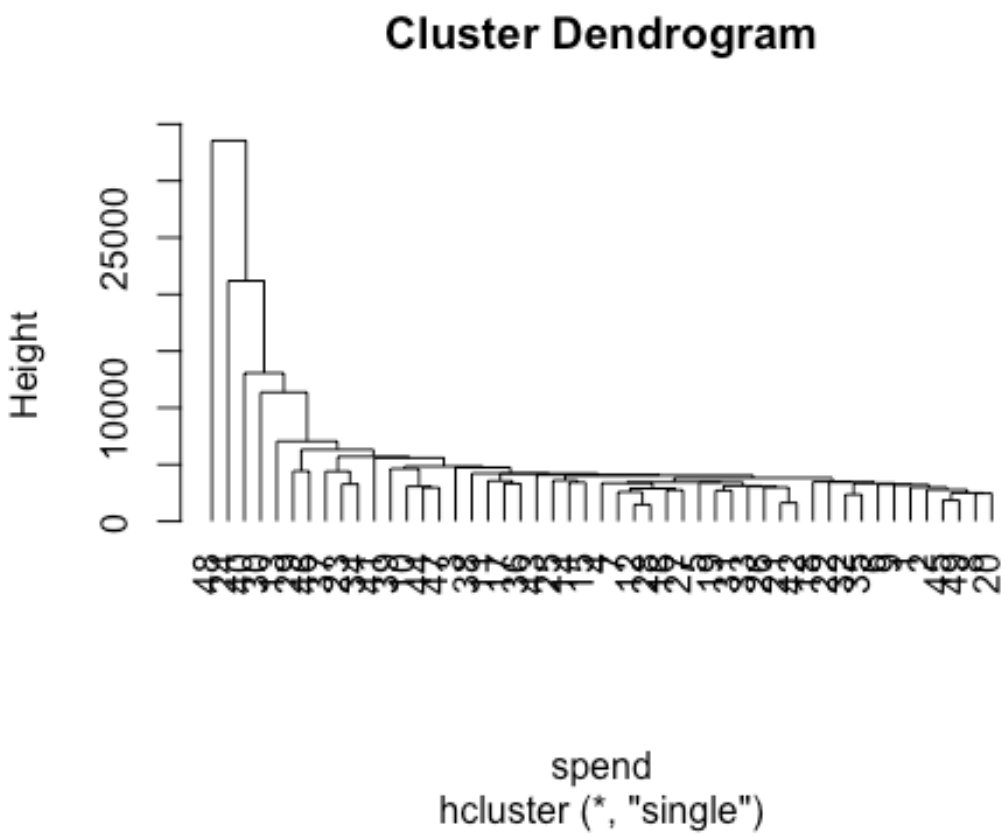

```
par(opar)
```

Other combinations

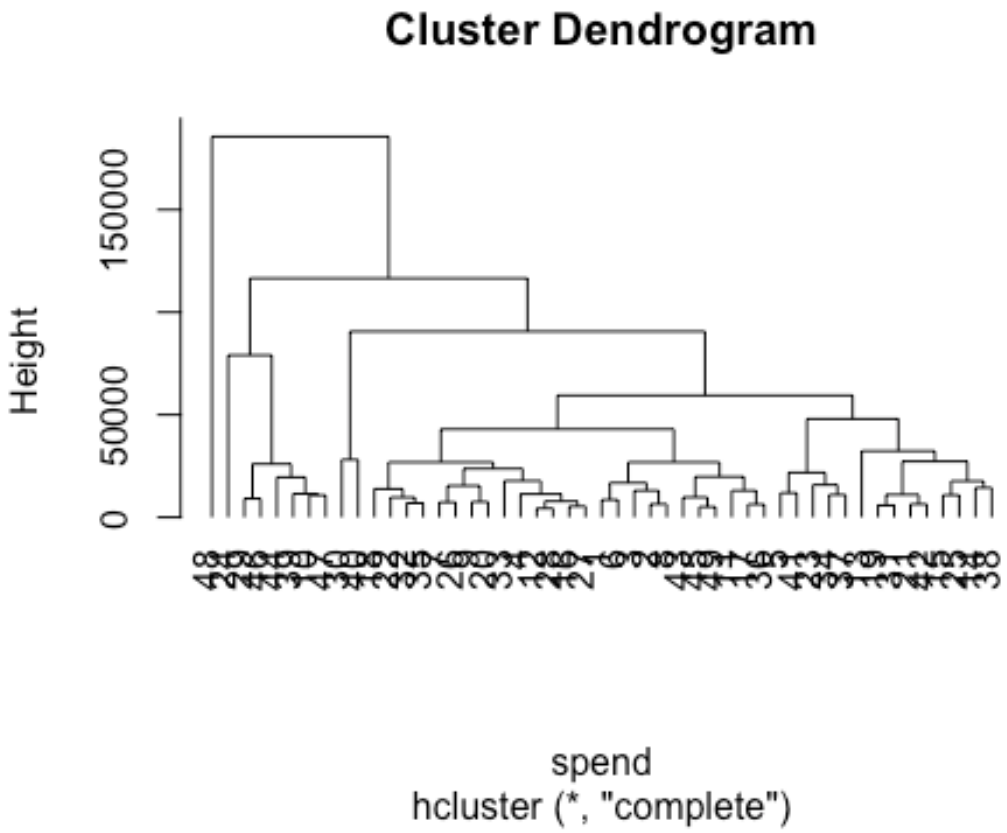
```
## other combinations

h_c <- hcluster(spend,method = "euc",link = "ward", nbproc= 1,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



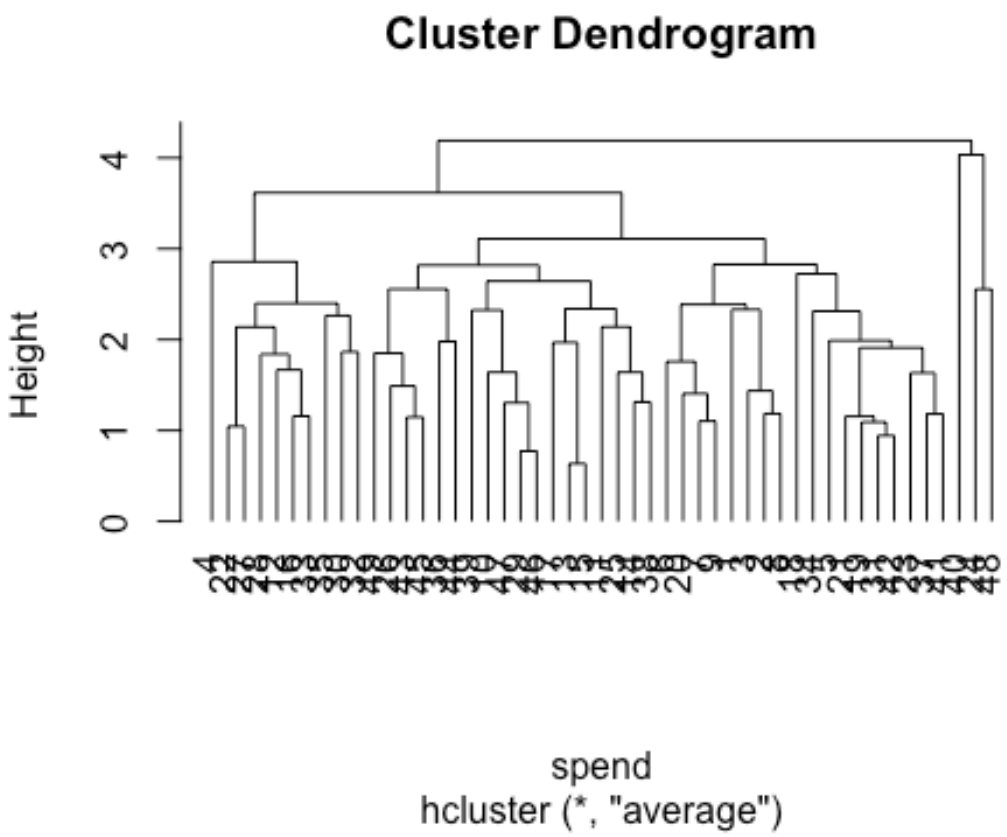

```
h_c <- hcluster(spend,method = "max",link = "single", nbproc= 2,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



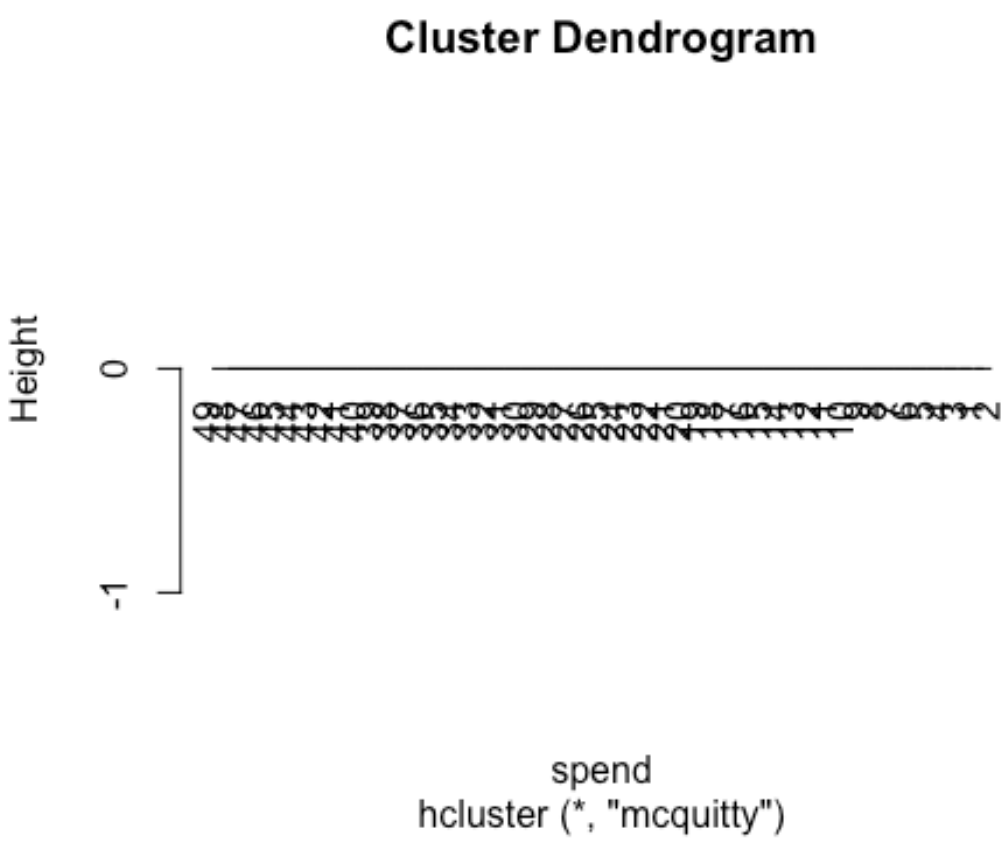

```
h_c <- hcluster(spend,method = "man",link = "complete", nbproc= 1,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



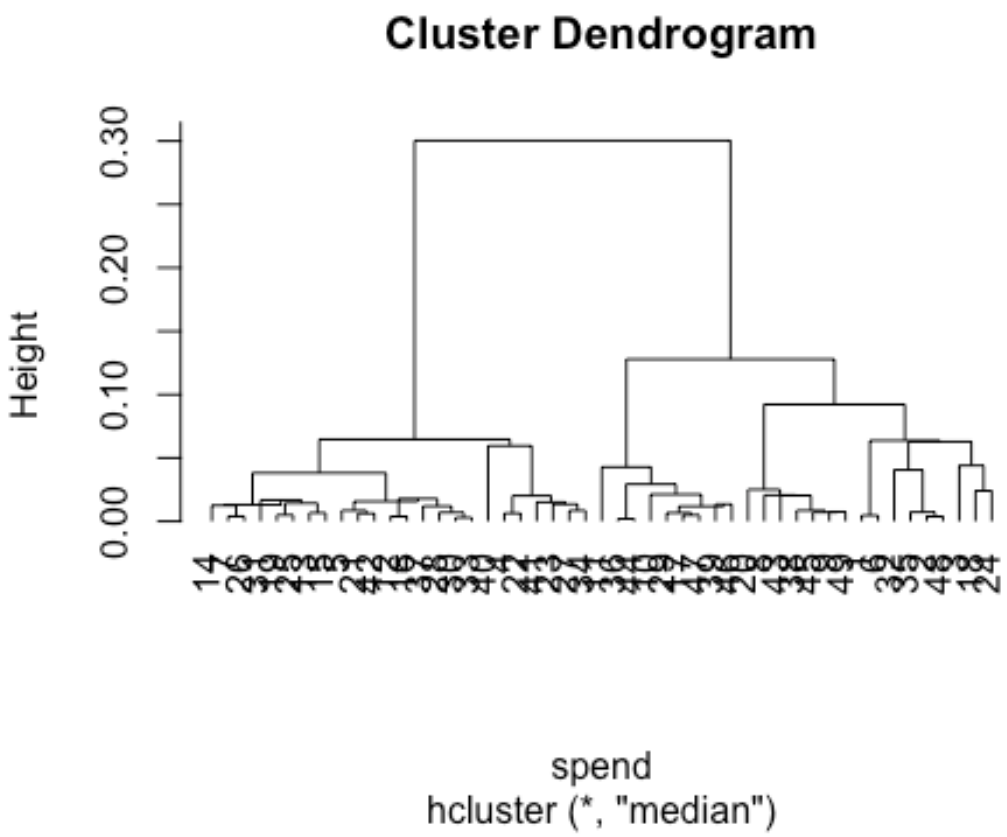

```
h_c <- hcluster(spend,method = "can",link = "average", nbproc= 2,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



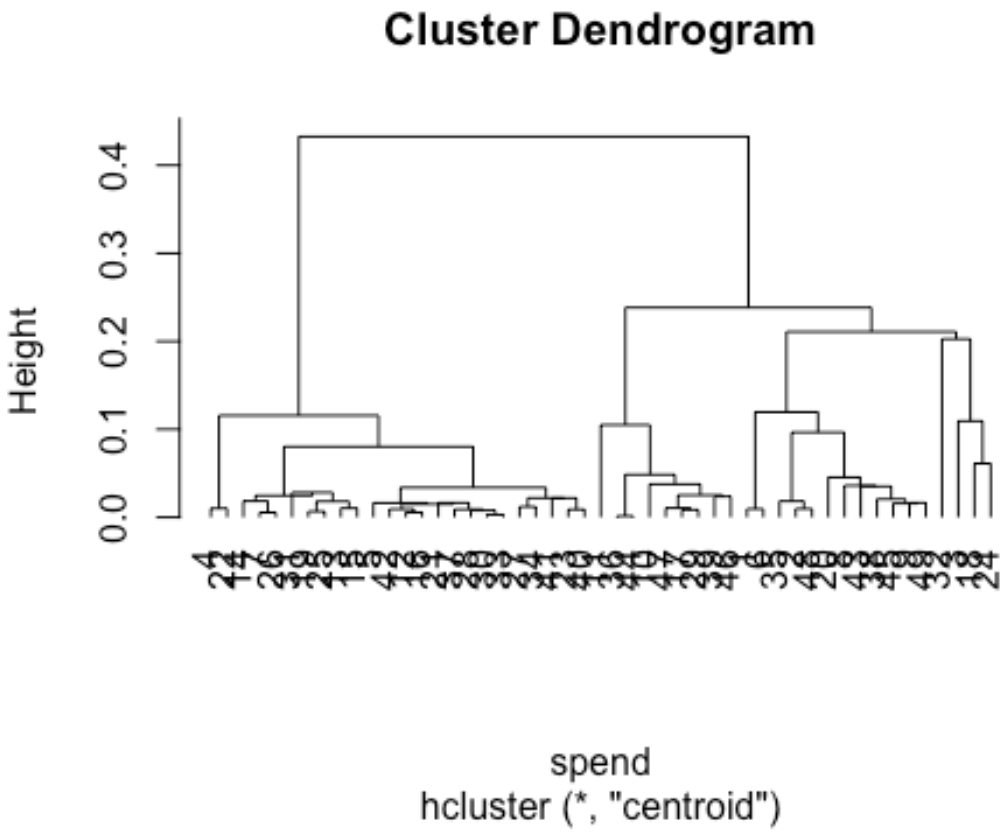

```
h_c <- hcluster(spend,method = "bin",link = "mcquitty", nbproc= 1,
                doubleprecision = FALSE)
plot(h_c, hang = -1)
```



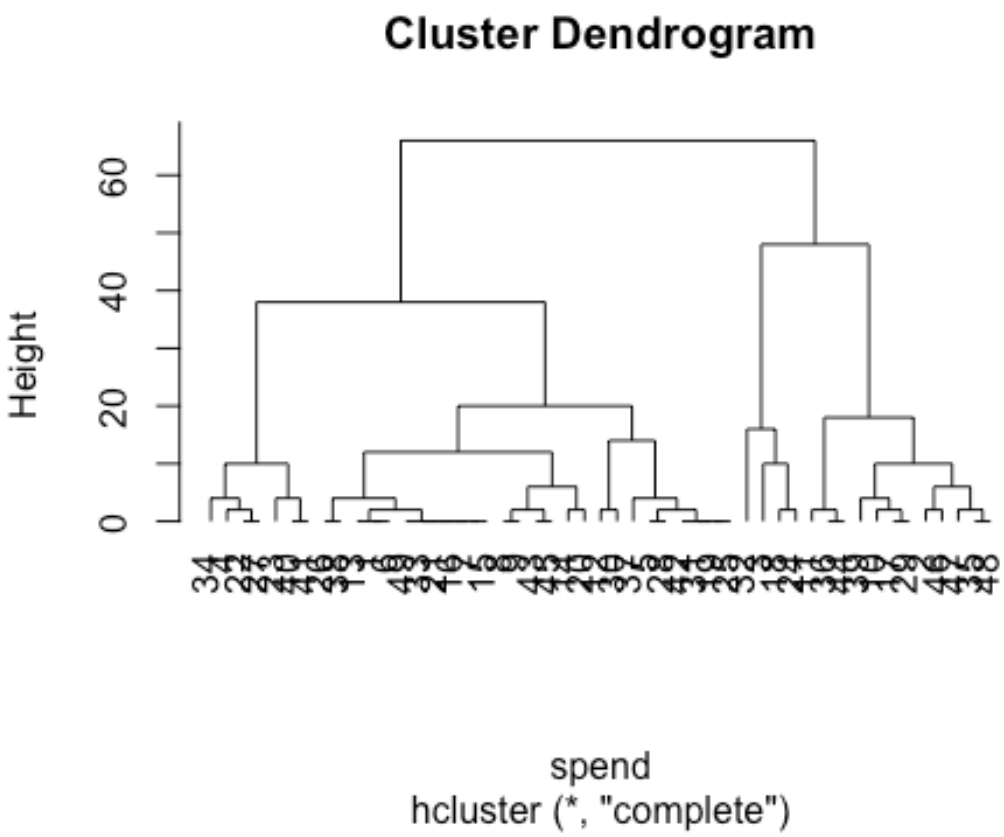

```
h_c <- hcluster(spend,method = "pea",link = "median", nbproc= 2,
                doubleprecision = FALSE)
plot(h_c, hang = -1)
```




```
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

K-means clustering is not a free lunch