**4 - Cluster Analysis**

**4.1 - Measuring Distances – Between Observations and Variables**

There are several statistical methods that start with some measure of distance between observations (*n*) or variables (*p*). In the next few chapters we will be examining multidimensional scaling (a dimension reduction technique) and cluster analysis. Both of these methods require a distance or similarity matrix between observations or variables as the starting point. In this handout we will examine methods for measuring such distances in statistics.

**4.1.1 – Measuring Distance when All Variables are Numeric/Continuous**  
To begin, consider the example below which is included in ***Appendix 1 – Basic Matrix Algebra in R*** handout you received earlier in the course. This example looks at issues regarding distances between observations based on a small set of numeric variables.

Example 4.1: U.S. Cities

Measuring Distance/Similarity Between Cities (distance between observations)  
On the basis of these three measured characteristics (median income, percent of population receiving welfare, and percentage of the population below the poverty line) how can we measure how different or similar two cities are, e.g. Detroit, MI and Minneapolis, MN?

The Euclidean distance between two vectors is given by

> View(City)

> X = as.matrix(City[,c(15,17,18)]

> xd = X[9,] 🡨 extract data for Detroit (9th row in the data frame)

> xd

income welfare poverty

18742.0 26.1 32.4

> xm = X[47,] 🡨 extract data for Minneapolis (47th row in the data frame)  
> xm

income welfare poverty

25324.0 10.5 18.5

> t(xd-xm)%\*%(xd-xm) 🡨 Euclidean distance squared

[,1]

[1,] 43323161

> sqrt(t(xd-xm)%\*%(xd-xm)) 🡨 Euclidean distance

[,1]

[1,] 6582.033

Calculations “by hand”:

= 6582.03

Clearly the distance between Detroit and Minneapolis (and any other two cities for that matter) is dominated by the median income. Thus the discrepancies between the percentages on welfare and below poverty level have little to do with the total dissimilarity between these two cities on the basis of these characteristics.

If we standardize the variables first, we put them all on the same scale.  
> sX = scale(X)

> sxd = sX[9,] 🡨 Detroit

> sxd

income welfare poverty

-1.451120 3.182602 2.410516

> sxm = sX[47,] 🡨 Minneapolis

> sxm

income welfare poverty

-0.29123182 0.05672995 0.08767880

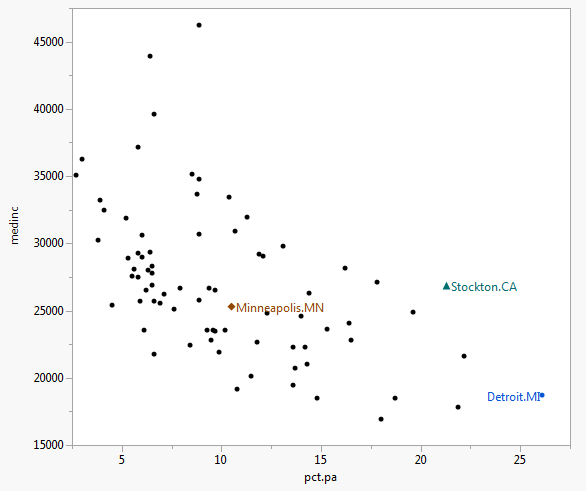
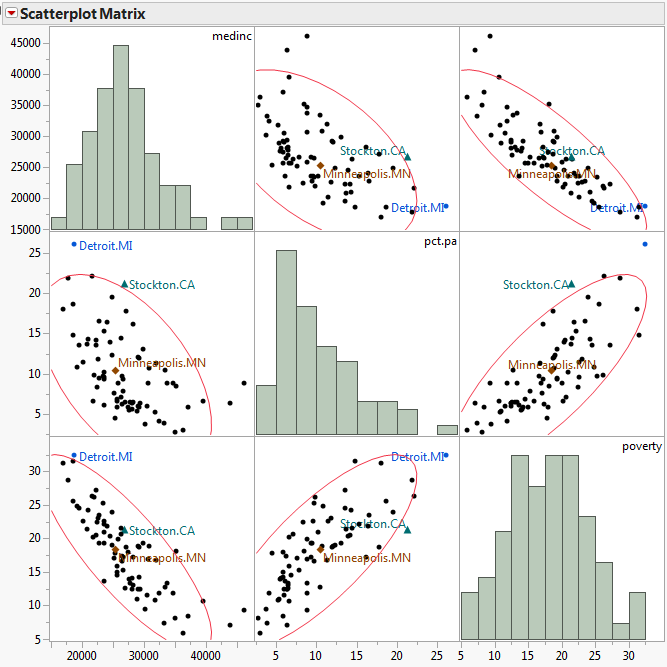
> sqrt(t(sxd-sxm)%\*%(sxd-sxm))

[,1]

[1,] 4.063495

= 4.0635

In the standardized scale the discrepancies between the percentages on welfare and below the poverty level are the largest contributors to the distance between Detroit and Minneapolis. Using a similar process the Euclidean distance between Detroit and Stockton, CA is 8134 and the standardized scale the distance is 2.52. Given the scatterplot below, which measure of discrepancy/distance is more appropriate? Note: These plots were created in JMP.

* *

As this example illustrates, simple Euclidean distance is not appropriate when variables involved in measure distances between observations are on different scales. Thus standardizing or scaling numeric/quantitative variables is almost done before measuring distances.

Euclidean distance, on scaled or unscaled variables, is not the only way in which we can measure distance between observations based upon a set of *p* exclusively numeric/quantitative variables. Below are some examples of other metrics that are used.

Distances between observations based on numeric variables

**Euclidean Distance**

**Minkowski Distance**

Note: *m = 2* is Euclidean distance

**Manhattan Distance** (Minkowski *m = 1* or *Taxi cab* metric)

**Chebyshev’s Distance**

**Canberra Distance**

What about other data types, e.g. ordinal or nominal? Can they be used in computing distances between observations and/or variables? Next we will examine how to measure distance using these variable types and then how to calculate dissimilarities based on set variables with a mixture of data types.

**4.1.2 - Measuring Distance between Observations using Ordinal Variables**

Values of this type of variable are often represented as contiguous integers, and the realizable values are considered to be an ordered set . To compute distance we replace the values by:

and we can then treat them a numeric/quantitative variables on this transformed scale.  
 **4.1.3 - Measuring Distance between Observations using Categorical Variables**  
  
With unordered categorical variables we could simply use 1 if they don’t agree and 0 otherwise. This is the approach JMP uses I believe.

If there are several categorical variables involved in measuring the distance between observations then we can count the number of matches between observations *i* and *j* relative to the “total” number of potential matches. Total is in quotes here because we may want to give more credit for certain type of matches, particularly if all the nominal variables are binary/dichotomous (i.e. have only 2 levels). It is important to note any multilevel nominal/categorical variable can be turned into a sequence of binaries, one for each level. However, doing this will create a great deal of 0-0 matches!

There are several indices in the literature that are used to measure similarity/dissimilarity between observations when we have a set of variables that are all binary. They all start essentially with a table that counts the number of matches and mismatches.

**Observation *j***

|  |  |  |
| --- | --- | --- |
| **Observation *i*** | 1 (TRUE) | 0 (FALSE) |
| 1 (TRUE) |  |  |
| 0 (FALSE) |  |  |

a = # of variables where observations *i* and *j* are a 1-1 match.  
b = # of variables where observation *i* is 1, but *j* is 0

c = # of variables where observation *i* is 0, but *j* is 1

d = # of variables where observations *i* and *j* are a 0-0 match.

On the basis of these counts we can measure ***similarity/dissimilarity*** a number of ways.

Matching Coefficient (dissimilarity)

Jaccard Coefficient (dissimilarity)

this measure completely ignores 0-0 matches

Another way to measure dissimilarity is to convert similarity measures to dissimilarity measures. Below are ways to measure ***similarities*** between observations *i* and *j* for a set of nominal variables. In the table below (*p = a + b + c + d*).

|  |  |
| --- | --- |
| **Coefficient of Similarity** | **Rationale** |
|  | Equal weights for 1-1 matches and 0-0 matches. |
|  | Double weight for 1-1 matches and 0-0 matches. |
|  | Double weight for unmatched pairs |
|  | No 0-0 matches in numerator |
|  | No 0-0 matches in numerator or denominator.  (The 0-0 matches are treated as irrelevant). |
|  | No 0-0 matches in numerator or denominator.  Double weight for 1-1 matches. |
|  | No 0-0 matches in numerator or denominator.  Double weight for unmatched pairs. |
|  | Ratio of matches to mismatches with 0-0 matches excluded. |

There are at least a dozen more besides these, but I think you probably get the general idea. In many applications the 1-1 matches are more important than the 0-0 matches and several of these and other measures reflect that fact. The dist()function in the proxy library has 20 different options for measuring similarities between observations based on binary variables.

When a variable is nominal and has *k* levels , we can create *k* binary variables, one for each level of the variable. We can then use the binary measures of similarity above, however it will be very important to discount 0-0 matches as there will be at least of them for a nominal variable with levels.

Example 4.2: Edibility of Mushrooms and their Physical Characteristics

**Aside:** These data were used in the 2001 Undergraduate Data Analysis Contest (UDAC) which is now defunct (replaced by MUDAC). The goal of the problem using these data was to develop a predictive model to classify a mushroom as poisonous or edible on the basis of a set of nominal characteristics of the mushrooms. Andrea Nibbe Storlie, a WSU student, earned 1st place in both aspects of the competition (written summary or her results and the accuracy of her predictions – 100% correct!). She went on to become the first undergraduate to win the prestigious Gertrude Cox Scholarship which awarded by the American Statistical Association (ASA) to the best female statistics student in the country.

The mushroom attributes are defined below:

Variable Levels \*  
     Poisonous edible=e,poisonous=p

X1 = cap-shape: bell=b,conical=c,convex=x,flat=f,  
 knobbed=k,sunken=s  
 X2 = cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s  
 X3 = cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,  
 pink=p,purple=u,red=e,white=w,yellow=y  
 X4 = bruises?: true=t,false=f  
 X5 = odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,  
 musty=m,none=n,pungent=p,spicy=s  
 X6 = gill-attachment: attached=a, free=f  
 X7 = gill-spacing: close=c,crowded=w  
 X8 = gill-size: broad=b,narrow=n  
 X9 = gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,  
 green=r,orange=o,pink=p,purple=u,red=e,  
 white=w,yellow=y  
 X10 = stalk-shape: enlarging=e,tapering=t  
 X11 = stalk-root: bulbous=b,club=c,cup=u,equal=e,  
 rhizomorphs=z,rooted=r,missing=? 🡨 has lots of missing values!  
 X12 = stalk-surface-above-ring: ibrous=f,scaly=y,silky=k,smooth=s  
 X13 = stalk-surface-below-ring: ibrous=f,scaly=y,silky=k,smooth=s 🡨 has lots of missing values!  
 X14 = stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X15 = stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X16 = veil-type: partial=p,universal=u 🡨 All partial (no universal)!  
 X17 = veil-color: brown=n,orange=o,white=w,yellow=y  
 X18 = ring-number: none=n,one=o,two=t  
 X19 = ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,  
 none=n,pendant=p,sheathing=s,zone=z 🡨 some levels not represented  
 X20 = spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,  
 orange=o,purple=u,white=w,yellow=y  
 X21 = population: abundant=a,clustered=c,numerous=n,  
 scattered=s,several=v,solitary=y  
 X22 = habitat: grasses=g,leaves=l,meadows=m,paths=p,  
 urban=u,waste=w,woods=d

The data frame Mushrooms.train can be formed by reading in the file **mushrooms.csv**.

We will consider applying some of the distance measures discussed above to these data using two of the physical characteristics of these mushrooms. Let’s consider whether or not the mushroom has bruises ( and the stalk surface above the ring ( and put them in a new data frame called mush.sub.

> table(Mushrooms.train$x4)

f t

2387 1675

> table(Mushrooms.train$x12)

f k s y

286 1176 2585 15

> mush.sub = data.frame(Bruises=Mushrooms.train$x4,SSAR=Mushrooms.train$x12)

> mush.sub[1000:1005,] 🡨 look at the values for mushrooms 1000 - 1005

Bruises SSAR

1000 t s

1001 f k

1002 f s

1003 f k

1004 t s

1005 t s

As SSAR ( has 4 levels, we can create a dummy variable for each level using the as.dummy command in the cab package. This function will create a dummy for each level of all nominal variables, including dichotomous ones. This is equivalent to the complete disjunctive table used in the conducting an MCA.

> mush.dummy = as.dummy(mush.sub) 🡨 requires cab package installed and loaded.

> mush.dummy[1000:1005,]

Bruises f Bruises t SSAR f SSAR k SSAR s SSAR y

1000 FALSE TRUE FALSE FALSE TRUE FALSE

1001 TRUE FALSE FALSE TRUE FALSE FALSE

1002 TRUE FALSE FALSE FALSE TRUE FALSE

1003 TRUE FALSE FALSE TRUE FALSE FALSE

1004 FALSE TRUE FALSE FALSE TRUE FALSE

1005 FALSE TRUE FALSE FALSE TRUE FALSE

Compute the Jaccard similarity/dissimilarity between mushrooms 1000 and 1001.

Compute the Jaccard similarity/dissimilarity between mushrooms 1002 and 1003.

Compute the Jaccard similarity/dissimilarity between mushrooms 1004 and 1005.

**4.1.4 - Relationship Between Similarities and Dissimilarities**In general we can turn similarities into dissimilarities intuitively by subtracting them from 1, assuming the similarity between an observation and itself is 1, i.e. max similarity = 1. Thus we have, , although some recommend using or  
 .

We can use R to compute the dissimilarity/similarity matrix for all pairs of mushrooms in our small data set mush.sub using functions in the library proxy. We can display the available similarity/dissimilarity measures (along with their formulae if we want) as shown below.

> library(proxy)  
> summary(pr\_DB,"long")

\* Similarity measures:

Jaccard/binary/Reyssac/Roux (binary) = a / (a + b + c)

Kulczynski1 (binary) = a / (b + c)

Kulczynski2 (binary) = [a / (a + b) + a / (a + c)] / 2

Mountford (binary) = 2a / (ab + ac + 2bc)

Fager/McGowan (binary) = a / sqrt((a + b)(a + c)) - sqrt(a + c) / 2

Russel/Rao (binary) = a / n

simple matching/Sokal/Michener (binary) = (a + d) / n

Hamman (binary) = ([a + d] - [b + c]) / n

Faith (binary) = (a + d/2) / n

Tanimoto/Rogers (binary) = (a + d) / (a + 2b + 2c + d)

Dice/Czekanowski/Sorensen (binary) = 2a / (2a + b + c)

Phi (binary) = (ad - bc) / sqrt[(a + b)(c + d)(a + c)(b + d)]

Stiles (binary) = log(n(|ad-bc| - 0.5n)^2 / [(a + b)(c + d)(a + c)(b + d)])

Michael (binary) = 4(ad - bc) / [(a + d)^2 + (b + c)^2]

Mozley/Margalef (binary) = an / (a + b)(a + c)

Yule (binary) = (ad - bc) / (ad + bc)

Yule2 (binary) = (sqrt(ad) - sqrt(bc)) / (sqrt(ad) + sqrt(bc))

Ochiai (binary) = a / sqrt[(a + b)(a + c)]

Simpson (binary) = a / min{(a + b), (a + c)}

Braun-Blanquet (binary) = a / max{(a + b), (a + c)}

cosine/angular (metric) = xy / sqrt(xx \* yy)

eJaccard/extended\_Jaccard (metric) = xy / (xx + yy - xy)

correlation (metric) = xy / sqrt(xx \* yy) for centered x,y

Chi-squared (nominal) = sum\_ij (o\_i - e\_i)^2 / e\_i

Phi-squared (nominal) = [sum\_ij (o\_i - e\_i)^2 / e\_i] / n

Tschuprow (nominal) = sqrt{[sum\_ij (o\_i - e\_i)^2 / e\_i] / n / sqrt((p - 1)(q - 1))}

Cramer (nominal) = sqrt{[Chi / n)] / min[(p - 1), (q - 1)]}

Pearson/contingency (nominal) = sqrt{Chi / (n + Chi)}

Gower (other) = Sum\_k (s\_ijk \* w\_k) / Sum\_k (d\_ijk \* w\_k)

\* Distance measures:

Euclidean/L2 (metric) = sqrt(sum\_i (x\_i - y\_i)^2))

Mahalanobis (metric) = sqrt((x - y) Sigma^(-1) (x - y))

Bhjattacharyya (metric) = sqrt(sum\_i (sqrt(x\_i) - sqrt(y\_i))^2))

Manhattan/City-Block/L1/taxi (metric) = sum\_i |x\_i - y\_i|

supremum/max/maximum/Tschebyscheff/Chebyshev (metric) = max\_i |x\_i - y\_i|

Minkowski/Lp (metric) = (sum\_i (x\_i - y\_i)^p)^(1/p)

Canberra (metric) = sum\_i |x\_i - y\_i| / |x\_i + y\_i|

Wave/Hedges (metric) = sum\_i (1 - min(x\_i, y\_i) / max(x\_i, y\_i))

divergence (metric) = sum\_i (x\_i - y\_i)^2 / (x\_i + y\_i)^2

Kullback/Leibler (metric) = sum\_i [x\_i \* log((x\_i / sum\_j x\_j) / (y\_i / sum\_j y\_j)) / sum\_j   
 x\_j)]

Bray/Curtis (metric) = sum\_i |x\_i - y\_i| / sum\_i (x\_i + y\_i)

Soergel (metric) = sum\_i |x\_i - y\_i| / sum\_i max{x\_i, y\_i}

Levenshtein (other) = Number of insertions, edits, and deletions between to strings

Podani/discordance (metric) = 1 - 2 \* (a - b + c - d) / (n \* (n - 1))

Chord (metric) = sqrt(2 \* (1 - xy / sqrt(xx \* yy)))

Geodesic (metric) = arccos(xy / sqrt(xx \* yy))

Whittaker (metric) = sum\_i |x\_i / sum\_i x - y\_i / sum\_i y| / 2

Hellinger (metric) = sqrt(sum\_i (sqrt(x\_i / sum\_i x) - sqrt(y\_i / sum\_i y)) ^ 2)

fJaccard/fuzzy\_Jaccard (metric) = sum\_i (min{x\_i, y\_i} / max{x\_i, y\_i})

> summary(pr\_DB,"short")

\* Similarity measures:

Braun-Blanquet, Chi-squared, correlation, cosine, Cramer, Dice, eJaccard, Fager, Faith, Gower,

Hamman, Jaccard, Kulczynski1, Kulczynski2, Michael, Mountford, Mozley, Ochiai, Pearson, Phi,

Phi-squared, Russel, simple matching, Simpson, Stiles, Tanimoto, Tschuprow, Yule, Yule2

\* Distance measures:

Bhjattacharyya, Bray, Canberra, Chord, divergence, Euclidean, fJaccard, Geodesic, Hellinger,

Kullback, Levenshtein, Mahalanobis, Manhattan, Minkowski, Podani, Soergel, supremum, Wave,

Whittaker

Example 4.2: Mushrooms (con’td)  
We will use the Jaccard index as our similarity metric as this metric ignores (False/False or 0-0) matches in both the numerator and the denominator, i.e.

where,

a = # of variables where observations i and j are a 1-1 match.

b = # of variables where observation i is 1, but j is 0

c = # of variables where observation i is 0, but j is 1.

> mush.sim = simil(mush.dummy,method=”Jaccard”) 🡨 these are similarities   
> summary(mush.sim)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.0000 0.0000 0.3333 0.4363 1.0000 1.0000  
> mush.dist = dist(mush.dummy,method="Jaccard") 🡨 these are dissimilarities

> summary(mush.dist)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.0000 0.0000 0.6667 0.5637 1.0000 1.0000

If we wish to use the alternative conversions of similarity to dissimilarity ( then we need to convert the similarities to a matrix, perform the conversion, and then express them as a distance matrix.

> mush.sim = simil(mush.dummy,method=”Jaccard”) 🡨 these are similarities   
> summary(mush.sim)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1. 0.0000 0.3333 0.4363 1.0000 1.0000

Convert similarities to a full matrix  
> mush.sim.mat = as.matrix(mush.sim,diag=1)

Convert similarities to distances using the alternative formulae

> mush.dist.mat1 = sqrt(1-mush.sim.mat)

> mush.dist.mat2 = sqrt(2\*(1-mush.sim.mat))

Convert full distance matrices back to lower-triangular storage  
> mush.dist1 = as.dist(mush.dist.mat1)

> mush.dist2 = as.dist(mush.dist.mat2)

Summarize the distances in the alternative scales  
> summary(mush.dist1)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.0000 0.0000 0.8165 0.6249 1.0000 1.0000

> summary(mush.dist2)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1. 0.0000 1.1550 0.8838 1.4140 1.4140

Compare to original distances   
> summary(mush.dist)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1. 0.0000 0.6667 0.5637 1.0000 1.0000

**4.1.5 - Combining Distance Measures when Variables are a Mixture of Types**

When the variables in an data matrix are a mixture of types and we wish to measure distance between observations *i* and *j* (i.e. rows *i* and *j* ) we need a method to combine the different distance metrics into an overall distance between them to obtain .

Gower (1971) proposed the following distance metric:

where,

* if either or is missing
* if and variable is asymmetric binary
* otherwise.

The distance portion () is defined as follows according to the data type of variable *k*:

1. If *k* is binary or nominal: if and otherwise.
2. If *k* is numeric: (scaled taxi-cab metric)
3. If *k* is ordinal compute ranks 1,…, and compute

and treating them as numeric using (2) above.

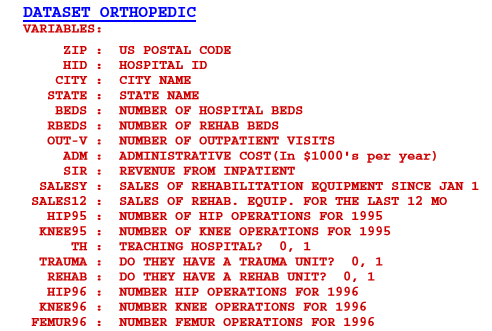
The function daisy()in the cluster library has a Gower’s metric option.

The end result of all of this is an symmetric distance or dissimilarity matrix of the form:

where the diagonal elements are all 0 and the off-diagonal elements are all of the pairwise distances/dissimilarities between each pair of observations . The example of the next page shows an example of Gower’s metric.

**Example of mixture of data types (**uses daisy in the cluster library**)**

Example 2.3: Orthopedic Equipment Sales (**Ortho.csv**)  
One goal in the analysis of these data is to identify hospitals that are underperforming in terms of orthopedic sales given what we know about them. Another thing we might want to look for are groups of similar hospitals on the basis of these characteristics. To do this we need to some measure of similarity/dissimilarity between hospitals given their characteristics. The variables available for this purpose are shown below:



You will notice that three of the variables are binary/dichotomous categorical variables: Teaching Hosptial, Trauma Unit, and Rehabilitation Unit. The others are numeric, and as we will see later, most are very skewed right with lots of zeroes. For the purpose of this example of measuring distance with a mixture of data types we will focus on three dichotomous variables – whether or not the hospital is a teaching hospital, has a trauma unit, and whether or not it has a rehabilitation unit. Because of extreme right skewness with lots zeroes, we will consider the for the numbers of each type of orthopedic surgery conducted at the hospital in the past 12 months.

> library(cluster)

> Ortho = read.table(file.choose(),header=T,sep=",") # read in **Ortho.csv**  
> names(Ortho)

[1] "ZIP" "HospID" "City" "State" "Beds" "RBeds" "Outpatients" "Admin"

[9] "Inpatient" "Hip95" "Knee95" "SalesYr" "Sales12" "Teach" "Trauma" "Rehab"

[17] "Hip96" "Knee96" "Femur96" "logBeds" "logOut" "logRBeds" "logAdmin" "logInpat"

[25] "logHip95" "logKnee95" "logSales" "logSales12" "logHip96" "logKnee96" "logFemur96"

> Ortho.test = Ortho[1:10,c(14:16,29:31)] 🡨 extract info for first 10 hospitals in our data

on the basis of the six variables discussed above.

> Ortho.test

Teach Trauma Rehab logHip96 logKnee96 logFemur96

1 0 0 0 4.624973 4.927254 4.615121

2 1 0 0 4.204693 3.713572 5.802118

3 0 0 0 4.584967 4.174387 4.077537

4 0 0 0 5.030438 4.564348 4.762174

5 1 1 0 5.117994 4.709530 5.468060

6 1 1 1 4.060443 3.806662 3.713572

7 0 0 1 0.000000 0.000000 0.000000

8 1 0 0 4.927254 4.430817 5.153292

9 1 0 0 5.337538 5.187386 5.181784

10 1 1 0 4.304065 3.583519 4.976734

> ortho.distance = daisy(Ortho.test,metric="gower",type=list(asymm=1:3))

> ortho.distance

Dissimilarities :

1 2 3 4 5 6 7 8 9

2 0.37932210

3 0.08176069 0.36432804

4 0.05708961 0.37448731 0.09221081

5 0.45626904 0.28413609 0.48853676 0.43321028

6 0.57952849 0.40082229 0.53864814 0.58475355 0.27909545

7 0.90294303 0.90072878 0.84162252 0.91078062 0.96819565 0.68910020

8 0.31127201 0.09636660 0.32474213 0.27812071 0.22874306 0.42180957 0.93309172

9 0.32032816 0.15081784 0.38164868 0.31249057 0.23651818 0.45974726 0.97861695 0.05690654

10 0.47629721 0.23718889 0.46430196 0.47242937 0.09084782 0.21772778 0.89248913 0.26210477 0.30763000

Metric : mixed ; Types = A, A, A, I, I, I

Number of objects : 10

In general, if you have *m* sets of variables of the same type, you can compute a dissimilarity matrix based on each set and then combine them into one using the formula below:

In my opinion what to use as weights is a subjective choice and an area of active research. One simply scheme would be to use weights inversely proportional to the average distance in , i.e. .

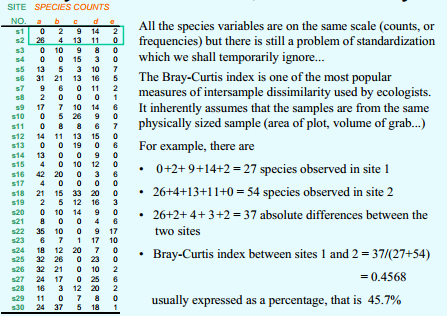
Now that we have a distance matrix representing the dissimilarities, and hence the similarities, between the observations what can we do with it?

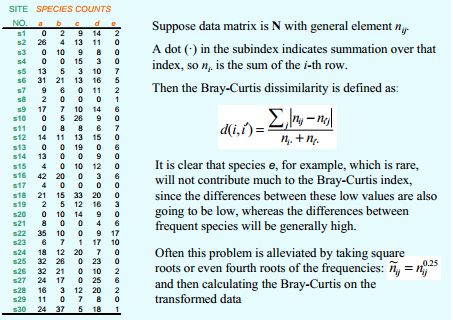
1. We can try to reproduce these distances between observa**t**ions in a lower k-dimensional space (k = 1, 2, or 3 hopefully). By doing so we can identify groups of similar observations (or clusters), look for potential multivariate outliers, and in some cases see what role the variables play in this lower-dimensional orientation or representation of the data. This is the idea behind **multidimensional scaling** or **principal coordinate analysis** (PCoA). To avoid confusion with principal component analysis (PCA) we will refer to this PCoA as metric multidimensional scaling (metric MDS). The goals if PCA and metric MDS are essentially the same but focuses on representing individuals in a lower dimensional space without also considering the structure in the variables.
2. Perhaps the main use of the pairwise distances between a set of *n* observations is to find groups or *clusters* of similar observations. This is idea behind **cluster analysis** which is sometimes called data segmentation in other disciplines (e.g. market segmentation is finding groups of similar customers or potential customers on the basis of their characteristics).
3. We can also use distances to find the most similar observations in problems where we are trying to estimate something. For example, we might have movie ratings from a large number of individuals and want to predict the rating a person who has not seen a particular movie might give it. We can use distances between this person and other movie watchers in our database to find say five individuals with the most similar rating profiles who have rated the movie in question, and then use their average rating for this movie as our prediction. This is the idea behind nearest neighbor regression and also the idea collaborative filtering/recommender systems which we will discuss later in the course.

**4.1.6 - Measuring Distance for Observations Based on Counts   
 (**special case of numeric variables**)**

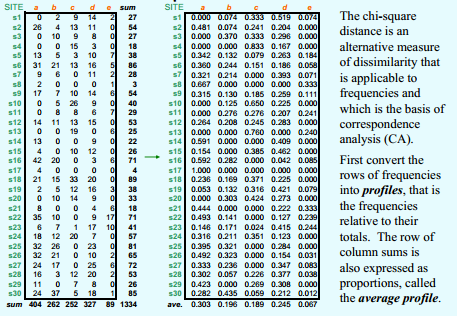
When the set of measurements for an observation represent counts in a specific category, e.g. counts for several species of interest at different sites, then we need to adjust our distance measure relative to total counts for each observation. Two measures that do this are the Bray-Curtis Index of dissimilarity and the Chi-square distance (not to be confused with Mahalanobis’ Distance which is sometimes referred to as *chi-square* distance).

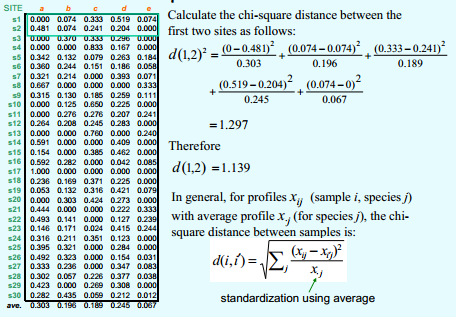
**Bray-Curtis Index**





**Chi-square Distance**





Let proportion counts for observation i in column k, then distance between observations *i* and *j* is given by:

**2.1.7 - Measuring Distance Between Variables**

Above we considered the distance between two observations in a data matrix, i.e. distance between rows. We now turn our attention to the “distance” or “dissimilarity” between two variables, i.e. column vectors in our data matrix. Assuming there are no missing values, two variables have a pair of values for each observation in our sample, and so we can measure the similarity/dissimilarity between these column vectors using a number of potential metrics that are contingent upon the data type of the columns.

**Correlation-based**

One measure of the “similarity” of two variables would be the correlation between them. There are several correlation measures that can be used depending on the data type of variables involved. If both variables *i* and *j* are continuous we could simply use the usual correlation coefficient (i.e. Pearson’s Product Moment Correlation) between them as a measure of similarity.

This measures the degree of linear association between two numeric variables. Alternatively, we could use Spearman’s Rho () which is simply the Pearson’s Product Moment Correlation between and after they have been **rank ordered**.

As the correlation is a measure of similarity we can convert it to a dissimilarity using the formula we have seen previously,

where here is one of the correlation measures discussed above.

**Distances between count variables**

We can first transpose the data matrix so the rows correspond to variables and the columns correspond to observations and then apply the Bray-Curtis or Chi-square distances discussed above for the measuring distances between observations based on count variables.

**Distances between categorical variables**When both variables are categorical/nominal we can cross-tabulate them (i.e. set up a contingency table) and then use one of the plethora of measures of association between variables summarized in this matter. For example, one measure is that works well is Cramer’s V coefficient.

This is a measure of “similarity” so we can easily convert this is to distance measure by subtracting from 1 or use .

**Measuring distance between variables with mixed types**

Measuring distances between variables that are mixture of data types is not discussed much in the literature, e.g. a continuous variable and a categorical/nominal variable.

There are numerous association/”correlation” measures in the literature for data of mixed types (i.e. similarities) and therefore we could turn those into distance measures using the methods discussed above.

**4.2 – Hierarchical Clustering**  
Starting with a dissimilarity/distance matrix containing “good” measures of pairwise distances between observations, cluster analysis seeks to form clusters/groups of observations that are similar. Ideally the clusters will have small intra-cluster distances between observations within the same cluster and have large inter-cluster distances between observations in different clusters. There are several different algorithms for forming clusters from an initial distance matrix. We will examine the main algorithms for forming clusters in the examples that follow.

We begin by examining Hierarchical Cluster Analysis using agglomerative methods. In agglomerative clustering each observation starts its own cluster and then observations that are most similar are fused together successively until all observations in the same cluster. The key part of this process is concept of measuring distances between clusters, which is called “linkage”. The example below will illustrate how hierarchical clustering works and explore different linkage options.

Example 4.4: - Simple Examples (“by hand”)

Example 4.5: Illicit drug unking in the U.S.  
These data, which look at illicit drug use and binge drinking for the 50 states, come from the U.S. Census Bureau.

The variables in the data frame IllDrug (read in **IllDrug.csv**):

* DrugUse = estimate of the % of the population in the state that use illicit drugs.
* BingeDrink = estimate of the % of the population in the state that binge drink.
* Poverty = estimate of the % of the population in the state living in poverty.
* HSdrop = estimate of the % of the population that has dropped out of high school.
* Income = estimate of the per capita income ($).

> attach(IllDrug)

> names(IllDrug)

[1] "State" "DrugUse" "BingeDrink" "Poverty" "HSdrop" "Income"   
  
> head(IllDrug)

State DrugUse BingeDrink Poverty HSdrop Income

Alabama Alabama 3.3 15.6 14.5 12.6 22946

Alaska Alaska 8.5 19.8 9.4 10.9 28523

Arizona Arizona 5.4 17.4 16.6 14.4 25307

Arkansas Arkansas 2.8 16.8 14.8 11.4 22114

California California 6.2 16.7 15.4 14.2 29819

Colorado Colorado 6.5 19.8 9.2 9.8 31678

> X <- as.matrix(IllDrug[,2:6])

> head(X)

DrugUse BingeDrink Poverty HSdrop Income

Alabama 3.3 15.6 14.5 12.6 22946

Alaska 8.5 19.8 9.4 10.9 28523

Arizona 5.4 17.4 16.6 14.4 25307

Arkansas 2.8 16.8 14.8 11.4 22114

California 6.2 16.7 15.4 14.2 29819

Colorado 6.5 19.8 9.2 9.8 31678

If we use the Euclidean metric (or any other for that matter) to construct the distances between the states using the raw data what will happen?

What do we need to do?

As with PCA and ICA if the variables are on vastly different scales, as is the case with income in this example, we need to standardize the data first!

> X <- scale(X)

> head(X)

DrugUse BingeDrink Poverty HSdrop Income

Alabama -0.9150748 -1.2127521 0.6098588 0.9105382 -1.0569412

Alaska 3.0829126 0.4407902 -0.8767792 0.2063128 0.2459105

Arizona 0.6994970 -0.5040911 1.2220039 1.6561886 -0.5053843

Arkansas -1.2994966 -0.7403114 0.6973081 0.4134379 -1.2513061

California 1.3145720 -0.7796815 0.8722067 1.5733385 0.5486712

Colorado 1.5452251 0.4407902 -0.9350788 -0.2493624 0.9829551

To perform *hierarchical cluster analysis* in R we can use the hclust function which is a base function in R. To use hclust we first need to decide two things:

1) What metric will be used to measure distance between observations?

2) What method will be used to measure distance between clusters?

The clusters can differ greatly depending on which metric is used to measure distance and to a greater degree which linkage is used when clustering.

The dist function in R allows you to choose the following for distance metrics:

'euclidean': Usual square distance between the two vectors   
 (L2 norm).

'maximum': Maximum distance between two components of x and y

(supremum norm).

'manhattan': Absolute distance between the two vectors (1 norm).

'canberra': sum(|x\_i - y\_i| / |x\_i + y\_i|). Terms with zero

numerator and denominator are omitted from the sum and

treated as if the values were missing.

'binary': (aka \_asymmetric binary\_): The vectors are regarded as

binary bits, so non-zero elements are 'on' and zero elements

are 'off'. The distance is the \_proportion\_ of bits in which

only one is on amongst those in which at least one is on.

'minkowski': The p norm, the pth root of the sum of the pth powers

of the differences of the components.

If we have loaded the proxy library we have many more choices for measuring distance between observations as we have seen previously.

The hclust function in R allows you to choose from the following linkages:

method: the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of '"ward"', '"single"', '"complete"', '"average"', '"mcquitty"', '"median"' or

'"centroid"'.

How do you decide which distance metric and linkage to use? There is no easy/standard answer! Usually we try different combinations and choose which is “best” subjectively. For some methods of clustering we will examine there are more objective measures for the “goodness” of the clustering achieved.

We begin by setting up some distance matrices using different metrics for the standardized numeric variables in our data matrix .

> dman <- dist(X,method="manhattan")

> dcan <- dist(X,method="canberra")

> de <- dist(X,method="euclidean")

> dmax <- dist(X,method="max")

Using Euclidean perform clustering using different linkages...  
  
> desing <- hclust(de,method=”sing”)

> decomp <- hclust(de,method=”comp”)

> deave <- hclust(de,method=”ave”)

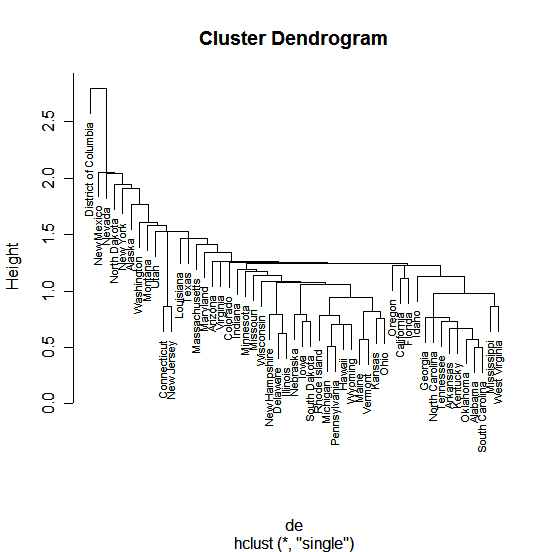
> deward <- hclust(de,method=”ward.D”)

> demcq <- hclust(de,method=”mcqu”)

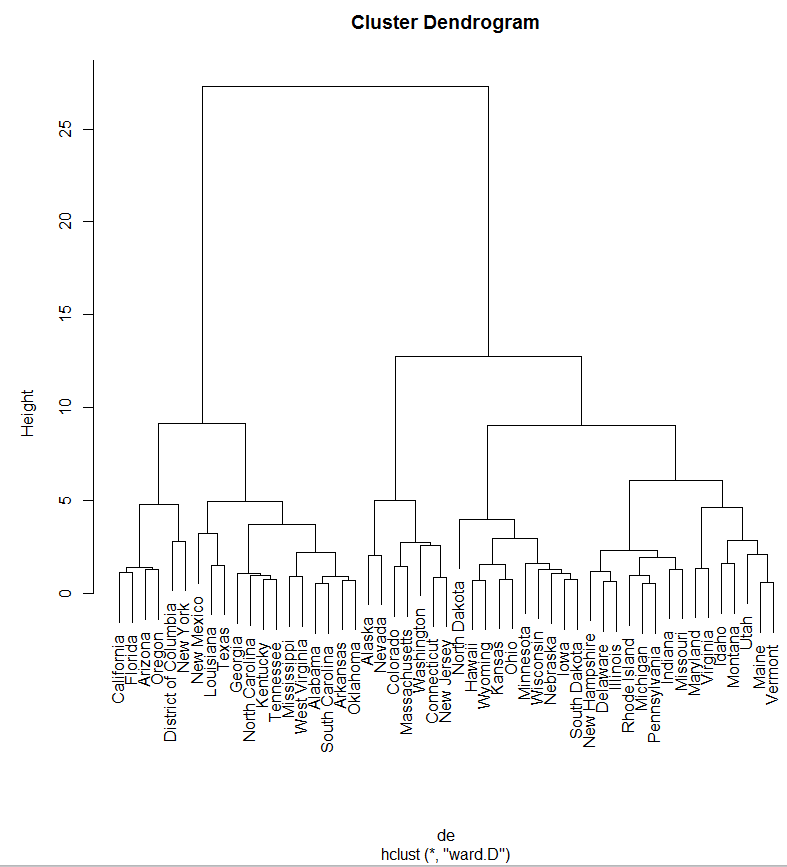
Plot cluster dendograms/trees...

> state <- IllDrug$State 🡨 Extract the state names from the IllDrug data frame.

> plot(desing,labels=state,cex=.7)



> plot(deward,labels=state,cex=.7)

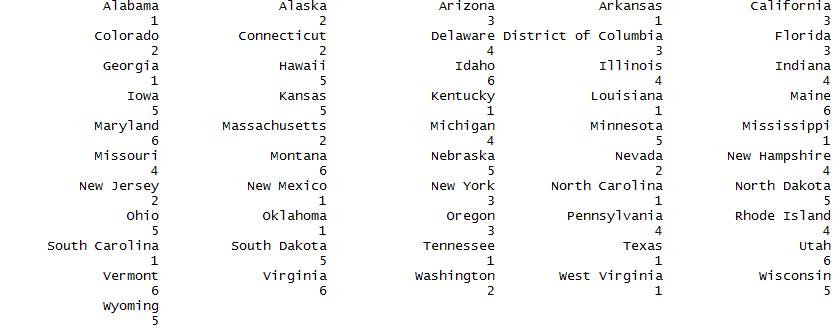


How are clusters formed? To form clusters we need to cut the tree at a height of our choosing, which then determines our clusters by looking at observations that are on the same branch. You can form any number of clusters you want by cutting the tree at the appropriate height. The function cutree allows you to either cut the tree at a certain height to determine the clusters or to specify the number of clusters you want and the height required for that many clusters will be used. The latter approach is generally easier.

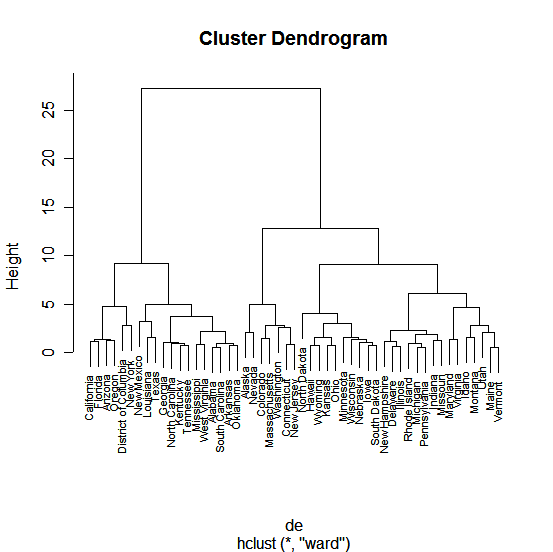
We now consider using the tree from the Ward’s linkage to form clusters for these data.

> drug.clust <- cutree(deward,k=6)

> drug.clust



Below you can see where the Ward’s linkage dendogram is cut in order form groups.



Cutting the tree here will produce the six clusters above.

Which states are in each cluster? The function clust.grps (code shown below) the results from cutting the tree and the original data matrix as arguments and prints the clusters.  
  
> library(MASS)  
> X.unscaled = as.matrix(IllDrug[,2:6])

> clust.grps(X.unscaled,drug.clust,parcoord=T)

Cluster 1 consists of:

=========================================================

Alabama Arkansas Georgia Kentucky Louisiana Mississippi New Mexico North Carolina Oklahoma South Carolina Tennessee Texas West Virginia

Variables means in this cluster are:

DrugUse BingeDrink Poverty HSdrop Income

3.507692 16.869231 15.530769 12.246154 23560.846154

Cluster 2 consists of:

=========================================================

Alaska Colorado Connecticut Massachusetts Nevada New Jersey Washington

Variables means in this cluster are:

DrugUse BingeDrink Poverty HSdrop Income

6.585714 19.285714 9.271429 10.514286 33121.857143

Cluster 3 consists of:

=========================================================

Arizona California District of Columbia Florida New York Oregon

Variables means in this cluster are:

DrugUse BingeDrink Poverty HSdrop Income

5.416667 17.100000 16.516667 13.083333 30409.666667

Cluster 4 consists of:

=========================================================

Delaware Illinois Indiana Michigan Missouri New Hampshire Pennsylvania Rhode Island

Variables means in this cluster are:

DrugUse BingeDrink Poverty HSdrop Income

4.7875 19.7250 10.4000 10.4250 28923.3750

Cluster 5 consists of:

=========================================================

Hawaii Iowa Kansas Minnesota Nebraska North Dakota Ohio South Dakota Wisconsin Wyoming

Variables means in this cluster are:

DrugUse BingeDrink Poverty HSdrop Income

3.98 21.90 10.88 7.14 26738.20

Cluster 6 consists of:

=========================================================

Idaho Maine Maryland Montana Utah Vermont Virginia

Variables means in this cluster are:

DrugUse BingeDrink Poverty HSdrop Income

3.814286 17.000000 10.700000 9.200000 25945.285714

The code for the clust.grps function is given below:

clust.grps = function(X,grps,parcoord=F,suppress=F) {

k = length(unique(grps))

p = dim(X)[[2]]

Xmeans = matrix(0,nrow=length(unique(grps)),ncol=p+1)

X = as.data.frame(X)

for (i in 1:k){

cat("\n")

cat(paste("Cluster",i,"\n"))

cat("=======================================================================\n")

if (suppress==F){

cat(row.names(X)[grps==i])

cat("\n\n")}

cat("Variable means in this cluster are:\n")

cat("----------------------------------------------------------------------\n")

print(apply(X[grps==i,],2,mean))

Xmeans[i,]=c(apply(X[grps==i,],2,mean),as.numeric(i))

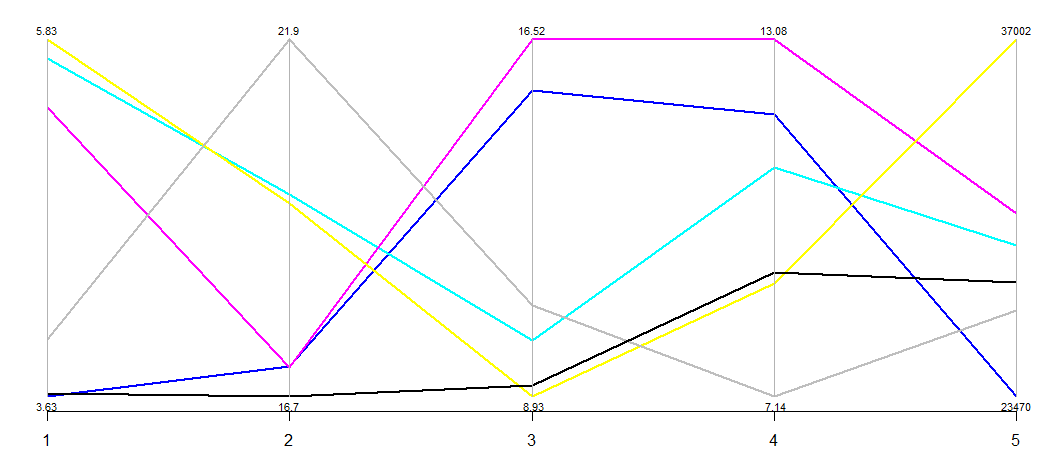
cat("\n\n")

}

if (parcoord) {parcoord(Xmeans[,-(p+1)],col=as.numeric(Xmeans[,(p+1)])+3,

lwd=2,var.label=T)}

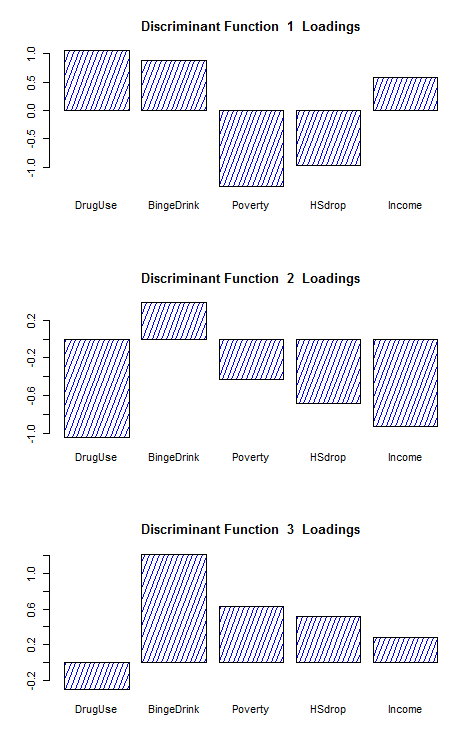
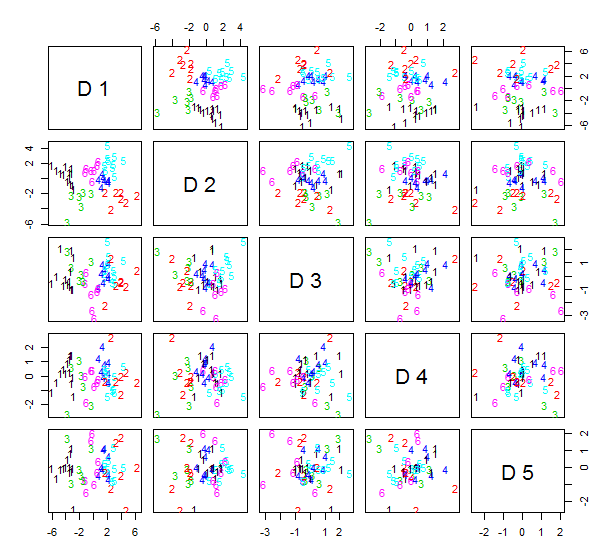
}

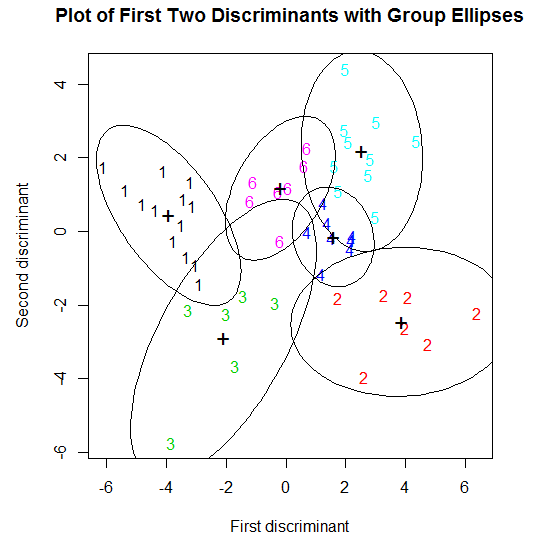
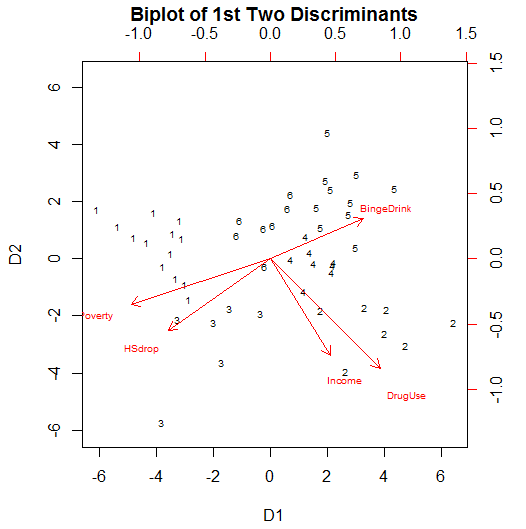


The parallel coordinate plots the mean for each variable for each cluster. The clusters are color coded, but not labelled. Thus the plot is simply a visualization of the means reported for each of the clusters in the first part of the output. This can aid in identifying what makes the clusters distinct.

**4.2 – Exploring Clusters**

I have written a function called Discrim that performs a discriminant analysis using the clusters returned from a cluster analysis. We will discuss discriminant analysis later in the course, however given what we know about principal components we should be able to make sense of the results of this function as they are interpreted similarly. The Discrim function uses *linear discriminant analysis* to try to discriminate between cluster groups. Linear discriminant analysis finds linear combinations of the variables (think PCA) to try to maximally separate the identified groups (clusters here) in a data set. The results of this function give us some idea about how distinct our clusters are and what defining characteristics the clusters have.   
  
> Discrim(X,drug.clust)

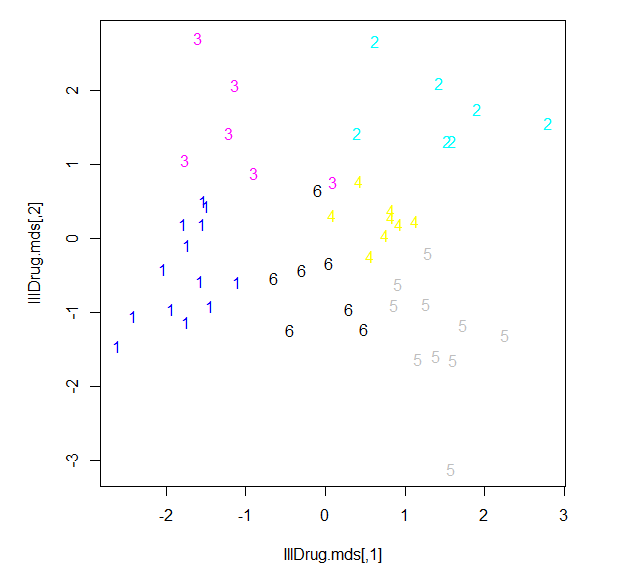
We can also label points in a multidimensional scaling of these data according to cluster membership.  
  
> library(MASS)

> X.dist = dist(X)

> IllDrug.mds = cmdscale(X.dist,k=2)

> plot(IllDrug.mds,type="n")

> text(IllDrug.mds,labels=as.character(drug.clust),col=as.numeric(drug.clust))



> IllDrug.iso = isoMDS(X.dist,k=2)

initial value 20.918363

iter 5 value 15.176204

iter 5 value 15.164617

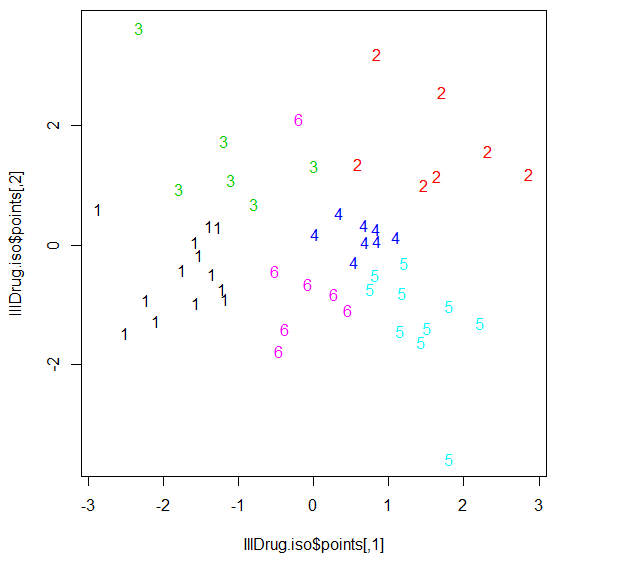
iter 5 value 15.164617

final value 15.164617

converged

> plot(IllDrug.iso$points,type="n")

> text(IllDrug.iso$points,labels=as.character(drug.clust),col=as.numeric(drug.clust))

> library(rgl)  
> IllDrug.iso = isoMDS(X.dist,k=3)

initial value 10.263657

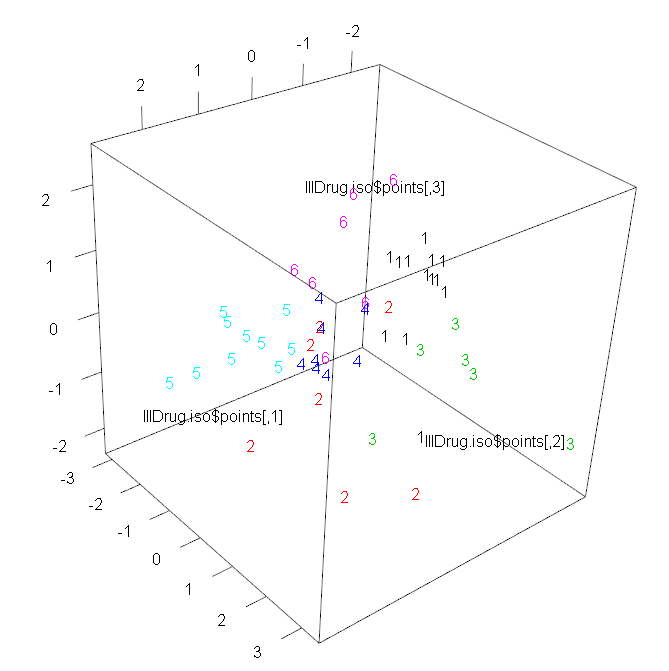
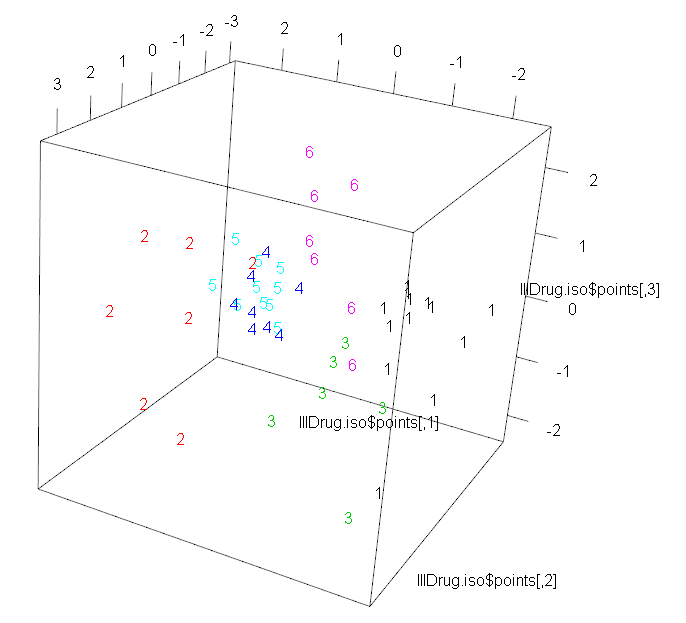
iter 5 value 7.096606

final value 6.988238

converged

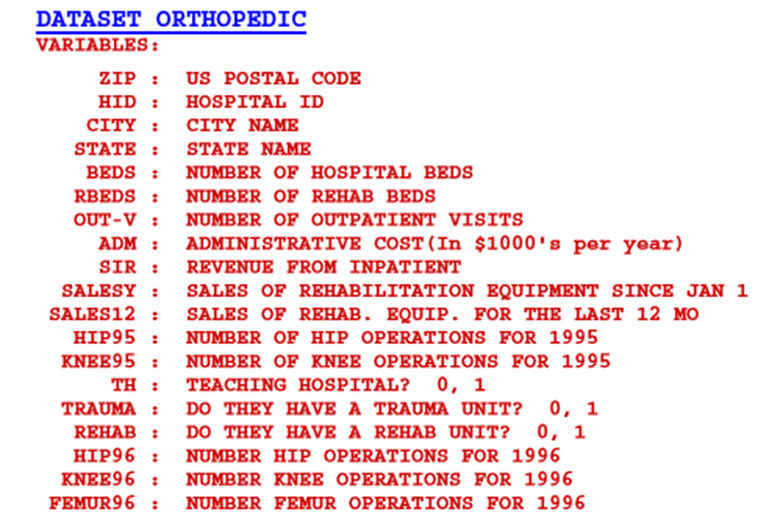
> plot3d(IllDrug.iso$points,type="n")

> text3d(IllDrug.iso$points,textss=as.character(drug.clust),col=as.numeric(drug.clust))



Example 4.6: Orthopedic Equipment Sales

These data were used in the multidimensional scaling notes (Section 3). Variable descriptions for these data are given below.



All of the numeric variables have been log transformed. For those with zeroes, was used.

> names(Orthopedic)

[1] "ZIP" "HospID" "City" "State" "Beds" "RBeds" "Outpatients"

[8] "Admin" "Inpatient" "Hip95" "Knee95" "SalesYr" "Sales12" "Teach"

[15] "Trauma" "Rehab" "Hip96" "Knee96" "Femur96" "logBeds" "logRBeds1"

[22] "logOut1" "logAdmin" "logInpat1" "lnHip951" "lnKnee951" "lnSales1" "lnSales121"

[29] "lnHip961" "lnKnee961" "lnFem961"

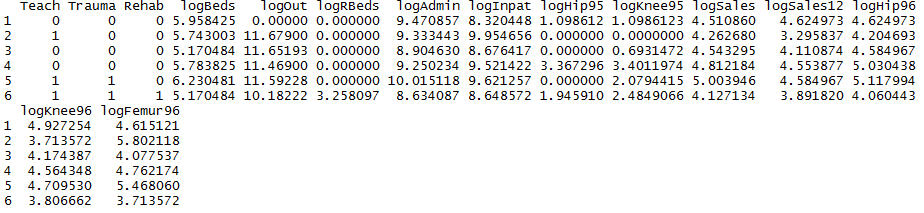
> ortho.mat = Orthopedic[,c(14:16,20:31)]

> names(ortho.mat)

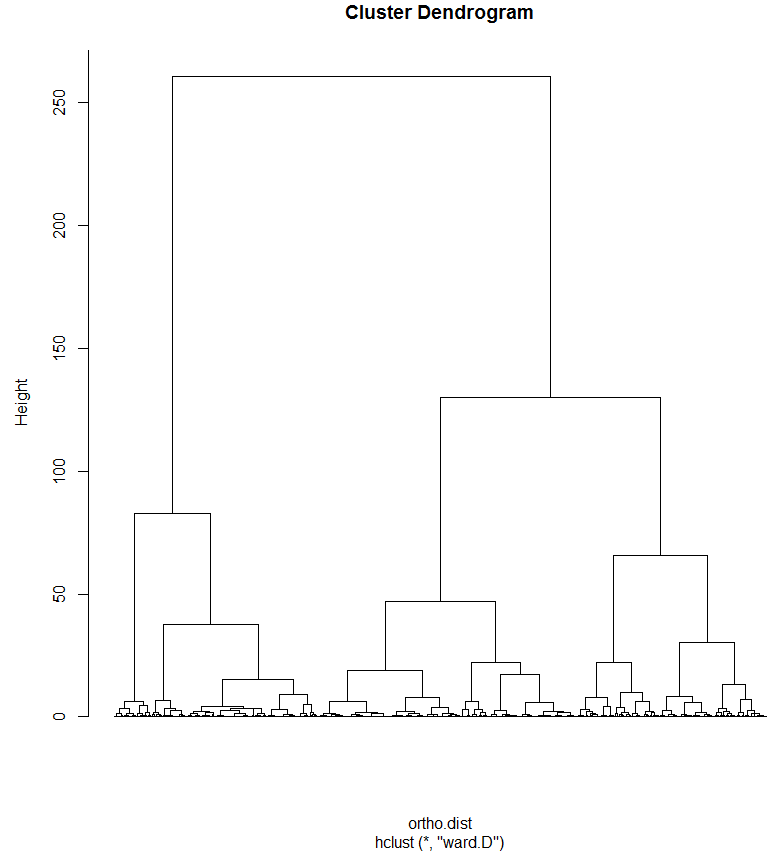
[1] "Teach" "Trauma" "Rehab" "logBeds" "logRBeds1" "logOut1" "logAdmin"

[8] "logInpat1" "lnHip951" "lnKnee951" "lnSales1" "lnSales121" "lnHip961" "lnKnee961"

[15] "lnFem961"

> head(ortho.mat)  


> ortho.dist = daisy(ortho.mat,type=list(asymm=c(1:3))) 🡨 Use Gower’s metric

> ortho.clust = hclust(ortho.dist,method=”ward.D”)  
> plot(ortho.clust,labels=FALSE,hang=-1)  
> grps = cutree(ortho.clust,k=10)  


> clust.grps(ortho.mat,grps,parcoord=T)

Cluster 1 consists of

=======================================================================

1 12 16 37 39 65 71 83 94 106 107 136 161 163 167 175 179 184 191 195 198 200 201 207 208 211 233 240 270 274 283 295 313 344 348 356 374 375 380 402 415 416 431 443 459 466 468 487 490 492 497 499 519 520 523 532 533 536 554 560 563 575 581 584 598 600 606 616 624 626 640 642 649 651 652 655 658 659 661 668 672 692 702 721 722 724 725 726 736 743 744 746 763 768 769 777 796 808 831 833 844 848 857 863 876 879 882 886 890 900 912 953 971 982 996 1007 1075 1082 1083 1111 1122 1127 1128 1186 1209 1234 1254 1276 1281 1290 1298 1308 1318 1321 1324 1330 1333 1335 1336 1356 1358 1371 1375 1380 1397 1409 1418 1437 1438 1441 1462 1495 1502 1512 1527 1532 1533 1554 1572 1581 1582 1586 1599 1612 1623 1630 1638 1640 1649 1652 1654 1655 1677 1687 1688 1689 1702 1703 1709 1713 1734 1737 1751 1754 1756 1758 1762 1763 1765 1767 1768 1770 1781 1782 1797 1801 1803 1804 1809 1810 1817 1818 1820 1825 1832 1837 1842 1843 1846 1848 1849 1891 1916 1919 1921 1922 1932 1982 2001 2013 2015 2054 2062 2074 2075 2078 2106 2110 2113 2123 2134 2150 2152 2184 2192 2205 2206 2209 2219 2224 2237 2253 2262 2263 2265 2268 2271 2275 2285 2295 2305 2308 2310 2311 2317 2318 2331 2349 2353 2359 2378 2393 2395 2415 2417 2420 2431 2433 2436 2441 2458 2459 2462 2471 2489 2490 2491 2499 2502 2505 2508 2510 2514 2515 2516 2520 2527 2529 2533 2543 2544 2547 2550 2551 2552 2553 2554 2563 2567 2568 2569 2583 2591 2593 2598 2614 2624 2629 2630 2636 2646 2656 2667 2668 2682 2684 2695 2697 2702 2707 2716 2720 2722 2733 2735 2736 2748 2770 2772 2783 2793 2795 2818 2819 2820 2825 2826 2827 2828 2834 2843 2860 2874 2883 2894 2895 2915 2922 2934 2935 2937 2940 2946 2947 2948 2954 2957 2964 2970 2979 2980 2981 2983 2984 2985 2987 2989 2990 2992 3005 3011 3012 3015 3018 3024 3028 3032 3035 3042 3047 3048 3051 3052 3053 3067 3069 3078 3087 3089 3090 3103 3105 3109 3115 3118 3120 3143 3150 3160 3165 3167 3168 3176 3180 3189 3195 3196 3197 3202 3207 3209 3211 3213 3215 3217 3219 3224 3231 3232 3237 3241 3245 3257 3260 3261 3268 3280 3282 3286 3288 3290 3295 3298 3299 3300 3305 3306 3320 3324 3339 3340 3347 3353 3367 3371 3377 3396 3397 3399 3401 3404 3413 3414 3435 3449 3482 3486 3489 3492 3494 3499 3504 3506 3510 3511 3513 3527 3529 3557 3558 3563 3584 3621 3649 3665 3678 3687 3716 3727 3748 3751 3752 3753 3754 3755 3772 3779 3780 3782 3787 3792 3793 3795 3796 3802 3808 3826 3846 3847 3875 3895 3896 3899 3904 3906 3908 3911 3913 3920 3924 3928 3932 3933 3937 3945 3946 3949 3956 3983 3984 3987 3988 3997 4002 4008 4009 4020 4056 4059 4061 4071 4078 4086 4088 4090 4111 4113 4115 4121 4124 4126 4128 4138 4152 4158 4166 4169 4177 4188 4191 4193 4205 4207 4209 4215 4220 4228 4233 4235 4237 4244 4252 4253 4254 4256 4257 4260 4263 4268 4269 4270 4273 4287 4288 4289 4298 4302 4306 4312 4317 4327 4330 4337 4338 4343 4346 4348 4357 4358 4363 4364 4366 4368 4376 4389 4395 4398 4404 4416 4417 4421 4430 4431 4434 4438 4443 4446 4448 4454 4459 4468 4469 4472 4473 4478 4482 4483 4486 4487 4488 4489 4490 4494 4495 4496 4499 4512 4516 4520 4524 4525 4526 4527 4533 4534 4536 4538 4540 4541 4543 4544 4552 4553 4573 4577 4579 4580 4589 4590 4592 4596 4605 4608 4610 4611 4621 4622 4624 4631 4634 4637 4638 4639 4643 4644 4646 4648 4649 4650 4652 4655 4661 4662 4668 4672 4673 4677 4678 4679 4681 4683 4686 4688 4689 4691 4696 4701

Variable means in this cluster are:

--------------------------------------------------

Teach Trauma Rehab logBeds logRBeds1 logOut1 logAdmin logInpat1 lnHip951 lnKnee951 lnSales1 lnSales121

0.0000000 0.0000000 0.0000000 2.0438232 0.0000000 3.2514276 3.5835799 3.4662752 0.9311017 1.0941619 1.4747648 1.2495588

lnHip961 lnKnee961 lnFem961

1.4940390 1.2796823 1.5363210

Cluster 2 consists of

=======================================================================

2 8 9 11 14 20 22 27 29 30 38 40 48 50 57 62 67 68 76 77 90 92 93 96 99 100 101 102 105 114 115 120 124 127 128 130 132 134 140 141 143 145 148 150 151 152 153 158 159 169 174 177 178 181 182 183 192 193 194 196 197 202 209 215 227 232 234 236 243 254 260 261 265 266 267 273 287 288 296 309 310 314 316 320 321 322 324 325 330 331 338 343 347 353 358 361 362 365 366 369 370 376 377 378 385 387 392 393 404 417 418 424 430 438 440 444 448 449 453 456 465 470 480 494 495 513 518 538 546 558 561 574 585 588 591 601 603 610 623 629 641 708 710 712 715 718 728 729 731 733 753 759 764 766 775 780 783 784 786 789 801 809 810 813 816 841 849 860 866 869 871 903 904 907 910 918 924 932 933 950 958 959 967 968 973 976 977 985 986 988 989 991 992 1001 1006 1009 1011 1012 1014 1019 1020 1021 1026 1031 1042 1045 1056 1059 1069 1071 1086 1096 1098 1112 1115 1117 1120 1121 1159 1195 1222 1227 1239 1241 1264 1282 1295 1296 1303 1304 1309 1312 1315 1317 1323 1325 1326 1328 1329 1332 1341 1344 1363 1367 1376 1377 1379 1385 1386 1393 1399 1403 1408 1425 1427 1428 1459 1467 1471 1475 1513 1520 1523 1535 1546 1555 1563 1564 1565 1566 1568 1571 1573 1580 1587 1593 1596 1604 1607 1619 1663 1666 1674 1678 1682 1686 1704 1722 1730 1749 1759 1769 1775 1778 1786 1787 1789 1792 1794 1814 1815 1822 1826 1840 1850 1853 1855 1867 1869 1878 1882 1883 1906 1920 1927 1935 1946 1950 1958 1959 1966 1967 1974 1976 1978 1989 1990 1995 1996 2004 2006 2009 2010 2023 2025 2032 2049 2056 2065 2069 2072 2077 2079 2082 2085 2091 2096 2102 2103 2114 2117 2118 2119 2128 2136 2148 2156 2171 2173 2174 2194 2197 2204 2216 2227 2233 2236 2238 2242 2246 2247 2251 2255 2297 2302 2315 2320 2326 2327 2334 2337 2339 2340 2342 2357 2362 2363 2366 2372 2375 2379 2384 2387 2391 2404 2411 2414 2418 2426 2428 2442 2447 2467 2469 2484 2561 2572 2575 2579 2584 2585 2587 2601 2602 2605 2606 2610 2616 2619 2660 2662 2670 2699 2704 2705 2734 2749 2750 2784 2790 2792 2833 2853 2875 2898 2920 2931 2945 2974 3008 3009 3027 3046 3060 3061 3063 3107 3124 3125 3141 3172 3187 3203 3206 3210 3216 3239 3240 3243 3247 3248 3249 3283 3333 3334 3335 3351 3352 3355 3376 3430 3437 3444 3445 3448 3461 3472 3473 3474 3481 3487 3496 3501 3508 3617 3639 3646 3653 3654 3659 3664 3674 3683 3697 3703 3710 3744 3745 3801 3843 3858 3859 3871 3909 3910 3912 3919 3921 3951 3957 3965 3968 4025 4051 4098 4104 4105 4116 4123 4125 4144 4146 4176 4203 4204 4222 4230 4266 4322 4353 4354 4405 4406 4408 4435 4437 4515 4531 4563 4567 4582 4595 4623 4628 4636 4641 4647 4651 4669 4682 4687

Variable means in this cluster are:

--------------------------------------------------

Teach Trauma Rehab logBeds logRBeds1 logOut1 logAdmin logInpat1 lnHip951 lnKnee951 lnSales1 lnSales121

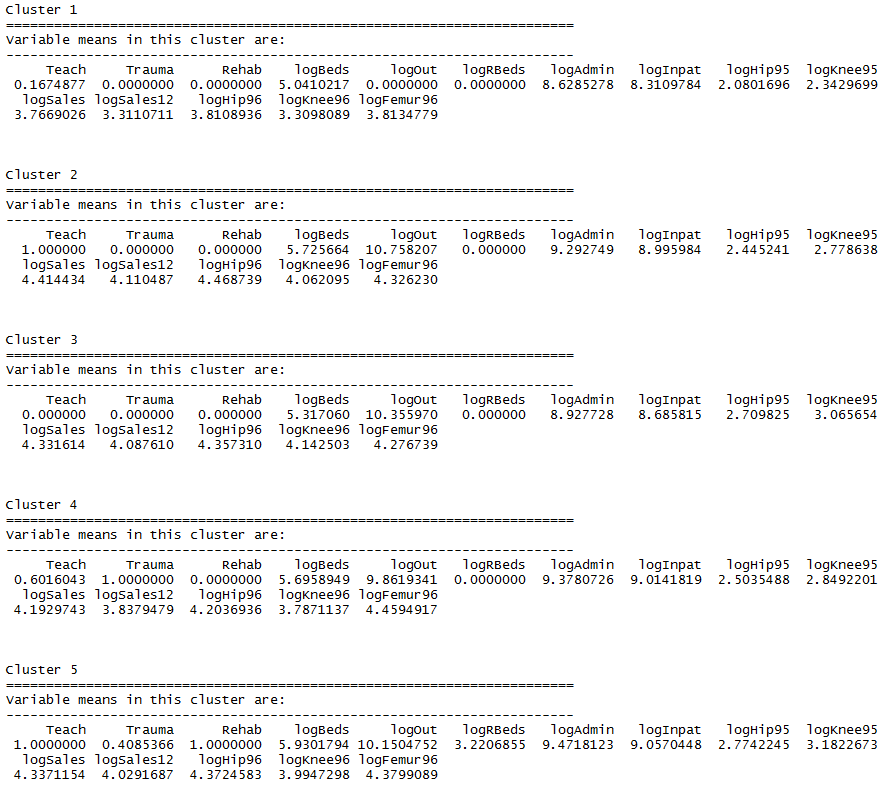
1.0000000 0.0000000 0.0000000 2.4429718 0.0000000 4.0783103 3.9752120 3.8077641 0.8711817 1.0005358 1.7632446 1.5987789

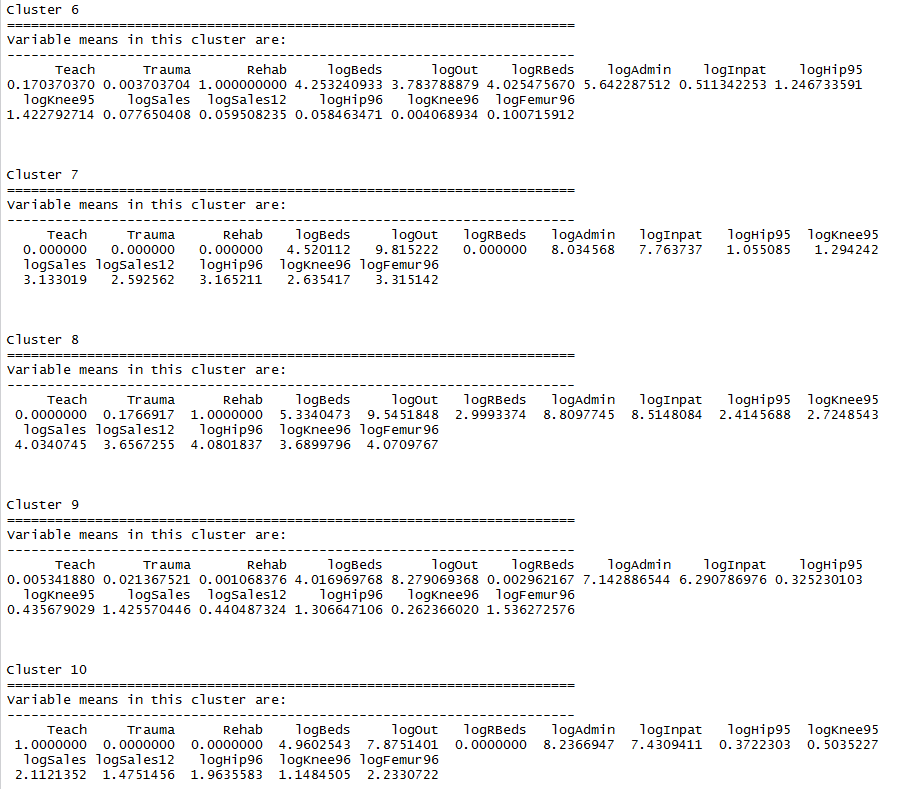
lnHip961 lnKnee961 lnFem961

1.7760076 1.5775709 1.7550238

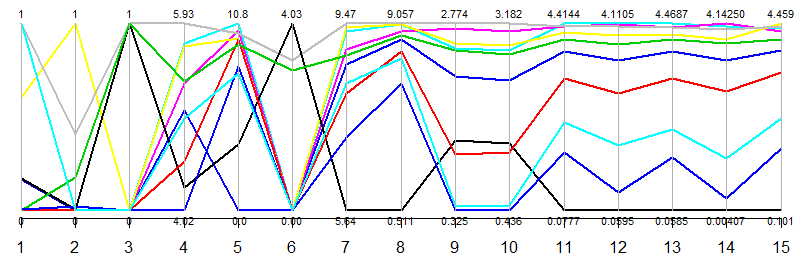
...   
YIKES! Too many observation labels to possibly digest! We can use the suppress=T option to suppress the observation labels from the cluster report.

> clust.grps(ortho.mat,grps,parcoord=T,suppress=T)





The parallel coordinate plot shows the mean variable values for each cluster.

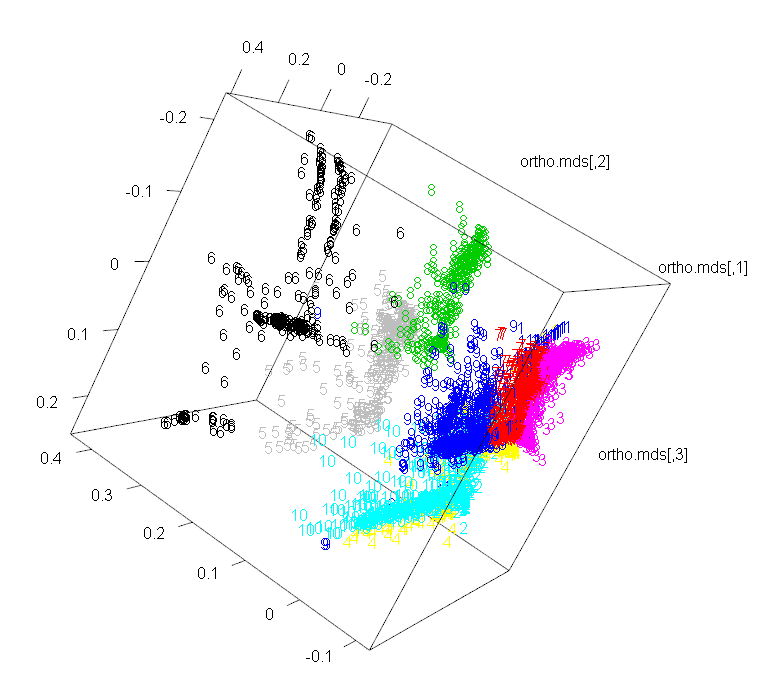
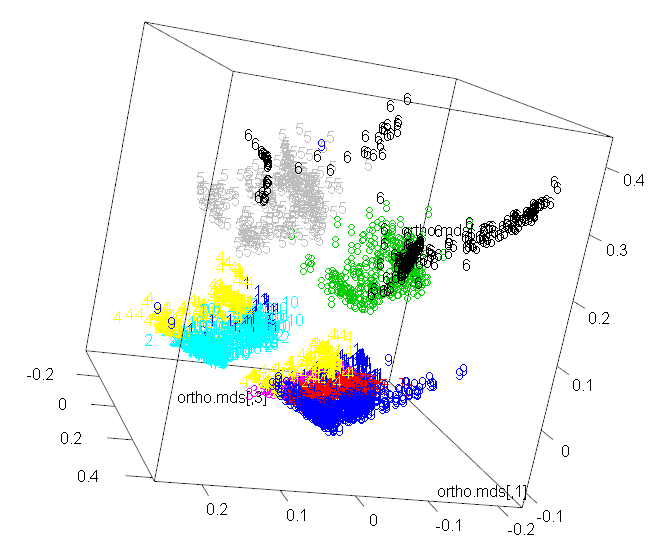


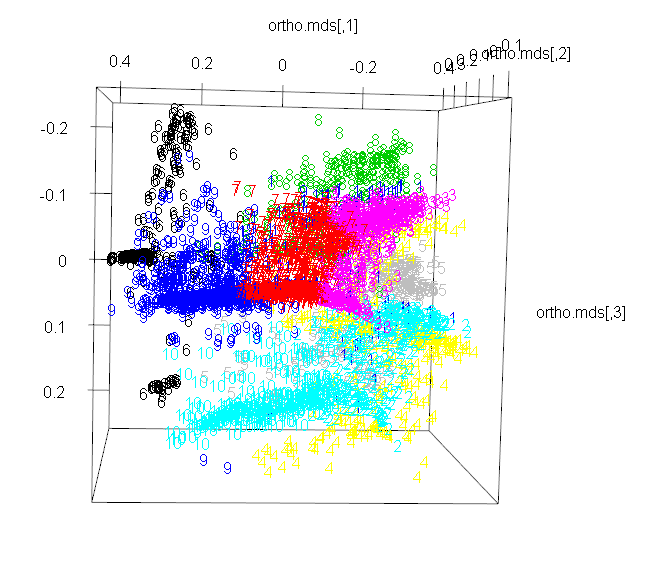
Again we use multidimensional scaling to view clusters.

> ortho.mds = cmdscale(ortho.dist,k=3)

> plot3d(ortho.mds,type="n")

> text3d(ortho.mds,texts=as.character(grps),col=as.numeric(grps)+3)





**4.3 – The cluster Library and Other Hierarchical Clustering Methods**  
Example 5.2: Illicit drug use in the U.S. (cont’d)

The library cluster has more advanced methods of performing cluster analysis as well functions for assessing the “goodness” of the clustering.

Agglomerative Clustering (agnes) – alternative to hclust

> names(IllDrug)

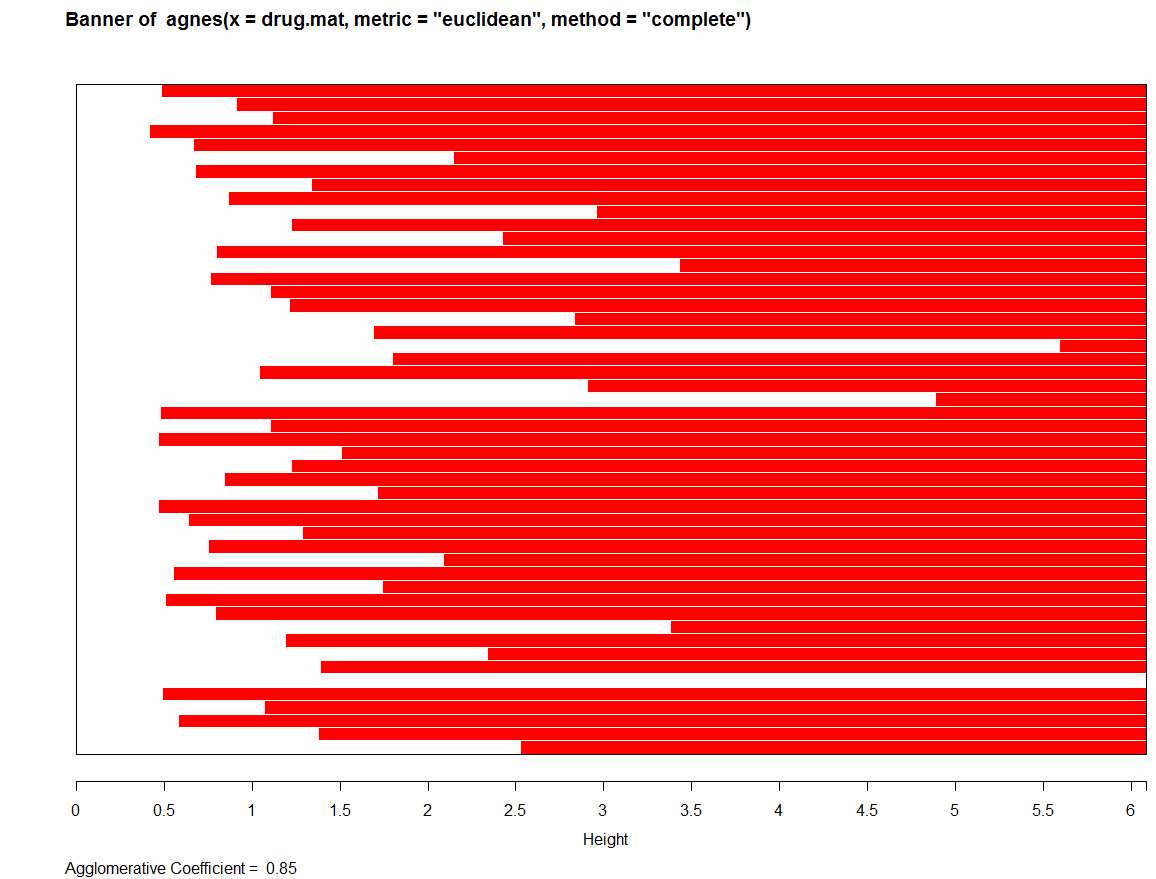
[1] "State" "DrugUse" "BingeDrink" "Poverty" "HSdrop" "Income"

> drug.mat = IllDrug[,2:5]

> drug.mat = scale(drug.mat)

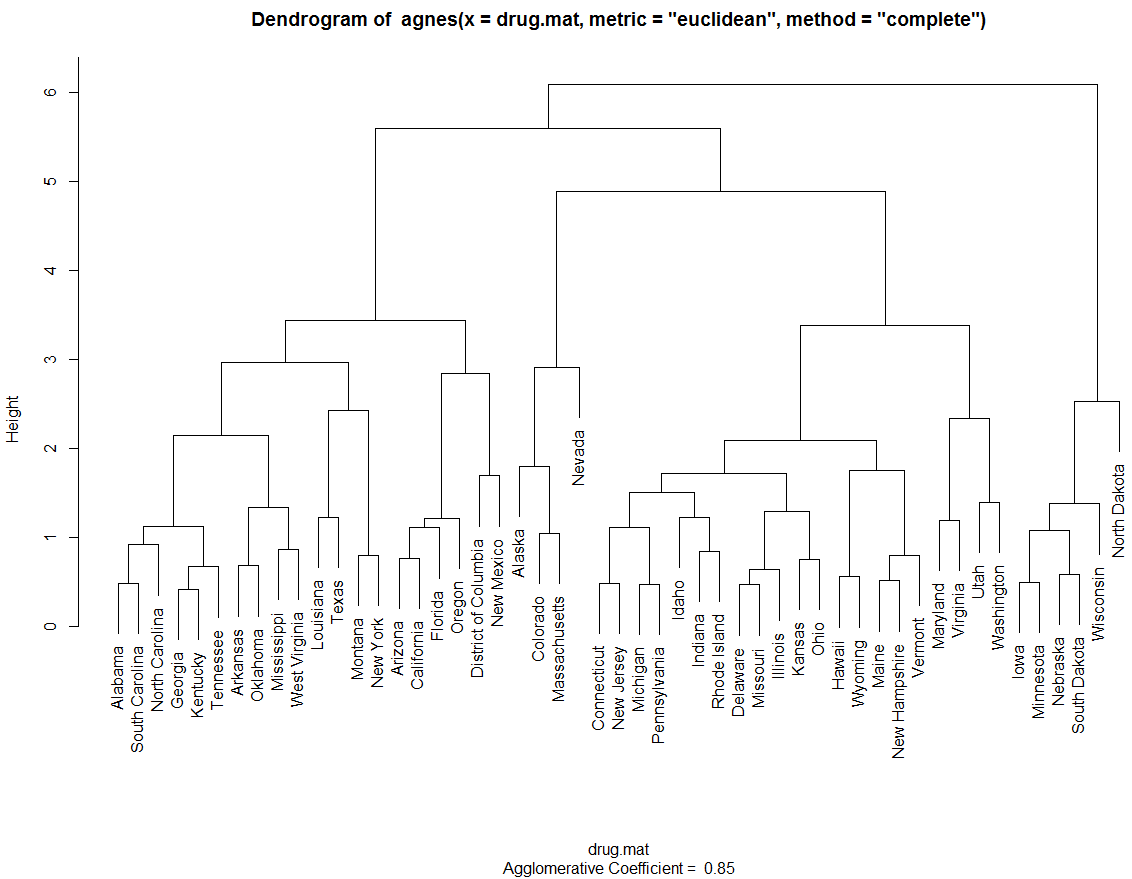
> drug.agnes = agnes(drug.mat,metric="euclidean",method="complete")

> plot(drug.agnes)



The *agglomerative coefficient* () measures the amount of clustering structure found. For each object *i* in the data set denote by its dissimilarity to the first cluster it is merged with, divided by the dissimilarity of the merger in the last step of the algorithm. The is then defined as the average of all . The closer the is to 1, the better the clustering structure found.

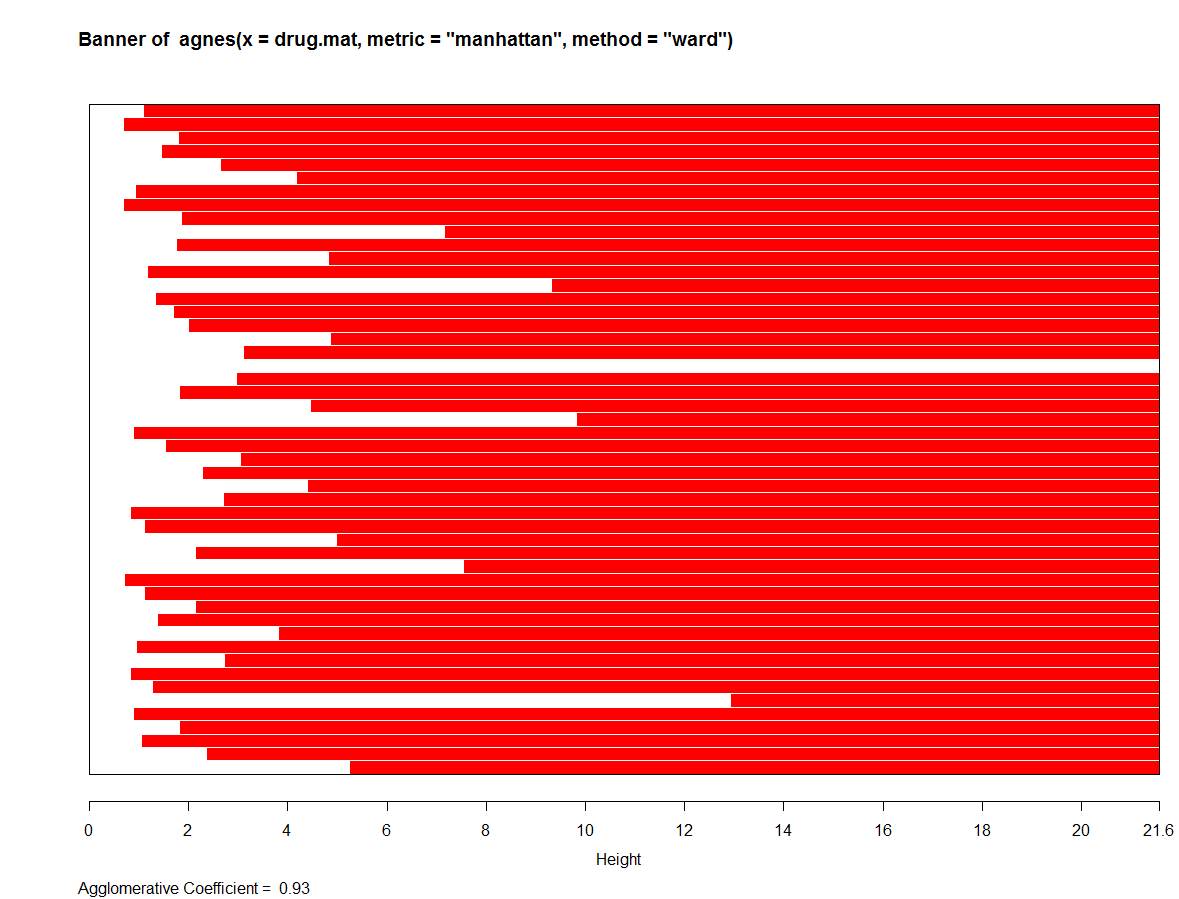
The banner plot is an alternative representation of the hierarchical cluster tree. The white bars are centered between observations that are joined together. The bar with the smallest height shows the two observations that were fused first. Subsequent bars show either pairs of observations that are merged to form new clusters or how observations are fused to existing clusters.

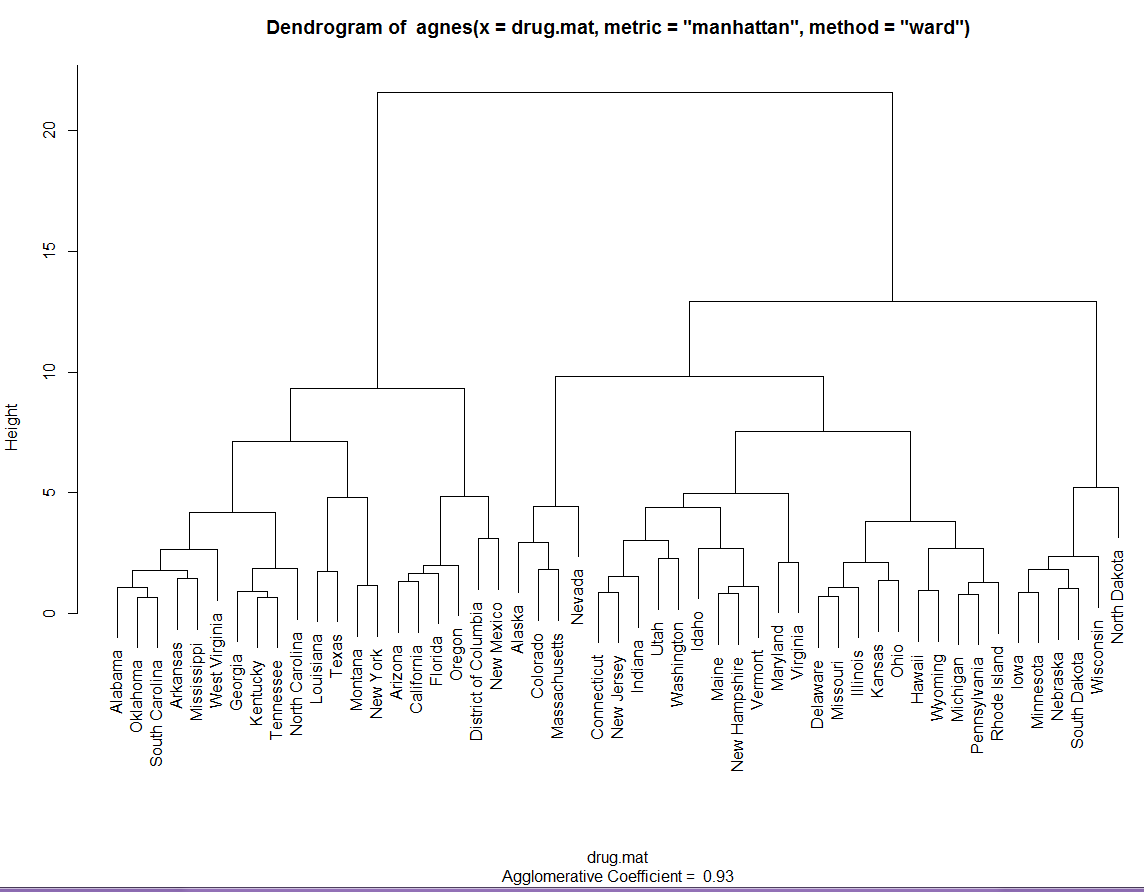


Let’s try a different metric and linkage.

> drug.agnes = agnes(drug.mat,metric="manhattan",method="ward")

> plot(drug.agnes)





This choice of metric & linkage appears to produce better clustering. As the result of agnes is a hierarchical cluster tree (dendrogram) we can use the cutree function to produce clusters.

> clust.grps(drug.mat,grps,parcoord=T)

Cluster 1 consists of

=======================================================================

Alabama Arkansas Georgia Kentucky Louisiana Mississippi Montana New York North Carolina Oklahoma South Carolina Tennessee Texas West Virginia

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

-0.7777813 -0.6559470 0.8493033 0.5880148

Cluster 2 consists of

=======================================================================

Alaska Colorado Massachusetts Nevada

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

2.2371845 0.7852782 -0.8549169 0.2891629

Cluster 3 consists of

=======================================================================

Arizona California District of Columbia Florida New Mexico Oregon

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

0.8917080 -0.5828312 1.3920442 1.2350342

Cluster 4 consists of

=======================================================================

Connecticut Idaho Indiana Maine Maryland New Hampshire New Jersey Utah Vermont Virginia Washington

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

-0.1252626 -0.6150431 -0.8476295 -0.3058510

Cluster 5 consists of

=======================================================================

Delaware Hawaii Illinois Kansas Michigan Missouri Ohio Pennsylvania Rhode Island Wyoming

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

0.2228139 0.6218925 -0.5182371 -0.3902074

Cluster 6 consists of

=======================================================================

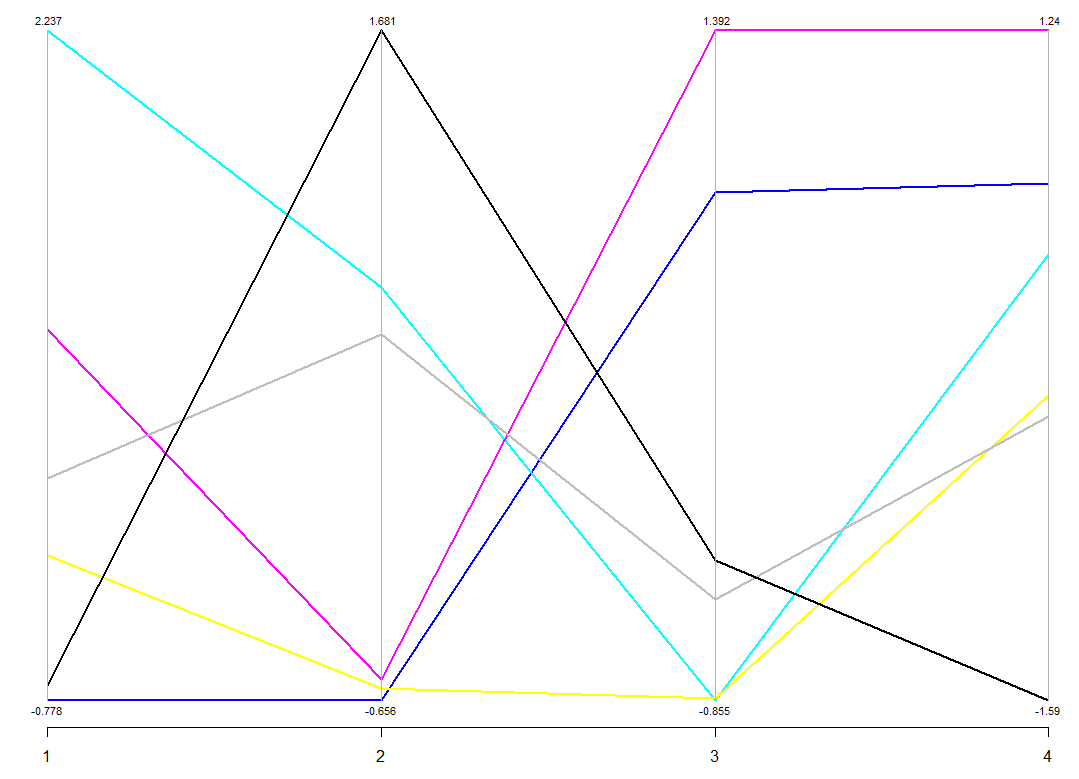
Iowa Minnesota Nebraska North Dakota South Dakota Wisconsin

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

-0.7100498 1.6809469 -0.3860915 -1.5887714



We can use boxplots to examine cluster differences on the individual variables.

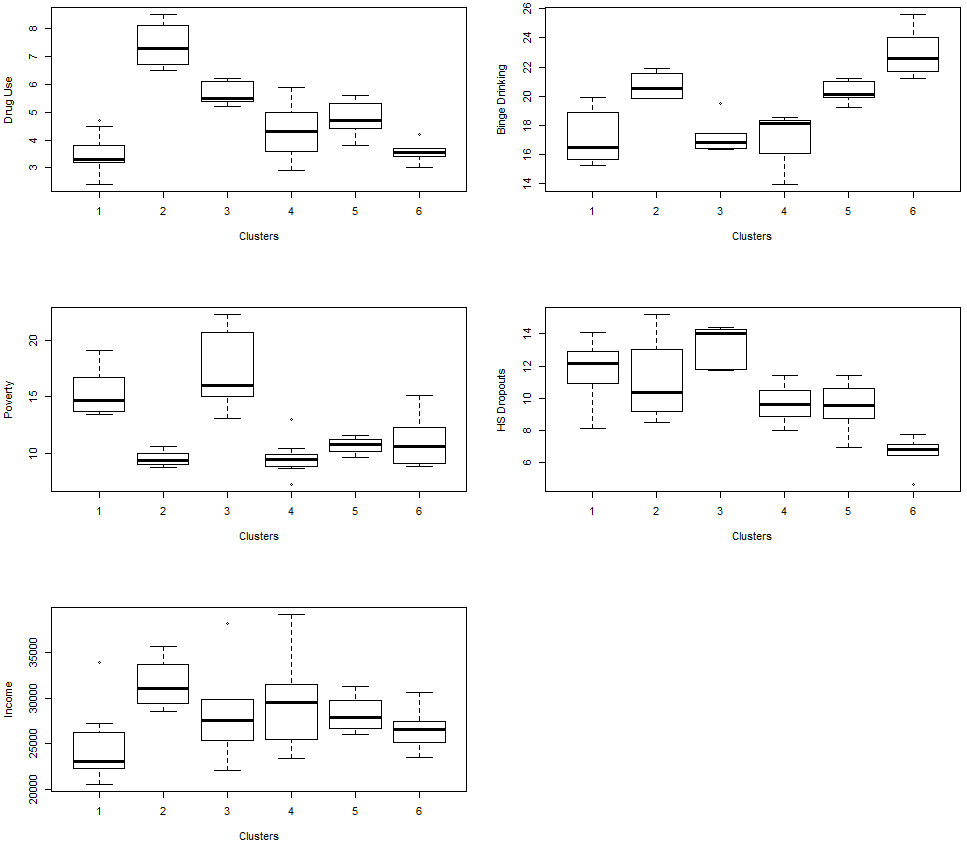
> par(mfrow=c(3,2))  
> boxplot(split(DrugUse,grps),xlab="Clusters",ylab="Drug Use")

> boxplot(split(BingeDrink,grps),xlab="Clusters",ylab="Binge Drinking")

> boxplot(split(Poverty,grps),xlab="Clusters",ylab="Poverty")

> boxplot(split(HSdrop,grps),xlab="Clusters",ylab="HS Dropouts")

> boxplot(split(Income,grps),xlab="Clusters",ylab="Income")



Divisive Hierarchical Clustering  
Divisive hierarchical clustering works in the opposite direction of agglomerative hierarchical clustering, as initially all objects to be clustered start in one large cluster and subsequent clusters are formed successively by splitting the objects into homogeneous groups. At the end of this process all objects are in their own group or cluster.

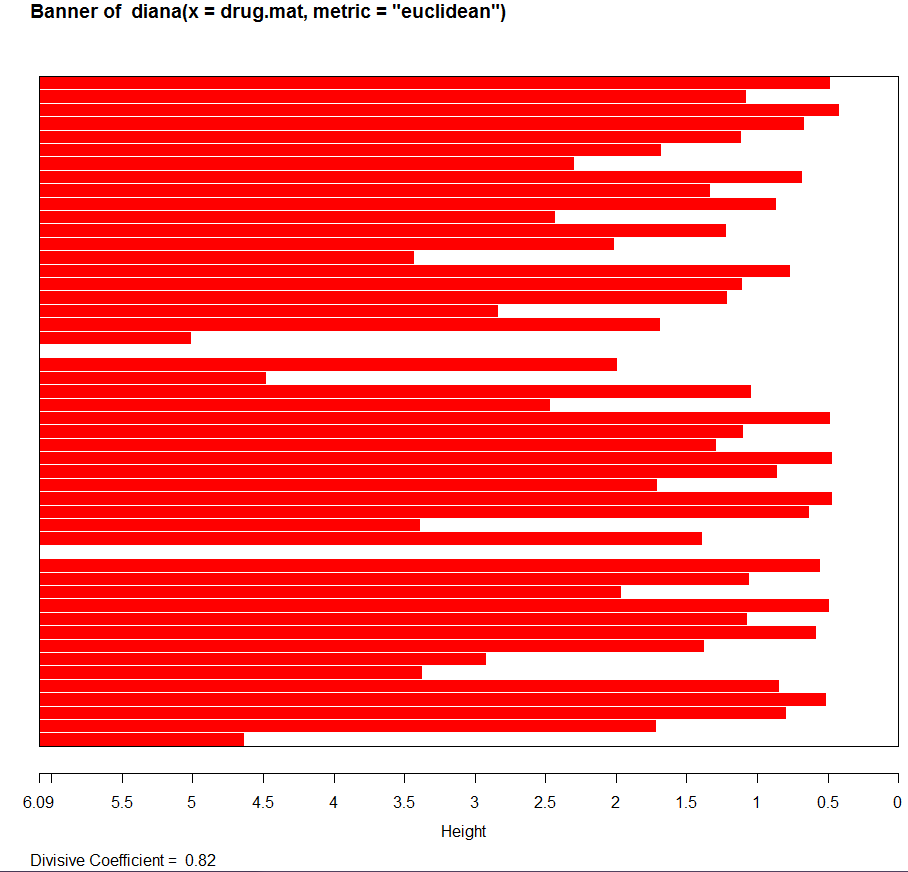
The function diana in the cluster library is a divisive hierarchical method. The initial cluster (at step 0) consists of one cluster containing all objects. In each subsequent step, the largest available cluster is split into two smaller clusters, until finally all clusters contain but a single object. The largest cluster is determined not by counting the number of observations in it but by computing the diameter of the clusters,

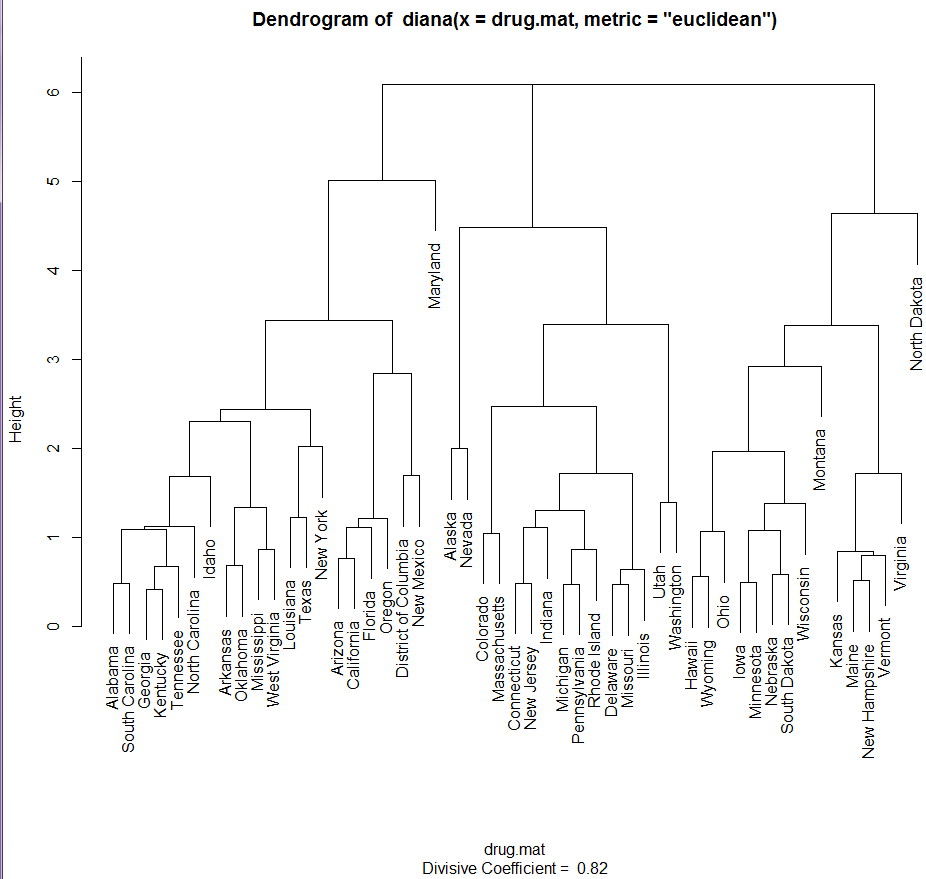
When a cluster is split into two clusters and it is done in such a way that the average dissimilarity of the two clusters is optimized.

Example 5.2: Illicit drug use in the U.S. (cont’d)

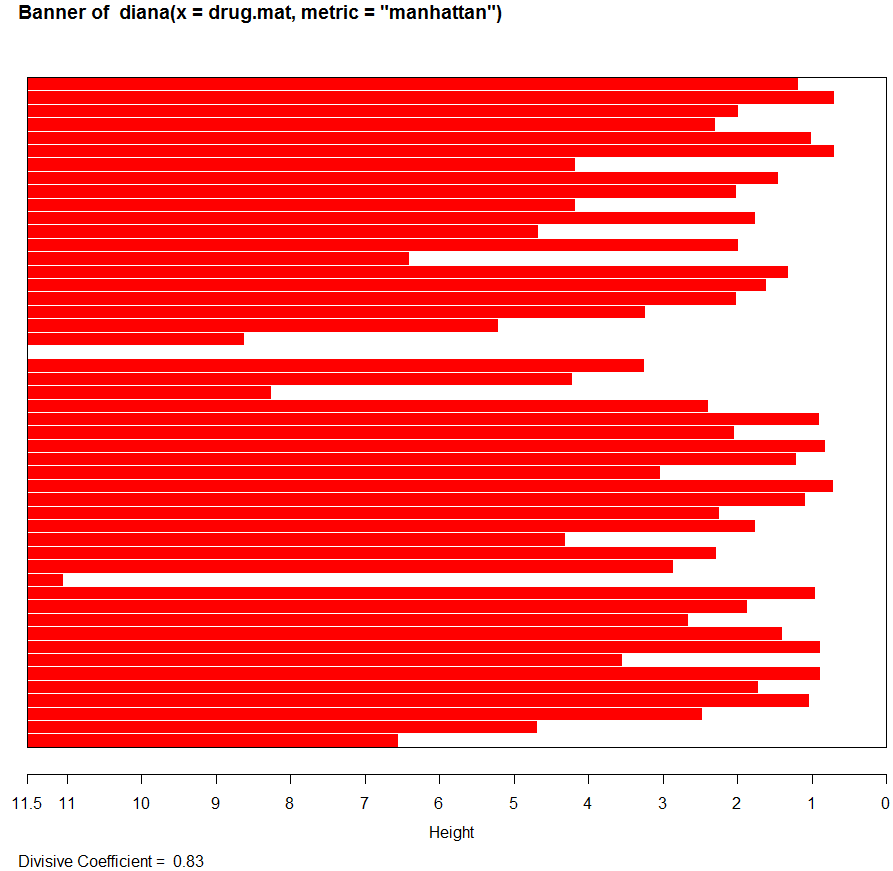
> drug.mat = Illdrug[,-1] 🡨 remove the first column containing the state names.

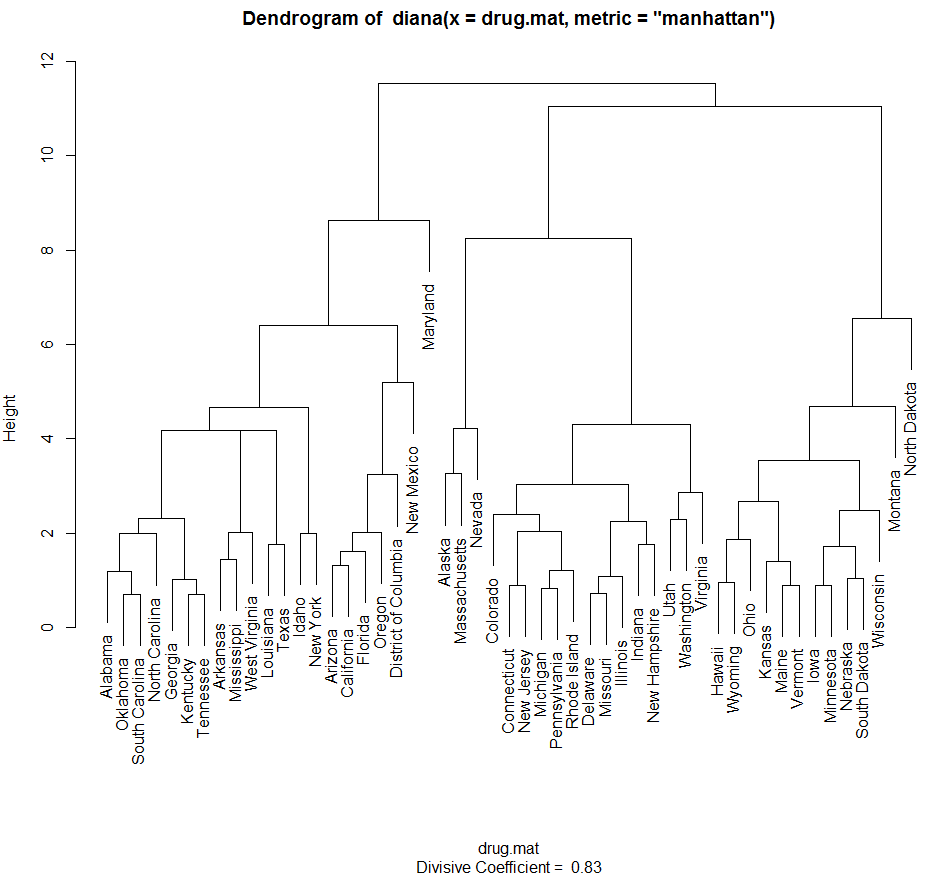
> drug.diana = diana(drug.mat,metric=”euclidean”,stand=T)





> drug.diana = diana(drug.mat,metric=”manhattan”,stand=T)  
> plot(drug.diana)





> grps = cutree(drug.diana,k=6)

> clust.grps(drug.mat,grps)

Cluster 1 consists of

=======================================================================

Alabama Arizona Arkansas California District of Columbia Florida Georgia Idaho Kentucky Louisiana Mississippi New Mexico New York North Carolina Oklahoma Oregon South Carolina Tennessee Texas West Virginia

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

-0.2846229 -0.6655083 0.9596560 0.8297594

Cluster 2 consists of

=======================================================================

Alaska Massachusetts Nevada

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

2.4678376 0.9001075 -0.8281963 0.4686713

Cluster 3 consists of

=======================================================================

Colorado Connecticut Delaware Illinois Indiana Michigan Missouri New Hampshire New Jersey Pennsylvania Rhode Island Utah Virginia Washington

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

0.32056691 -0.01477756 -0.76018016 -0.13396411

Cluster 4 consists of

=======================================================================

Hawaii Iowa Kansas Maine Minnesota Montana Nebraska Ohio South Dakota Vermont Wisconsin Wyoming

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

-0.3448490 0.8312099 -0.4443910 -1.1607128

Cluster 5 consists of

=======================================================================

Maryland

Variable means in this cluster are:

-------------------------------------------------------------------------

DrugUse BingeDrink Poverty HSdrop

-0.9150748 -1.8820430 -1.5180741 0.2063128

Cluster 6 consists of

=======================================================================

North Dakota

Variable means in this cluster are:

-------------------------------------------------------------------------

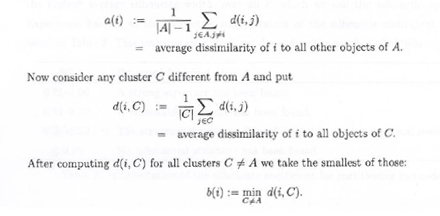
DrugUse BingeDrink Poverty HSdrop

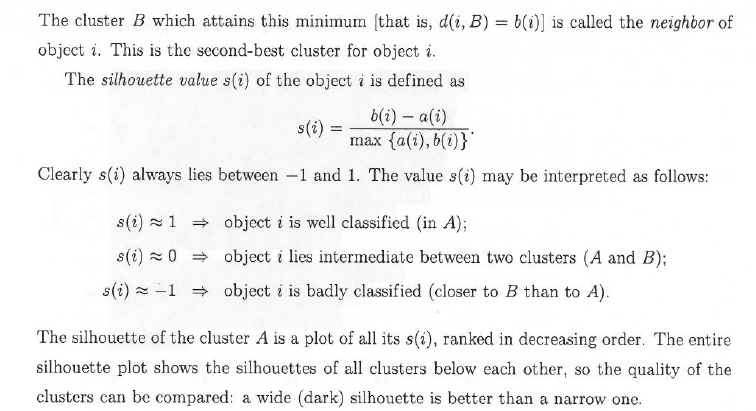
-1.1457279 2.7242534 0.7847574 -2.4034634

For this particular data set divisive clustering does not produce very satisfactory results as two of the six clusters formed contain a single observation. This seems to be the case with other data sets on which diana is used, it tends to place outliers on branches high on tree so when clusters are formed by cutting the tree they end up as singleton clusters.

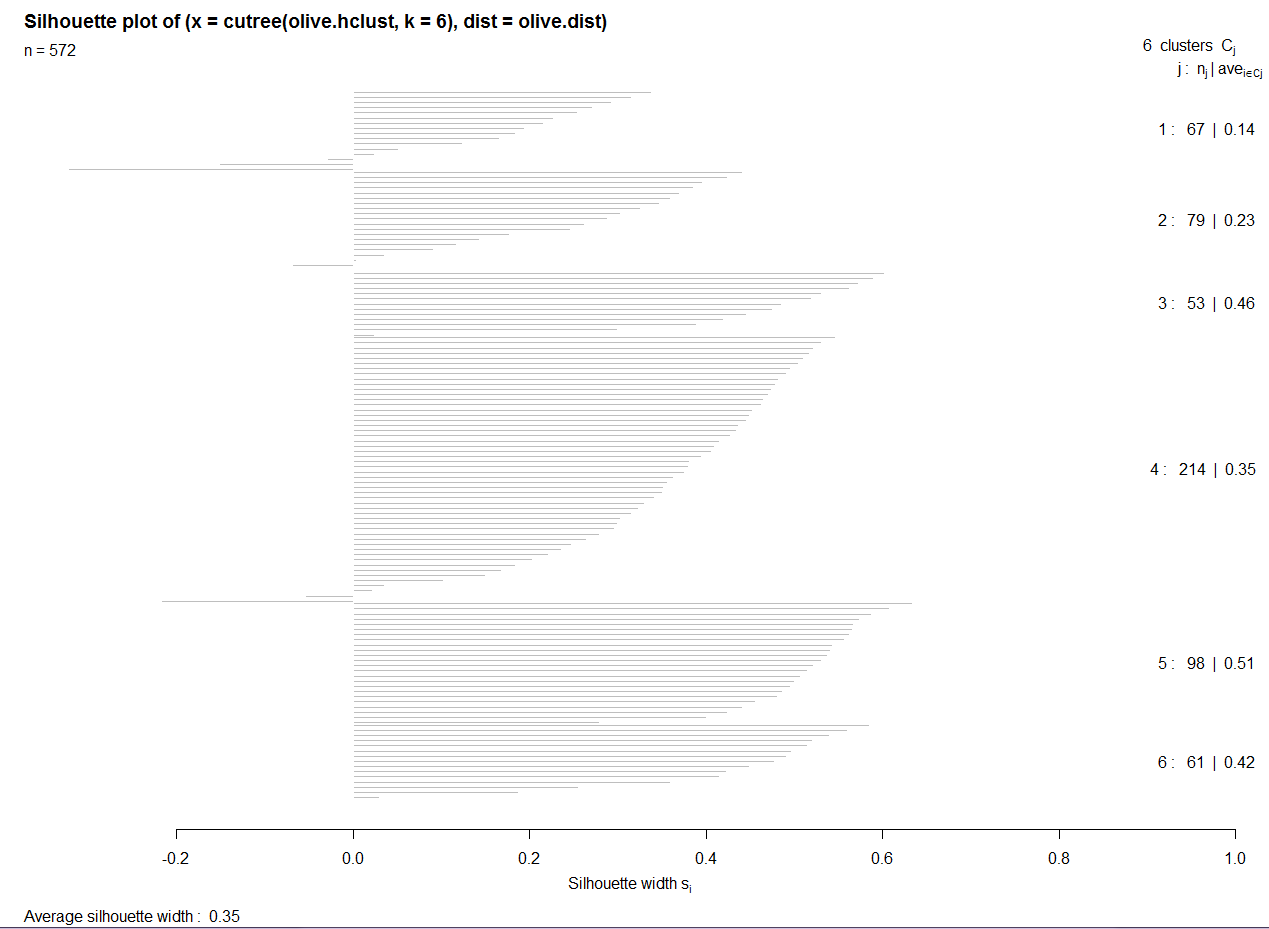
Silhouette Plots

*Silhouette Plots* are a graphical method for determining the goodness of a clustering. For each object in an arbitrary cluster denoted by *A* we define the following:

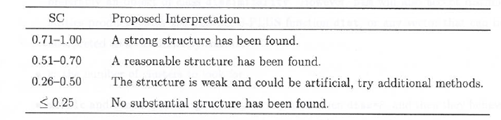




A sample silhouette plot is shown below:



In general, we can run a cluster analysis with different choices for the number of clusters *k* to compare the resulting silhouette plots. We can select the value of *k* yielding the highest average silhouette width, over all *k* clusters, which is called the *silhouette coefficient*. A subjective interpretation of the silhouette coefficient () is listed in the table below:



It is my experience that getting values greater than .50 is rare, especially for large datasets thus in my opinion the “Proposed Interpretation” needs to be taken with a *grain of salt*.

Example 4.5: Illicit drug use in U.S. (cont’d)

First we perform hierarchical cluster analysis using the taxi-cab metric and Ward’s linkage.

> drug.mat = scale(IllDrug[,-1])

> drug.dist = dist(drug.mat,method="manhattan")

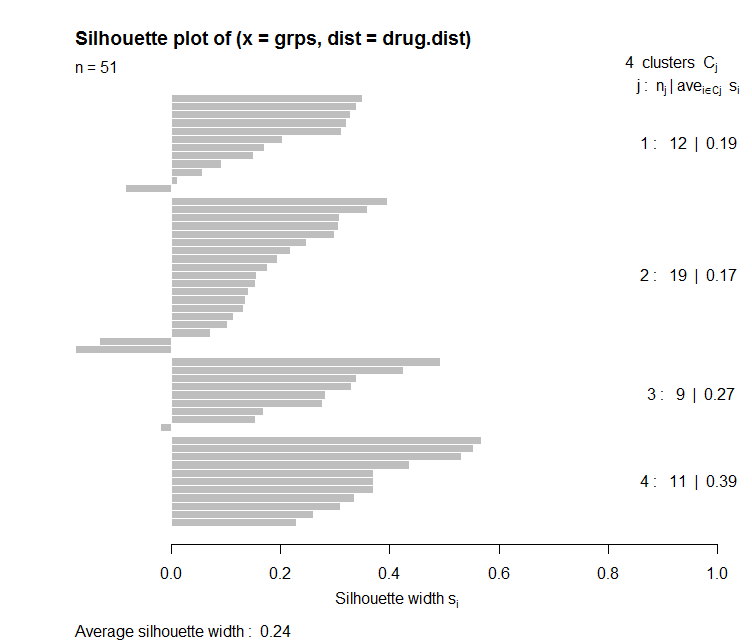
> drug.hc = hclust(drug.dist,method="ward")

> plot(drug.hc)

> grps = cutree(drug.hc,k=4)

> si = silhouette(grps,drug.dist) 🡨 give cluster memberships and pair-wise distances

> plot(si)



At this point we can try different combinations of metrics, linkages, and the number of clusters to maximize the average silhouette width. I tried several and got a maximum of around 0.29 for these data.

Example 4.2: Mushrooms  
  
> library(cba)

> names(Mushrooms.train)

[1] "Poisonous" "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8"

[10] "x9" "x10" "x11" "x12" "x13" "x14" "x15" "x16" "x17"

[19] "x18" "x19" "x20" "x21" "x22"   
  
> mush.subset = Mushrooms.train[,-c(1,12,17)] 🡨 remove mushrooms with missing   
 values & edibility.

> mush.dummy = as.dummy(mush.subset) 🡨 requires library cba

> mush.dist = dist(mush.dummy,method="jaccard")

> mush.hc = hclust(mush.dist,method="ward.D")

> mush.hc2 = hclust(mush.dist,method="ward.D2")

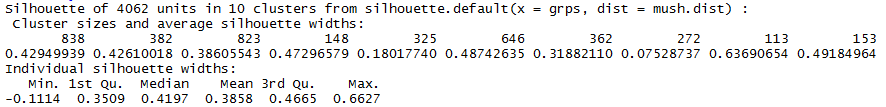
> grps = cutree(mush.hc,k=10) 🡨 we can try various choices for *k* trying

maximize the mean silhouette width.

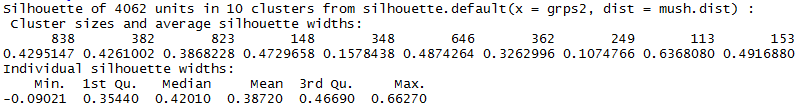
> grps2 = cutree(mush.hc2,k=10)

> si1 = silhouette(grps,mush.dist) 🡨 silhouette takes groups/clusters & distance   
 matrix used in clustering as arguments.

> si2 = silhouette(grps2,mush.dist)

> summary(si1)  


> summary(si2)



The maximum was obtained using using Ward D2 clustering, though most choices between 10 – 15 produced very similar mean silhouette widths.

> grps2.15 = cutree(mush.hc2,k=15)

> si2.15 = silhouette(grps2.15,mush.dist)

> summary(si2.15)



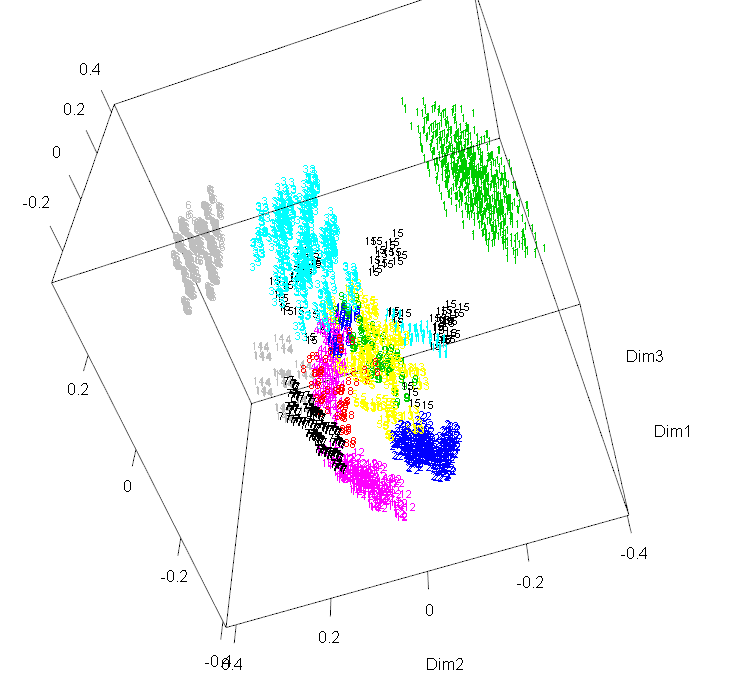
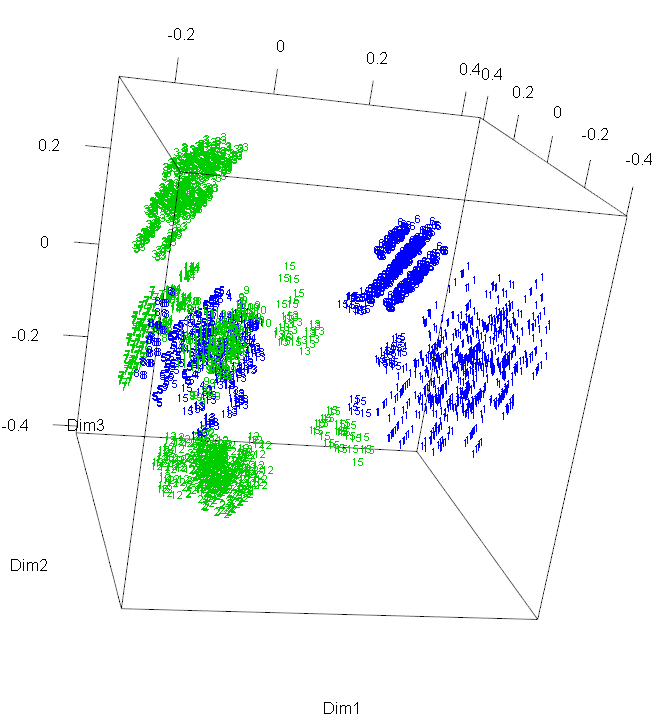
> mush.mds = cmdscale(mush.dist,k=3)

> library(rgl)

> plot3d(mush.mds,type="n",xlab="Dim1",ylab="Dim2",zlab="Dim3")

> text3d(mush.mds,texts=grps2.15,col=as.numeric(Mushrooms.train[,1])

,cex=.6)



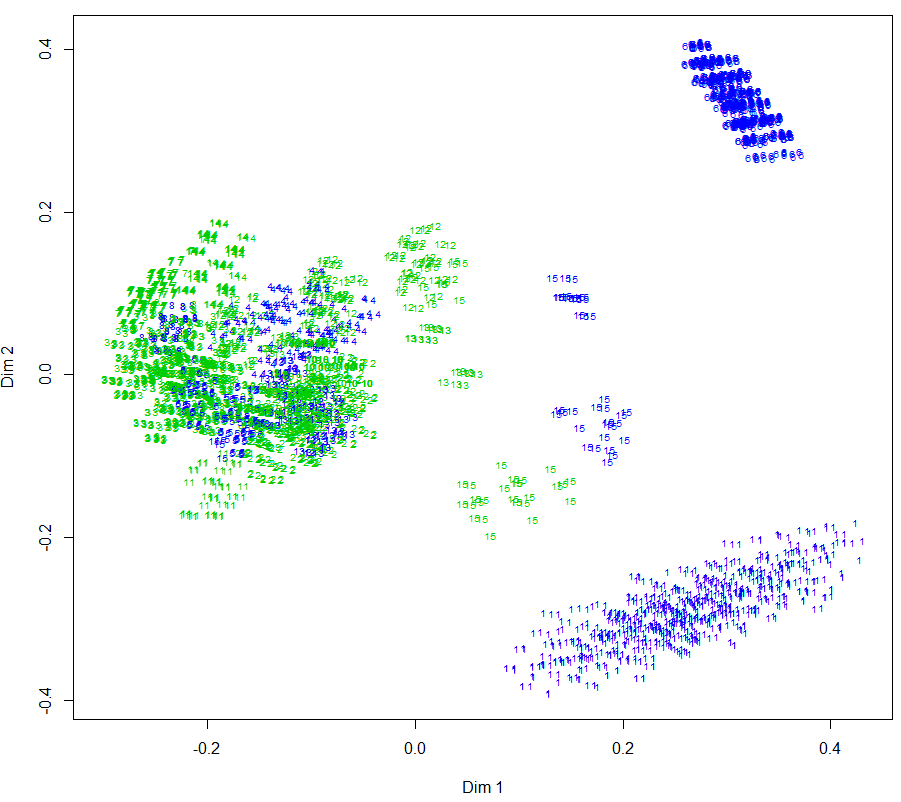
We would also use cluster number here, i.e. grps2.15.

We can also create some 2-D plots for the pairs of dimensions returned my MDS.

> plot(mush.mds[,1:2],type="n",xlab="Dim 1",ylab="Dim 2")

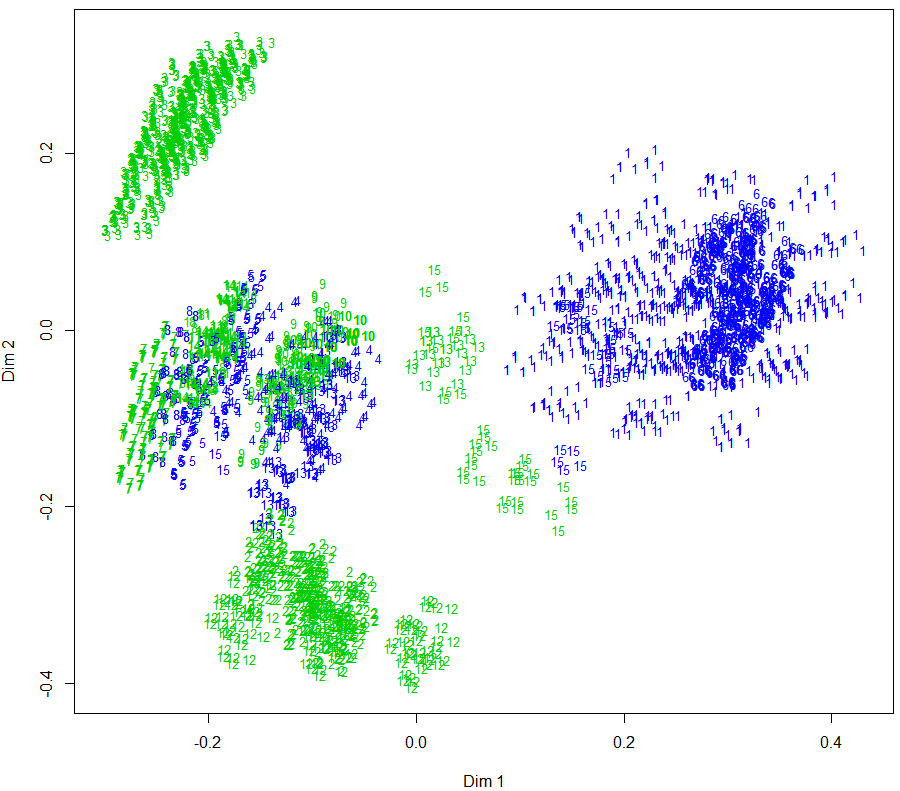
> text(mush.mds[,1:2],labels=grps2.15,

col=as.numeric(Mushrooms.train[,1])+2,cex=.6)



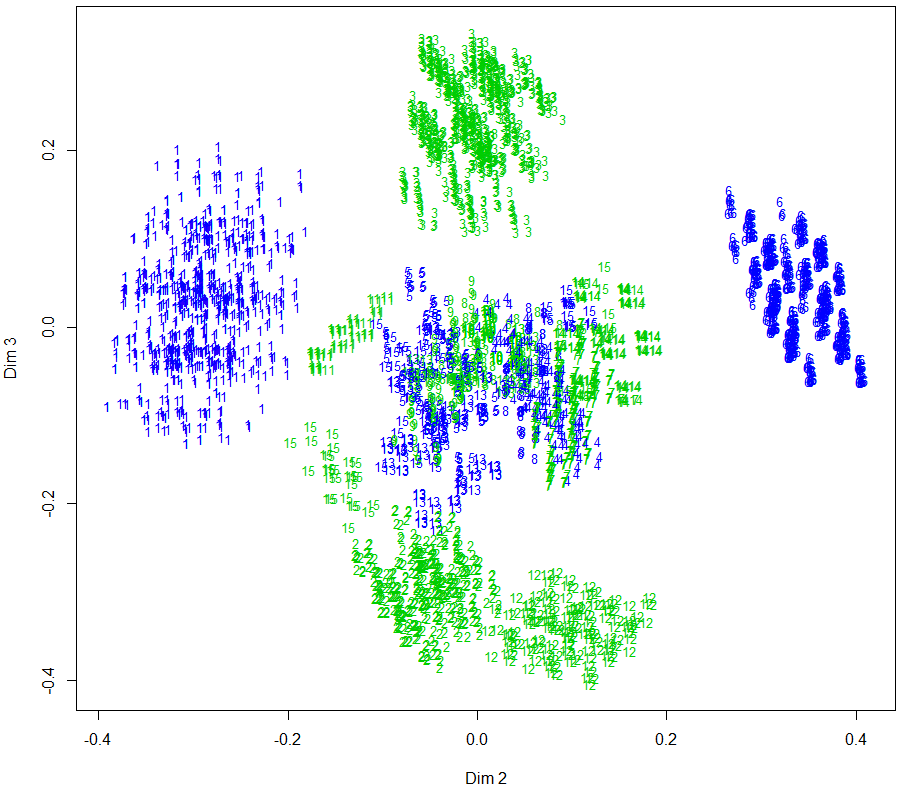
> plot(mush.mds[,c(1,3)],type="n",xlab="Dim 1",ylab="Dim 2")

> text(mush.mds[,c(1,3)],labels=grps2.15,  
col=as.numeric(Mushrooms.train[,1])+2,cex=.8)



> plot(mush.mds[,c(2,3)],type="n",xlab="Dim 2",ylab="Dim 3")

> text(mush.mds[,c(2,3)],labels=grps2.15,  
col=as.numeric(Mushrooms.train[,1])+2,cex=.8)



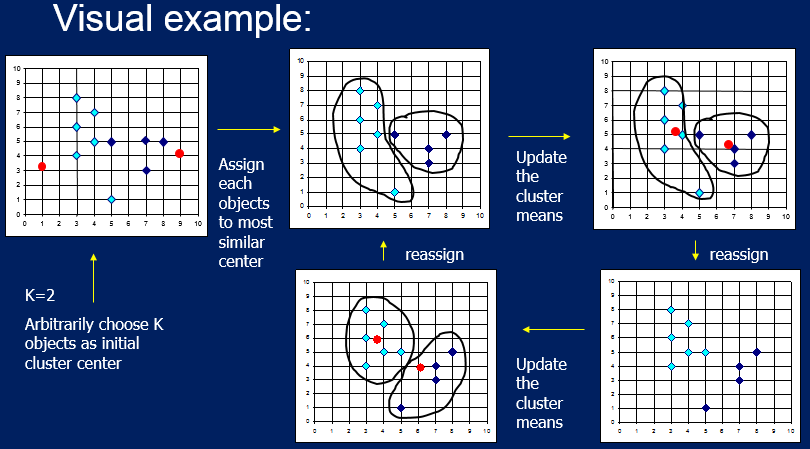
**4.4 - Cluster Analysis Using Partitioning Methods**

Partitioning methods use a different approach to form clusters. Given a choice for the number of clusters to be formed we start by randomly choosing k points in *p*-dimensional space to represent the cluster centers or *centroids*. Points are then classified to the cluster for whose centroid they are closest to. Then the cluster centroids are recomputed and the process is repeated until convergence, (i.e. no more changes can be made to the cluster assignments). The two methods we will examine are *k*-means and partitioning around medoids (PAM).

**K-Means Clustering**

The algorithm for *k*-means is summarized below for a given number of clusters *k*:

1. Partition objects to be clustered into *k* nonempty subsets.
2. Compute seed points as the centroids of the clusters of the current partition (the centroid is the mean vector for the observations in the cluster).
3. Assign each object to the cluster with the nearest seed point
4. Go back to Step 2, stop when no more new assignments.



Strength and Weaknesses of K-Means Clustering

Strength:

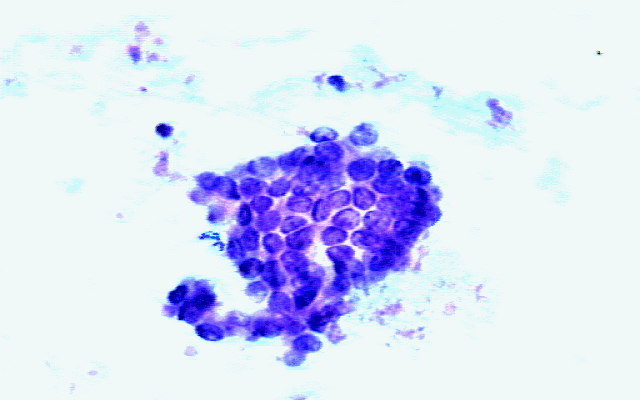
* Relatively efficient, does not require too many iterations to converge, though it often terminates at a local optimum and does not find the global optimum. Sometimes need to run it a few times.

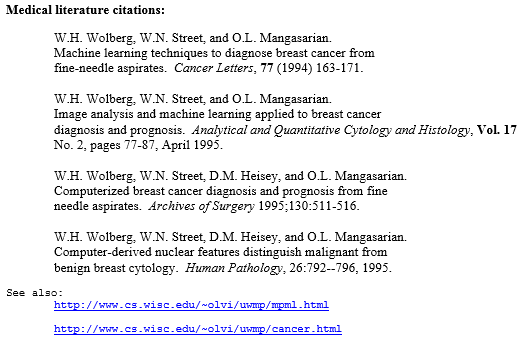
Weaknesses:

* Applicable only when mean is defined, then what about categorical data?
* Need to specify *k*, the number of clusters, in advance.
* Unable to handle noisy data and outliers.
* Not suitable to discover clusters with non-convex shapes.

Example 4.7: Breast Cancer Diagnosis Using Fine Needle Aspiration (**BreastDiag** in R save file)  
  
 Data Description:

Fine Needle Aspiration (FNA) is a fairly non-invasive method for sampling cells from a  
 breast tumor and then examining them using an electron microscope. The size and shape   
 features of cell can potentially effectively be used to classify a tumor as malignant or   
 benign. Features are computed from a digitized image of a fine needle aspirate (FNA) of   
 a breast mass. They describe characteristics of the cell nuclei present in the image. A   
 sample image is shown below.





Ten numeric features are the mean value based on three cells for the following cell   
 features:

* Radius = radius(mean of distances from center to points on the perimeter)
* Texture texture (standard deviation of gray-scale values)
* Perimeter = perimeter of the cell nucleus
* Area = area of the cell nucleus
* Smoothness = smoothness (local variation in radius lengths)
* Compactness = compactness (perimeter^2 / area - 1.0)
* Concavity = concavity (severity of concave portions of the contour)
* Concavepts = concave points (number of concave portions of the contour)
* Symmetry = symmetry (measure of symmetry of the cell nucleus)
* FracDim = fractal dimension ("coastline approximation" - 1)

The full data set contains the standard errors of the cell measurements (e.g. serad is the   
 standard error based on the three cell radius measurements) and worst case (maximum) value for   
 each (e.g. wrad = maximum cell radius of the three cells sampled)

> names(BreastDiag)

[1] "Id" "Diagnosis" "Radius" "Texture" "Perimeter" "Area"

[7] "Smoothness" "Compactness" "Concavity" "ConcavePts" "Symmetry" "FracDim"

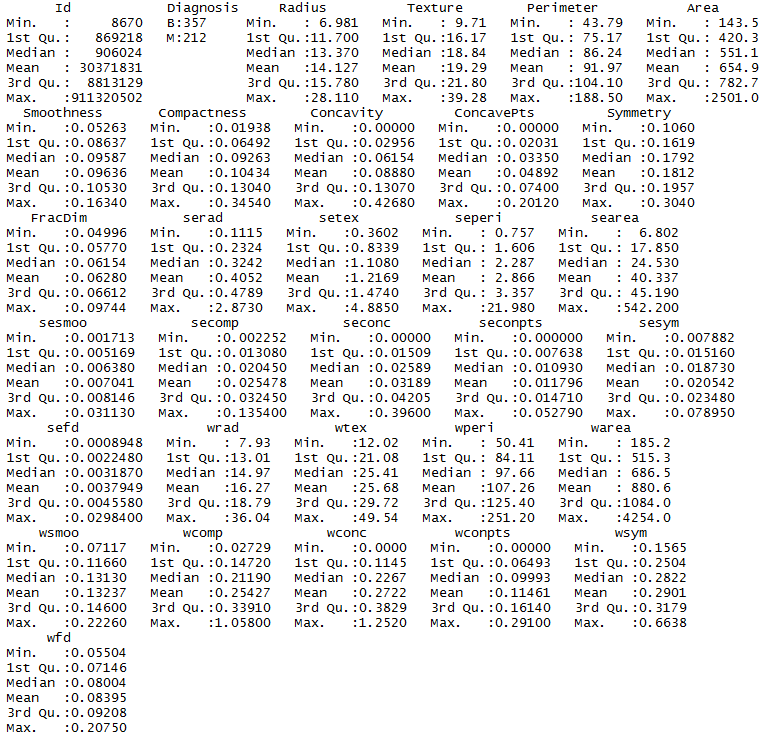
[13] "serad" "setex" "seperi" "searea" "sesmoo" "secomp"

[19] "seconc" "seconpts" "sesym" "sefd" "wrad" "wtex"

[25] "wperi" "warea" "wsmoo" "wcomp" "wconc" "wconpts"

[31] "wsym" "wfd"

> summary(BreastDiag)



> breast.mat = BreastDiag[,3:32]

> breast.mat = scale(breast.mat)

> breast.kmeans = kmeans(breast.mat,2)

> summary(breast.kmeans)

Length Class Mode

cluster 569 -none- numeric

centers 60 -none- numeric

totss 1 -none- numeric

withinss 2 -none- numeric

tot.withinss 1 -none- numeric

betweenss 1 -none- numeric

size 2 -none- numeric

iter 1 -none- numeric

ifault 1 -none- numeric

> table(breast.kmeans$cluster,BreastDiag$Diagnosis)

B M

1 343 37

2 14 175

Comments:

> bPCA = acp(breast.mat) 🡨 perform PCA using acp function in amap library

> summary(bPCA)

Length Class Mode

eig 29 -none- numeric

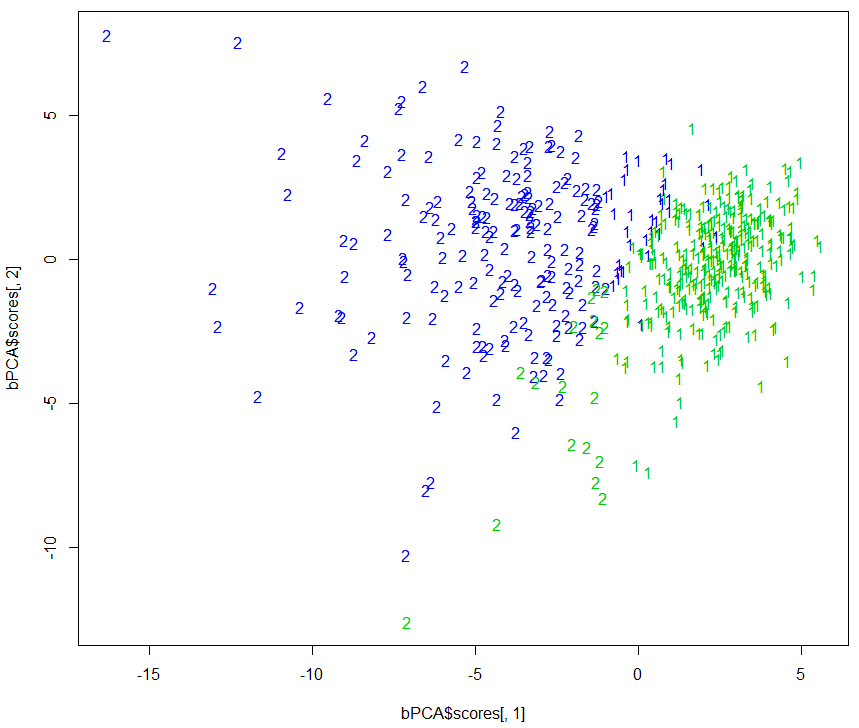
sdev 29 -none- numeric

scores 16501 -none- numeric

loadings 841 -none- numeric

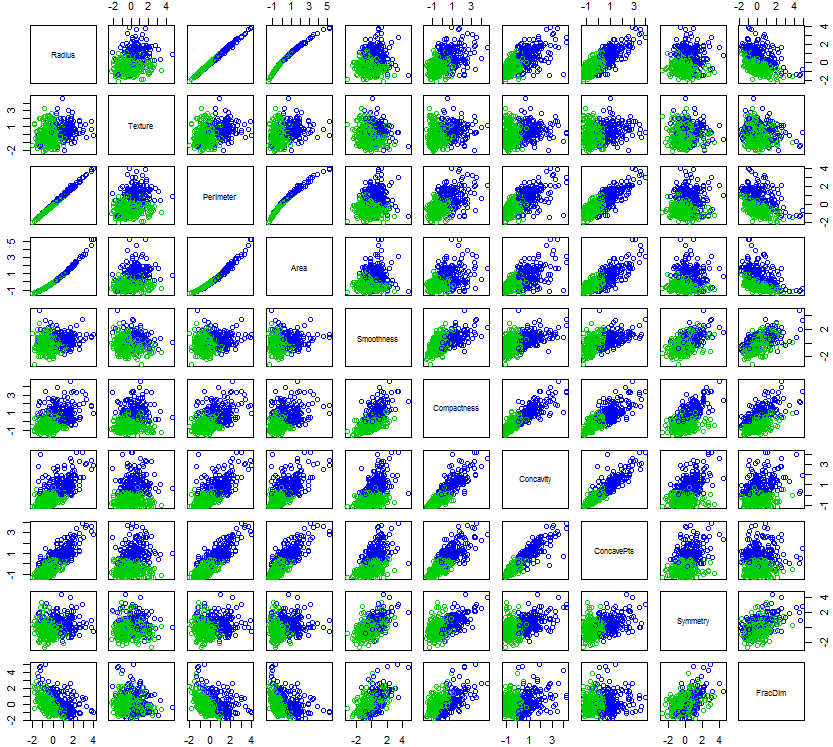
> plot(bPCA$scores[,1],bPCA$scores[,2],type="n")

> text(bPCA$scores[,1],bPCA$scores[,2],labels=breast.kmeans$cluster,

col=as.numeric(BreastDiag$Diagnosis)+2)  


> breast.kmeans$withinss

[1] 4971.437 6603.711

> pairs(breast.mat[,1:10],col=breast.kmeans$cluster+2)  


The two clusters formed using k-means clustering () almost perfectly coincide with cancer status of the cells.

Example 4.8: Sports Difficulty Data (Read in **SportsDiff.csv** file)

We will examine different methods of hierarchical clustering of these data before considering k-means.

> names(SportsDiff)

[1] "Endurance" "Strength" "Power" "Speed"

[5] "Agility" "Flexibility" "Nerves" "Durability"

[9] "Hand.Eye" "AnalyticAptitude" "Total\_Score" "Rank"

> head(SportsDiff)

Endurance Strength Power Speed Agility Flexibility Nerves Durability Hand.Eye

Boxing 8.63 8.13 8.63 6.38 6.25 4.38 8.88 8.50 7.00

Ice Hockey 7.25 7.13 7.88 7.75 7.63 4.88 6.00 8.25 7.50

Football 5.38 8.63 8.13 7.13 6.38 4.38 7.25 8.50 5.50

Basketball 7.38 6.25 6.50 7.25 8.13 5.63 4.13 7.75 7.50

Wrestling 6.63 8.38 7.13 5.13 6.38 7.50 5.00 6.75 4.25

Martial Arts 5.00 5.88 7.75 6.38 6.00 7.00 6.63 5.88 6.00

AnalyticAptitude Total\_Score Rank

Boxing 5.63 72.375 1

Ice Hockey 7.50 71.750 2

Football 7.13 68.375 3

Basketball 7.38 67.875 4

Wrestling 6.38 63.500 5

Martial Arts 6.88 63.375 6

Agglomerative Hierarchical Clustering

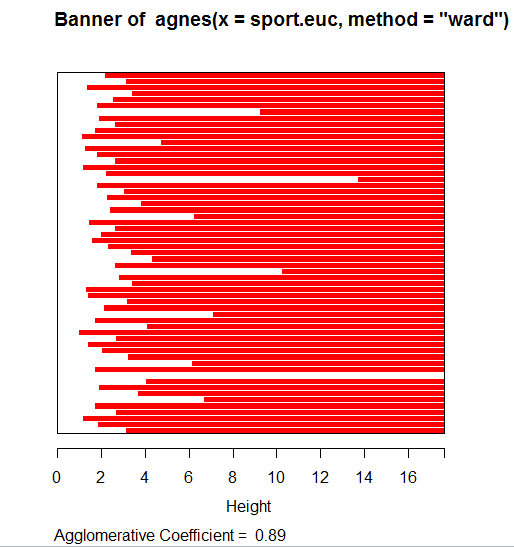
> sport.mat = scale(SportsDiff[,-c(11,12)])

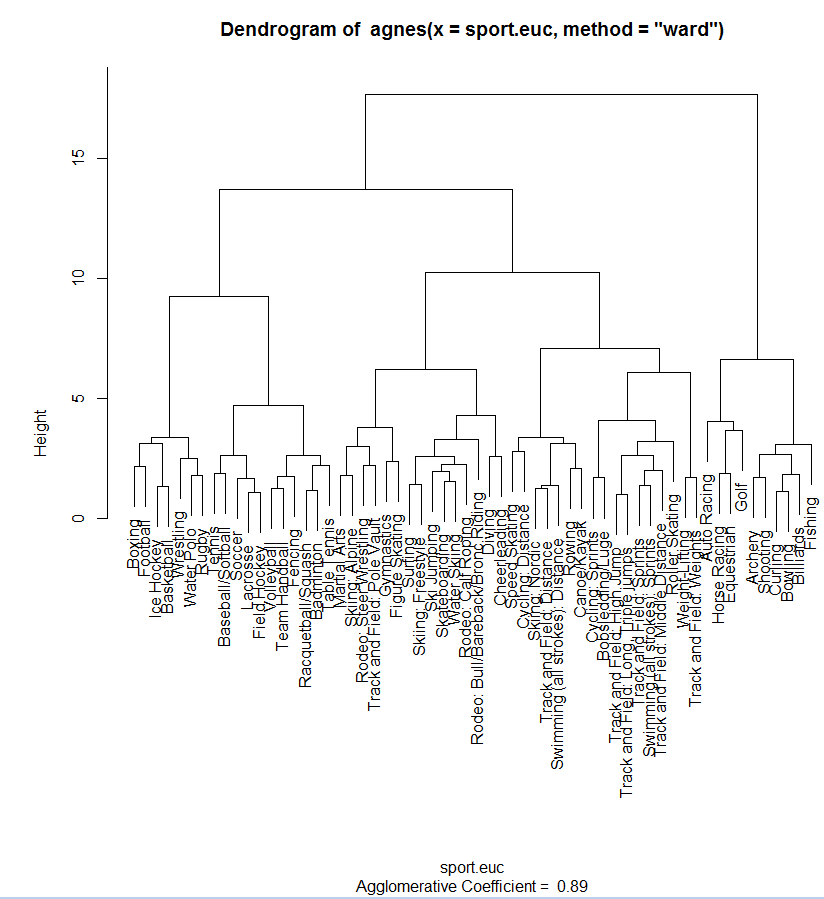
> sport.euc = dist(sport.mat)

> sport.man = dist(sport.mat,method=”manhattan”)

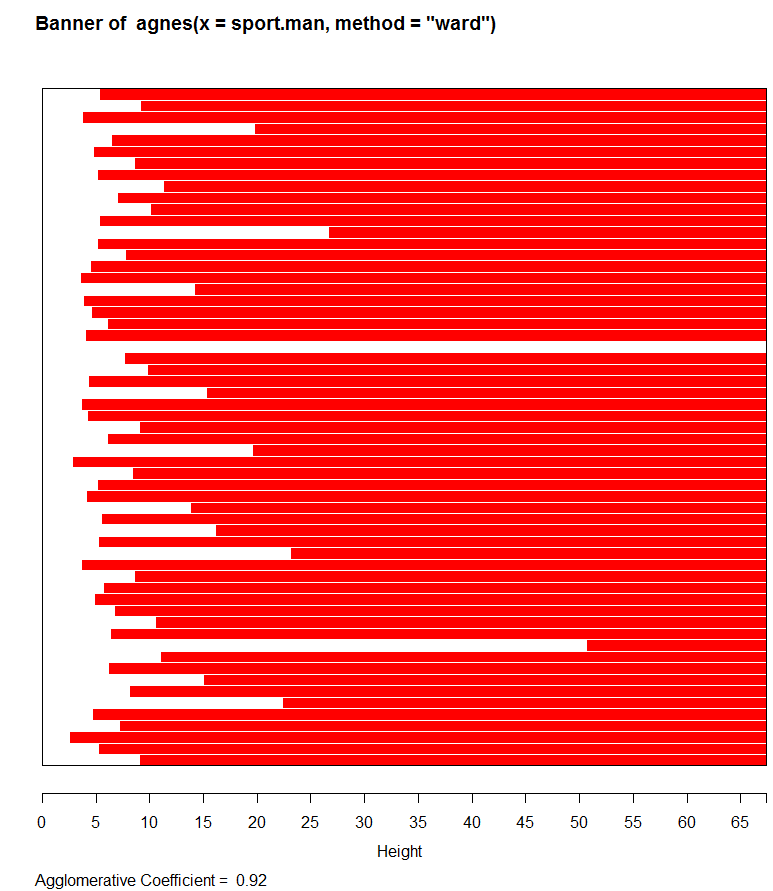
> sp1 = agnes(sport.euc,method="ward")

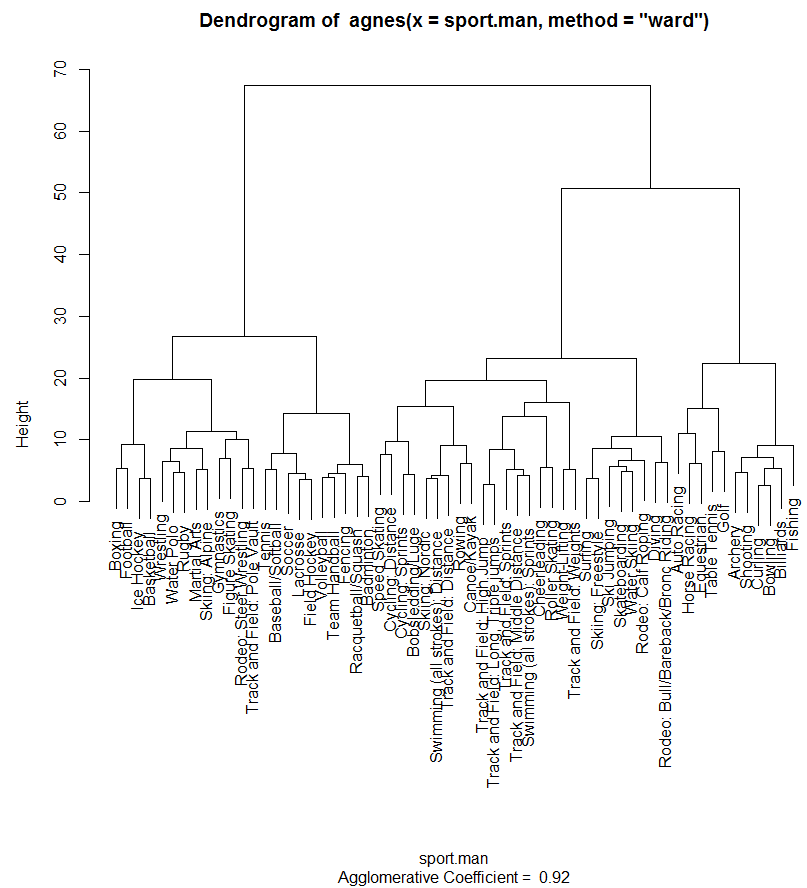
> plot(sp1)





> sp2 = agnes(sport.man,method=”ward”)

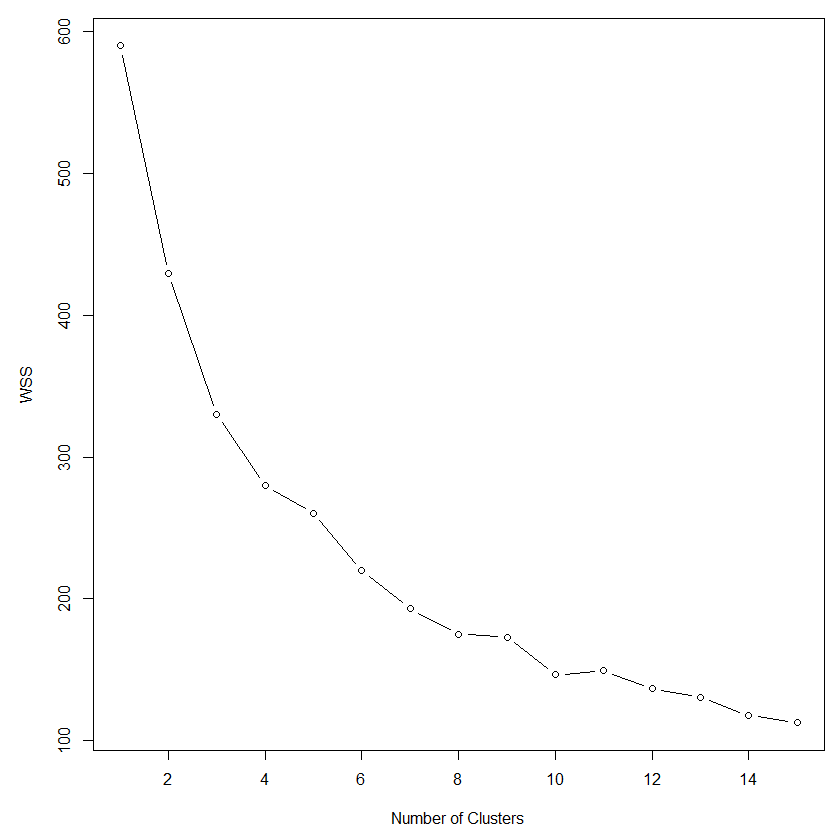
> plot(sp2)  




K-Means

> wss = rep(0,15)

> for (i in 1:15) wss[i]=sum(kmeans(sport.mat,centers=i)$withinss)  
> plot(1:15,wss,xlab=”Number of Clusters”,ylab=”WSS”,type=”b”)

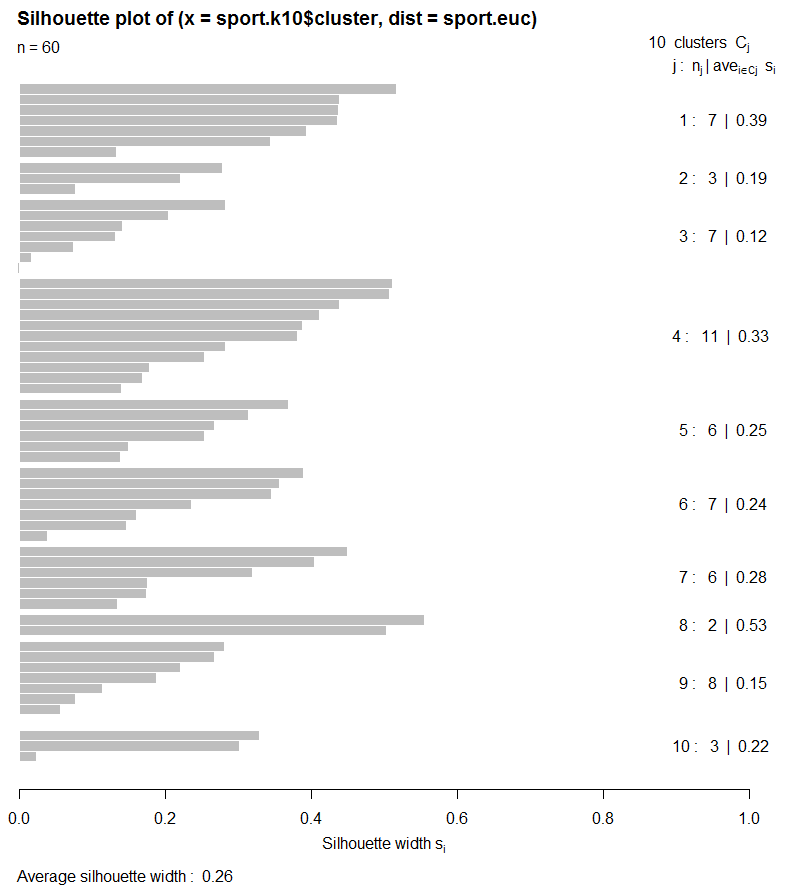


We can use this plot to help choose the number of clusters to use in our k-mean clustering solution.

> sport.k10 = kmeans(sport.mat,10)

> si = silhouette(sport.k10$cluster,sport.euc)

> plot(si)



> clust.grps(SportsDiff,sport.k10$cluster)

Cluster 1

=======================================================================

Golf Archery Curling Bowling Shooting Billiards Fishing

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

2.180000 2.698571 2.788571 1.074286 1.591429 2.450000

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

1.931429 1.538571 5.234286 4.574286 26.035714 56.571429

Cluster 2

=======================================================================

Speed Skating Cycling: Sprints Bobsledding/Luge

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

5.126667 6.293333 7.253333 7.710000 4.043333 3.460000

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

5.666667 4.210000 3.546667 4.083333 51.375000 24.000000

Cluster 3

=======================================================================

Martial Arts Gymnastics Skiing: Alpine Rodeo: Steer Wrestling Track and Field: Pole Vault Figure Skating Skiing: Freestyle

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

4.771429 5.931429 6.717143 5.575714 6.040000 7.108571

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

6.932857 5.574286 4.752857 4.645714 58.017857 14.285714

Cluster 4

=======================================================================

Tennis Baseball/Softball Soccer Lacrosse Field Hockey Volleyball Racquetball/Squash Fencing Team Handball Badminton Table Tennis

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

5.684545 4.274545 5.537273 5.853636 6.797273 5.013636

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

3.241818 4.378182 7.707273 6.468182 54.931818 20.545455

Cluster 5

=======================================================================

Track and Field: High Jump Track and Field: Long, Triple jumps Track and Field: Sprints Track and Field: Middle Distance Swimming (all strokes): Sprints Roller Skating

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

4.230000 5.086667 6.126667 7.253333 4.481667 5.231667

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

2.563333 3.710000 2.981667 2.961667 44.604167 40.833333

Cluster 6

=======================================================================

Boxing Ice Hockey Football Basketball Wrestling Water Polo Rugby

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

7.128571 7.450000 7.361429 6.414286 6.735714 5.128571

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

6.001429 7.715714 6.054286 6.468571 66.428571 5.571429

Cluster 7

=======================================================================

Cycling: Distance Skiing: Nordic Swimming (all strokes): Distance Rowing Track and Field: Distance Canoe/Kayak

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

8.731667 5.938333 5.295000 4.876667 3.313333 4.065000

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

3.106667 5.065000 2.900000 3.961667 47.229167 35.666667

Cluster 8

=======================================================================

Weight-Lifting Track and Field: Weights

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

3.6900 8.5650 9.4400 2.8150 2.8150 3.1900

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

3.1250 4.1900 3.1250 2.6300 43.5625 46.0000

Cluster 9

=======================================================================

Surfing Ski Jumping Diving Skateboarding Rodeo: Calf Roping Rodeo: Bull/Bareback/Bronc Riding Water Skiing Cheerleading

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

3.72250 4.72125 4.39250 3.40750 4.73750 5.56375

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

7.00250 4.94000 4.16000 3.36125 45.98438 38.00000

Cluster 10

=======================================================================

Auto Racing Horse Racing Equestrian

Variable means in this cluster are:

-----------------------------------------------------------------------

Endurance Strength Power Speed Agility Flexibility

4.420000 3.543333 2.420000 1.420000 2.710000 2.793333

Nerves Durability Hand.Eye AnalyticAptitude Total\_Score Rank

7.960000 3.876667 4.920000 6.376667 40.416667 45.333333

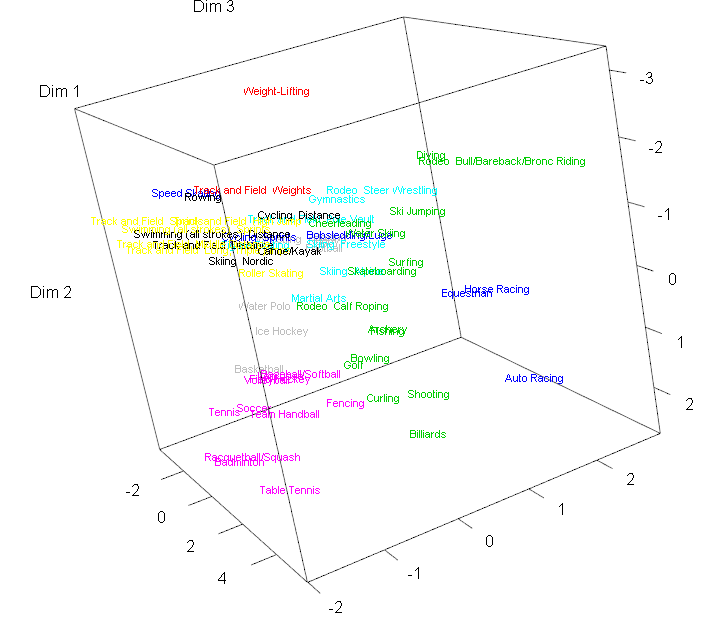
To potentially better understand clusters we can look at MDS.

> sport.mds = cmdscale(sport.euc,k=3)

> plot3d(sport.mds,type="n",xlab="Dim 1",ylab="Dim 2",zlab="Dim 3")

> text3d(sport.mds,texts=row.names(SportsDiff),

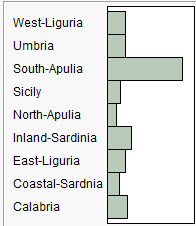
col=sport.k10$cluster+2,cex=.7)



Example 4.9: Fatty Acid Content of Italian Olive Oils

Researchers are interested in characterizing differences in the fatty acid content of olive oils made from olives grown in different regions of Italy. There are two geographic classifications in these data. The first classification is nine individual growing areas in Italy (Area Name) – East Liguria, West Liguria, Umbria, North-Apulia, South-Apulia, Sicily, Coastal Sardinia, Inland-Sardinia, and Calabria. A broader classification is the growing region in Italy (Region Name) – Northern, Southern, and Sardinia. The map below should help in your understanding of where these areas/regions are located in Italy.





Puglia = Apulia

Sardegna = Sardinia

Sicilia = Sicily

The bar graph above shows the number of olive oils in these data from each area.

***The fatty acids measured are as follows:***

Palmitic

Palmitoleic

Stearic

Oleic

Linoleic

Eicosanoic

Linolenic

Molecular formulae taken from *Wikipedia*, so if these are wrong it is not my fault. I don’t understand how small differences in the number of carbon and hydrogen molecules make distinct fatty acids. Chemistry is weird!

We begin this example by using k-means clustering with the Italian olive oils in these data.

> names(Olives)

[1] "Region.name" "Area.name" "Region" "Area" "palmitic" "palmitoleic"

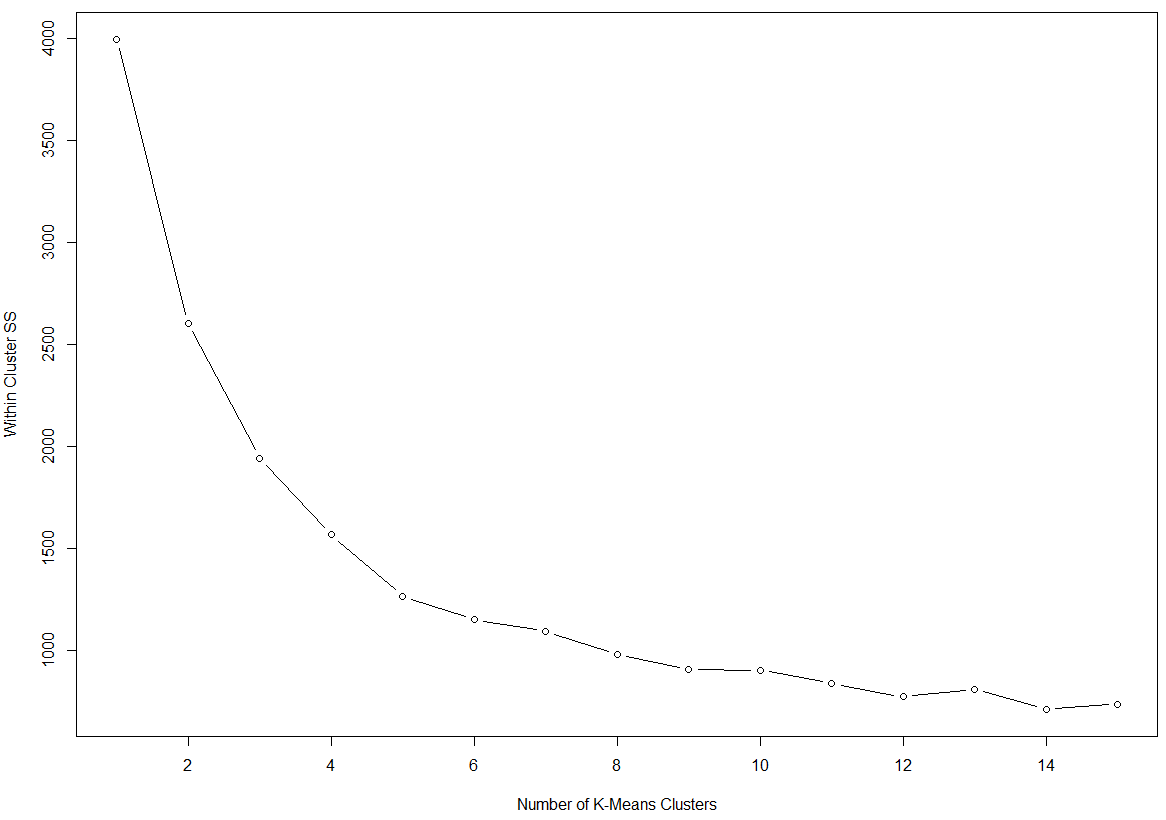
[7] "strearic" "oleic" "linoleic" "eicosanoic" "linolenic" "eicosenoic"  
  
> olive.mat = Olives[,5:11]

> olive.scale = scale(olive.mat 🡨scaling numeric variables when clustering   
 is critical when using distance based clustering   
 methods.

> wss = rep(0,15)

> for (i in 1:15) wss[i] = sum(kmeans(olive.scale,centers=i)$withinss)

> plot(1:15,wss,type="b",xlab="Number of K-Means Clusters",ylab="Within   
 Cluster SS")



The use of 7 – 10 clusters appears reasonable.   
  
Remember there are actually data from 9 growing areas in Italy.

> fit = kmeans(olive.scale,9) 🡨 use k = 9 clusters

> aggregate(olive.mat,by=list(fit$cluster),FUN=mean)🡨 more efficient way to obtain   
 group centers in the original scale.

Group.1 palmitic palmitoleic strearic oleic linoleic eicosanoic linolenic

1 1 1064.746 104.06780 256.1017 7698.644 861.8644 4.915254 9.491525

2 2 1438.226 183.41935 256.5806 6780.258 1177.9032 41.258065 69.322581

3 3 1080.564 73.29091 264.7455 7723.418 709.0909 41.654545 74.836364

4 4 1587.240 238.40000 210.3200 6563.680 1270.2000 32.880000 61.280000

5 5 1111.347 96.74490 226.1837 7268.020 1196.5306 27.091837 73.173469

6 6 1349.705 170.33333 219.5000 7058.103 1056.1410 34.192308 62.833333

7 7 1095.133 65.76000 203.5200 7914.027 618.6400 30.573333 47.013333

8 8 1290.538 113.83077 260.9846 7320.723 824.0154 46.246154 67.938462

9 9 1370.116 176.87209 192.0116 6939.942 1199.3488 34.151163 56.523256  
  
 Group centroids

> table(fit$cluster,Olives$Area)

1 2 3 4 5 6 7 8 9

1 0 0 0 0 0 0 9 50 0

2 0 1 25 5 0 0 0 0 0

3 15 0 0 15 0 0 24 0 1

4 0 0 25 0 0 0 0 0 0

5 0 0 0 0 65 33 0 0 0

6 0 2 70 6 0 0 0 0 0

7 9 2 0 0 0 0 14 0 50

8 1 51 0 10 0 0 3 0 0

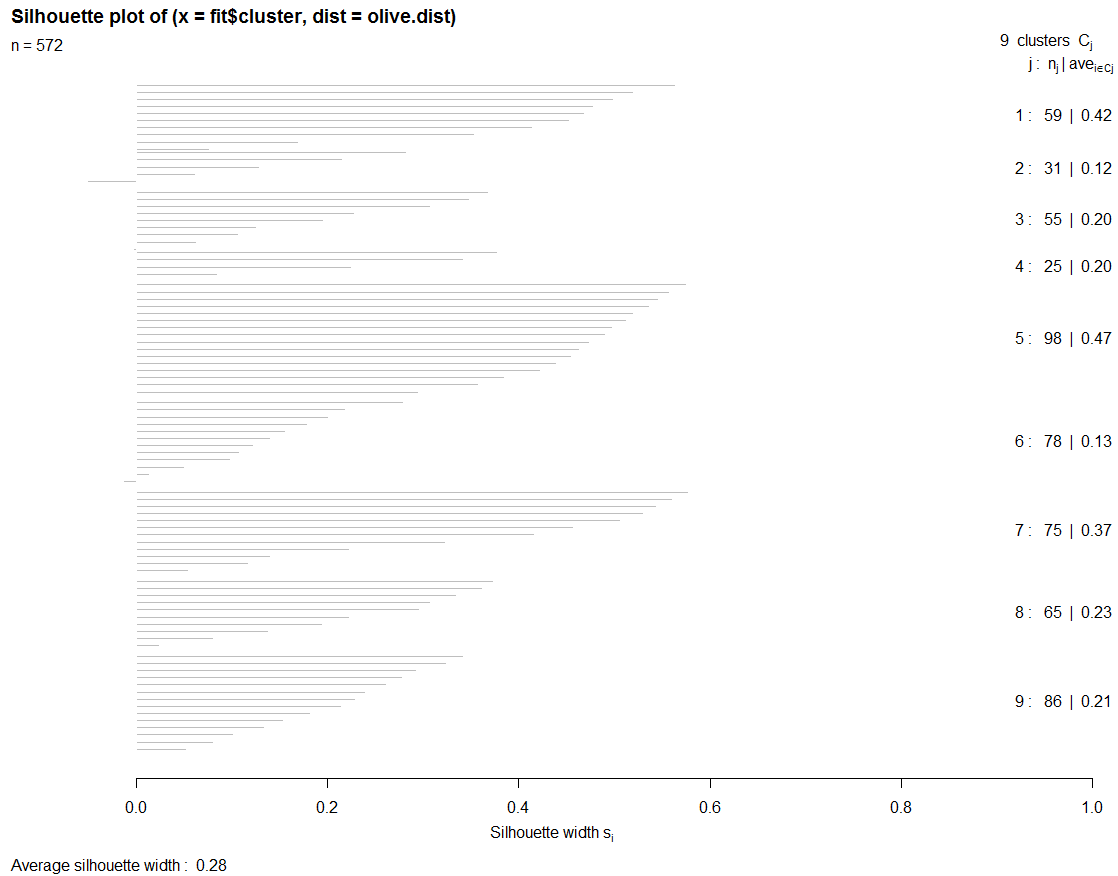
9 0 0 86 0 0 0 0 0 0

The clusters are fairly homogenous in terms of what growing areas the olive oils are from.

> olive.dist = dist(olive.scale)

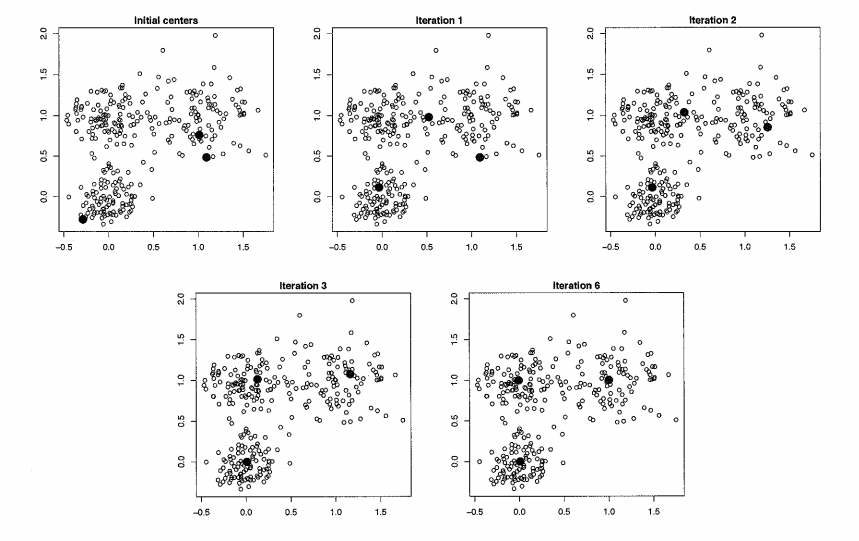
> si = silhouette(fit$cluster,olive.dist)

> plot(si)



Partitioning Around Medoids (PAM)

The partitioning around medoids (PAM) algorithm is similar to k-means but uses medoids (representative observations) rather than centroids (i.e. cluster means) to determine clusters. It takes either a data matrix or a dissimilarity matrix as an argument. The algorithm computes *k* representative objects, called medoids, which together with the distances determine the clustering. Each object in the data set is then assigned to the cluster corresponding to the nearest medoid. That is, object *i* is put into the cluster it is nearest to in terms of the distance to the *k* cluster medoids. After the initially assignment of all objects to the randomly chosen medoids, new medoids may be chosen and then assignment of all objects that are not medoids is done again. The process repeats until convergence.



PAM clustering is implemented by the pam function in the cluster library and there is a nice function for determining the “optimal” *k* value called pamk in the fpc library.

Below I illustrate the use of PAM clustering with the Italian olive oils data set.

> library(cluster)

> library(fpc) 🡨 needs to be installed from CRAN

> olive.mat = scale(Olives[,5:11]) 🡨 not using eicosenoic acid which is known to have errors.

> fits = pamk(olive.mat,krange=1:15,diss=F,criterion="asw",critout=T)

1 clusters 0

2 clusters 0.3120916

3 clusters 0.2831431

4 clusters 0.3273392

5 clusters 0.3703581

6 clusters 0.3585159

7 clusters 0.3342859

8 clusters 0.3283578

9 clusters 0.2734017

10 clusters 0.3085691

11 clusters 0.2757455

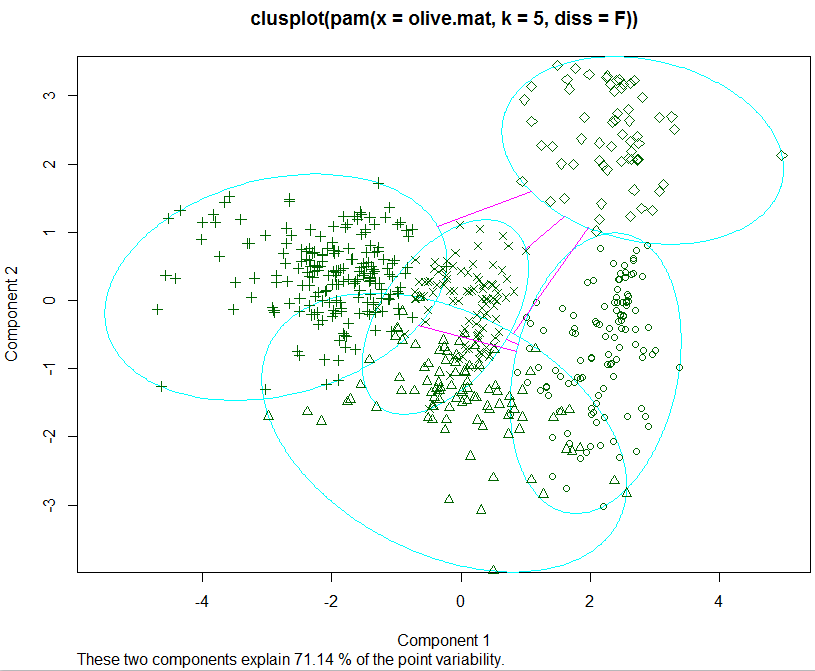
12 clusters 0.2646717

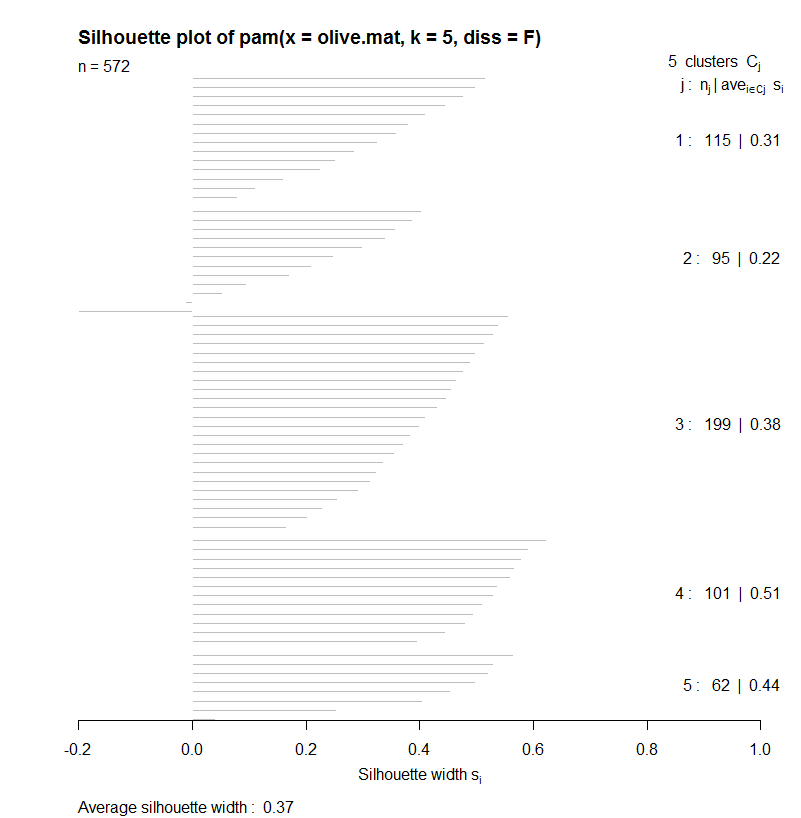
13 clusters 0.2591018

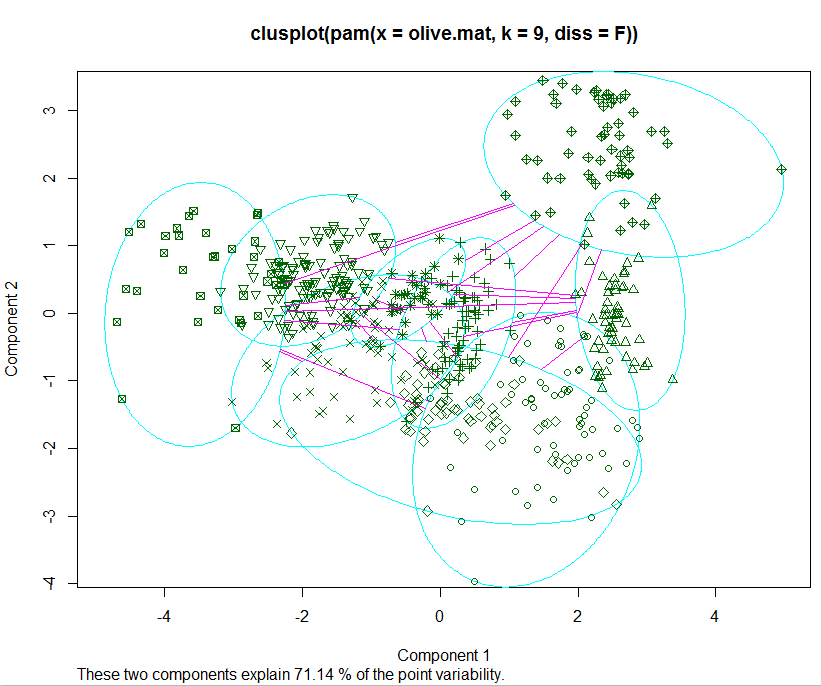
14 clusters 0.23787

15 clusters 0.2406563   
  
> fit5 = pam(olive.mat,diss=F,k=5)

> plot(fit5)





> fit9 = pam(olive.mat,diss=F,k=9)  




> table(Olives$Area,fit9$clustering)

1 2 3 4 5 6 7 8 9

1 20 3 2 0 0 0 0 0 0

2 4 0 0 9 43 0 0 0 0

3 0 0 2 47 1 122 34 0 0

4 7 0 0 9 18 1 1 0 0

5 0 0 65 0 0 0 0 0 0

6 0 0 2 0 0 0 0 31 0

7 32 9 0 0 0 0 0 0 9

8 0 0 0 0 0 0 0 0 50

9 3 48 0 0 0 0 0 0 0

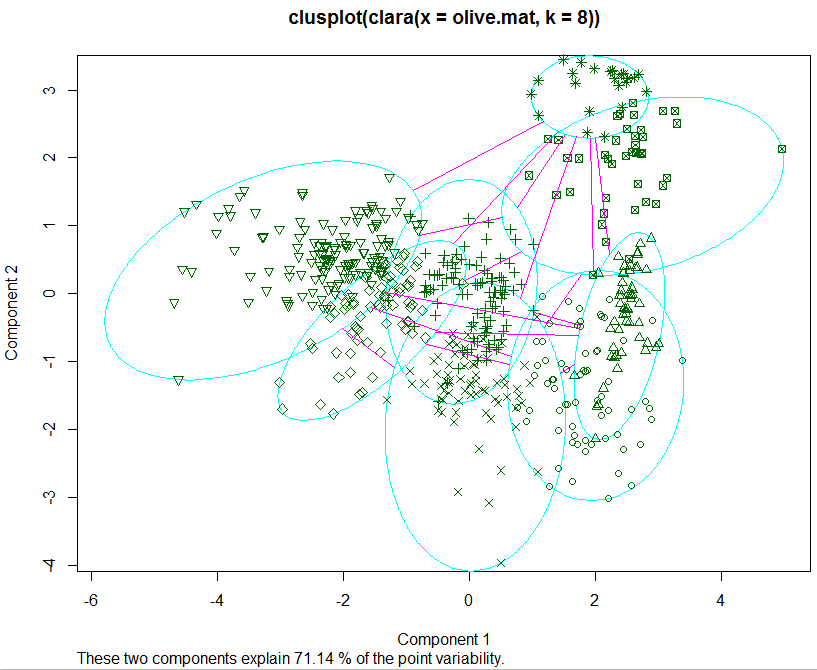
Again the clustering is strongly related to the nine growing areas, but not perfectly.

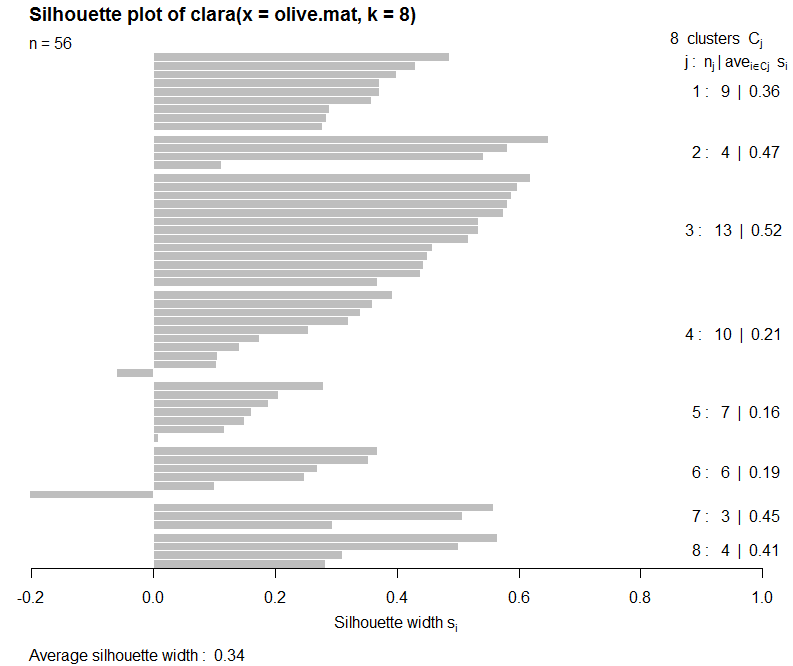
CLARA

CLARA stands for Clustering LARge Applications (large n). A random subset of the original observations is taken and each sub-dataset is partitioned into *k* clusters using the same algorithm as in PAM. Once k representative objects have been selected from the sub-dataset, each observation of the entire dataset is assigned to the nearest medoid. So basically CLARA performs PAM on a subset of the entire data set and then clusters the observations not chosen in the random subset to the clusters determined using PAM on the subset.

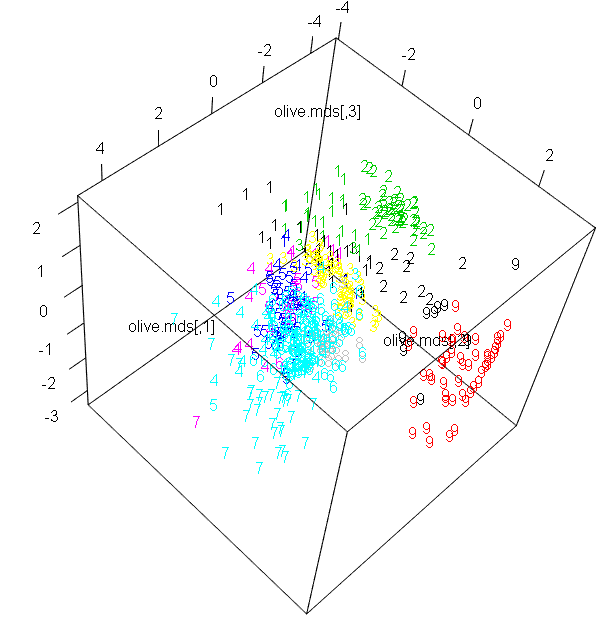
> fit8 = clara(olive.mat,8)

> plot(silhouette(fit8))





3-D Multidimensional Scaling (labels = clusters from CLARA, color = growing area)



Self-Organizing Maps (SOM)

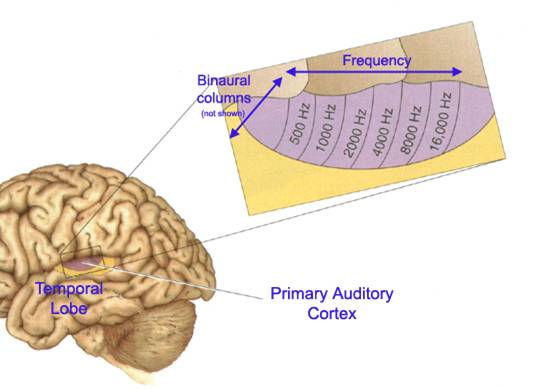
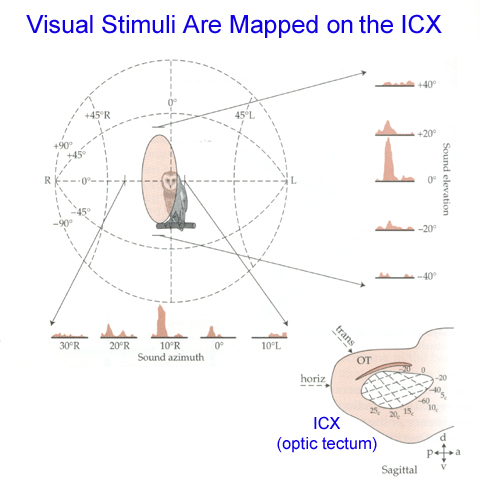
Self-organizing maps (SOMs, Kohonen 2001) work similarly to MDS, but instead of trying reproduce distances they aim at reproducing topology, or in other words, they try to keeps the same neighbors. So if two high-dimensional (p > 2) objects are very similar, then their position in a two-dimensional place should be very similar as well. Rather than mapping objects in a continuous space (i.e. the dimensions in 2-D MDS), SOMs use a regular grid of “units” onto which objects are mapped. The differences with MDS can be seen as both strengths and weaknesses: where in a 2-D MDS plot a distance between two objects can be directly interpreted as an “estimate” of the true distance between in the objects in the higher dimensional space, in a SOM plot this is not the case: one can only say that objects mapped to the same, or neighboring units, are very similar. In other words, SOMs concentrate on the largest similarities, whereas MDS concentrates on the largest dissimilarities. SOMs are also analagous to k-means clustering. In that analogy, every unit the SOM map corresponds to a “cluster”, the number of clusters is defined by the size of the grid, which typically is arranged in a rectangular or hexagonal fashion.

****

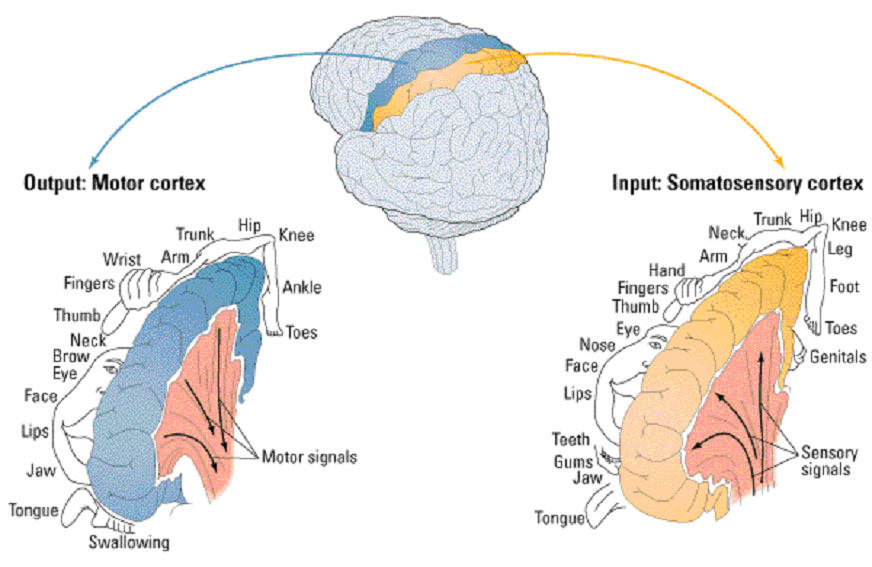
****

This is what takes place in the brain and is akin to the supervised learning modeling method of neural networks.

***Auditory Stimuli Visual Stimuli***

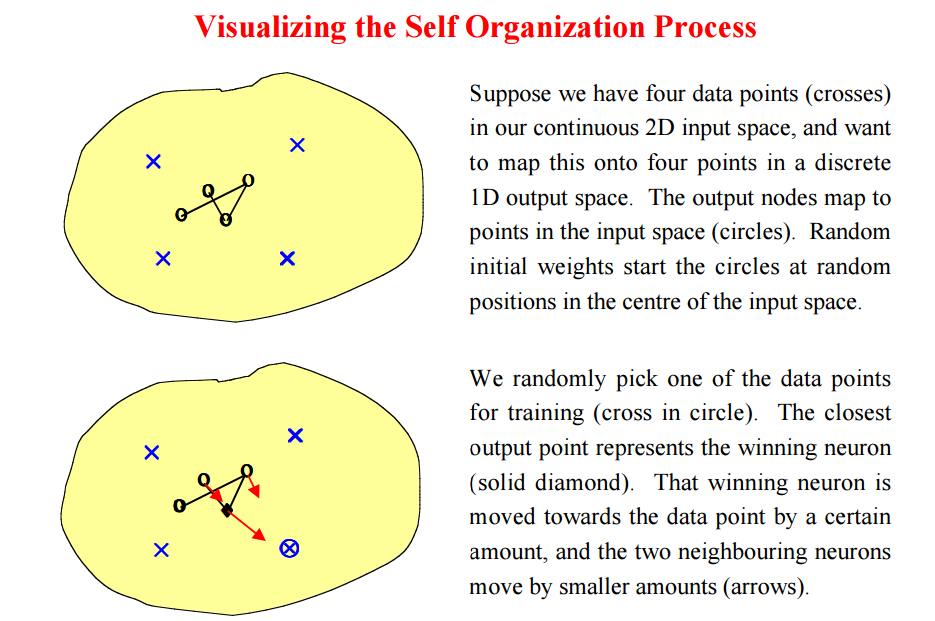
***Input nerve stimuli***

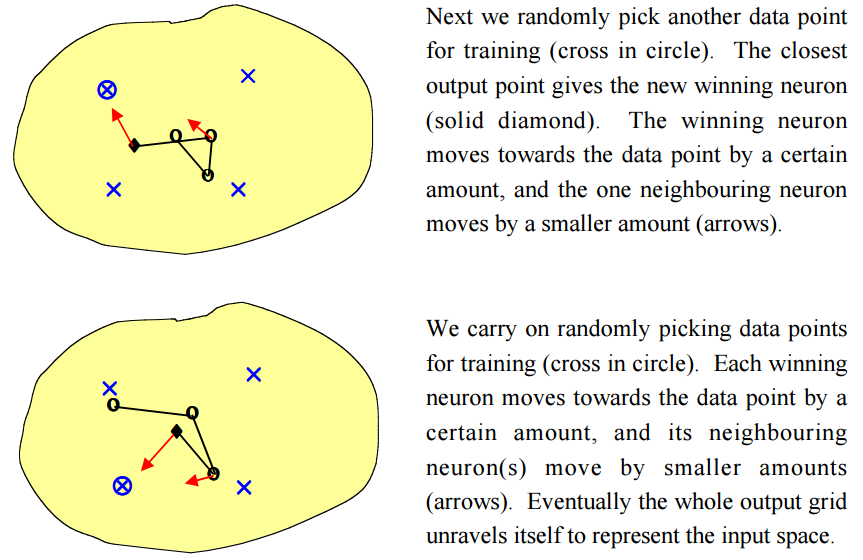


Example**:** SOM of poverty and income related inputs for countries throughout the world. We can see that the inputs are organized to form clusters of economically similar countries.

****

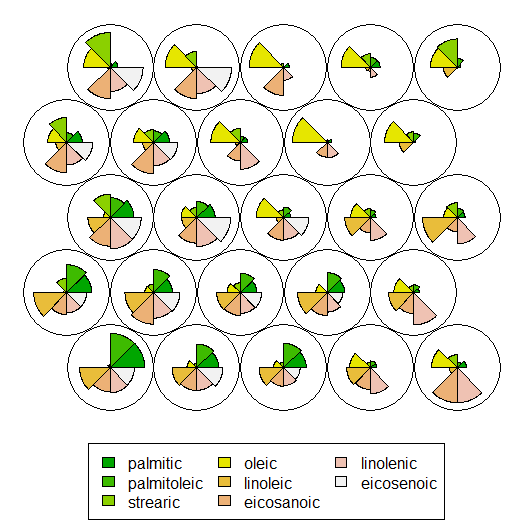
**SOM Algorithm**





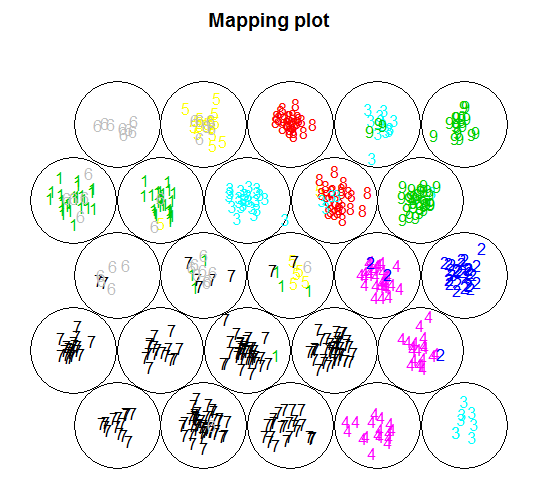
Example 5.7: Fatty Acid Content of Italian Olive Oils (cont’d)  
We first an unsupervised SOM fit to a 5 X 5 grid.

> kohmap = som(scale(O.mat),grid=somgrid(5,5,"hexagonal"))

> plot(kohmap)   


> plot(kohmap,type="mapping",labels=as.numeric(Olives$Area.name),

col=as.numeric(Olives$Area.name)+2)



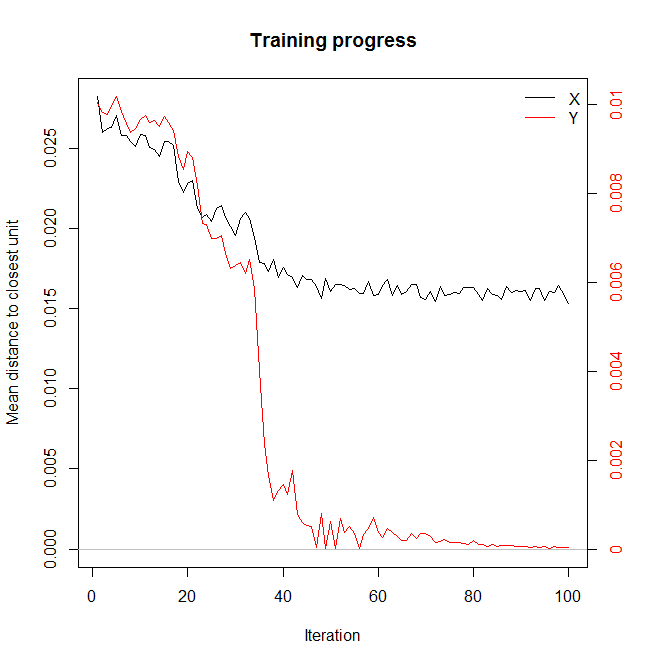
Here we can see that the SOM organizes the oils according to area grown to some degree but there if definitely some confusion. We can provide group labels to the algorithm by appending that information to the matrix of inputs **X** in form of a vector dummy variable with 0’s in eight positions and a 1 in the position corresponding to the area its from.

**X Y**

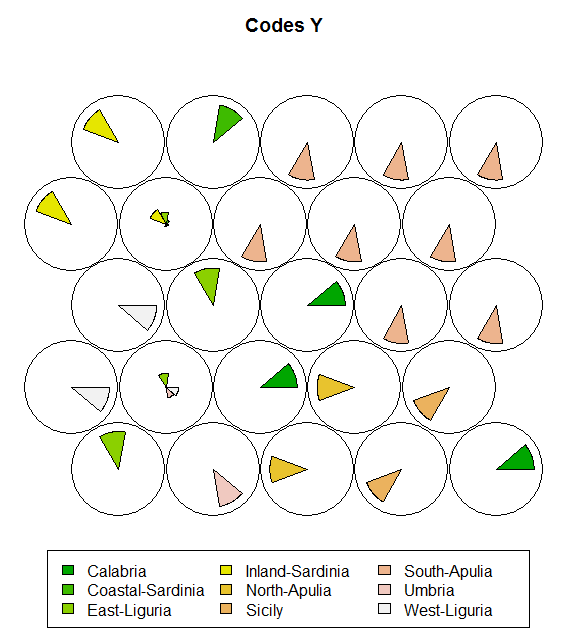
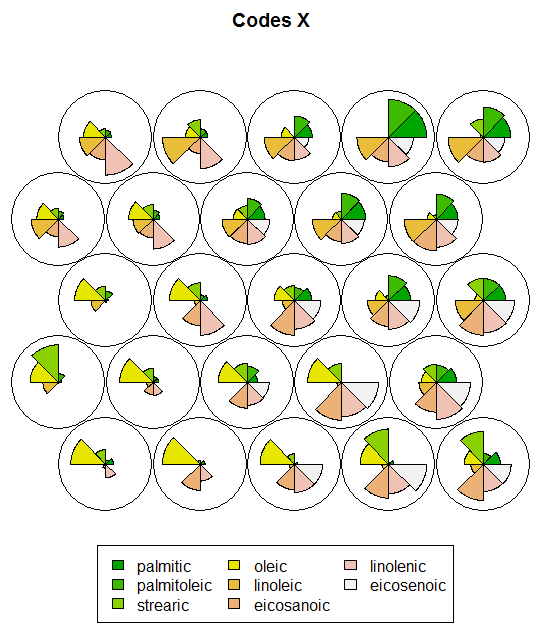
> kohmap <- xyf(scale(O.mat), classvec2classmat(O.area$Area.name),

+ grid = somgrid(5, 5, "hexagonal"), rlen=100)

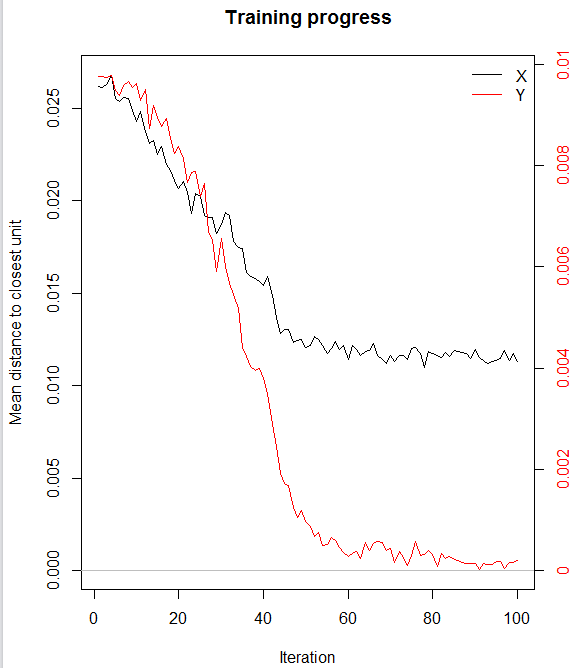
> plot(kohmap, type="changes")

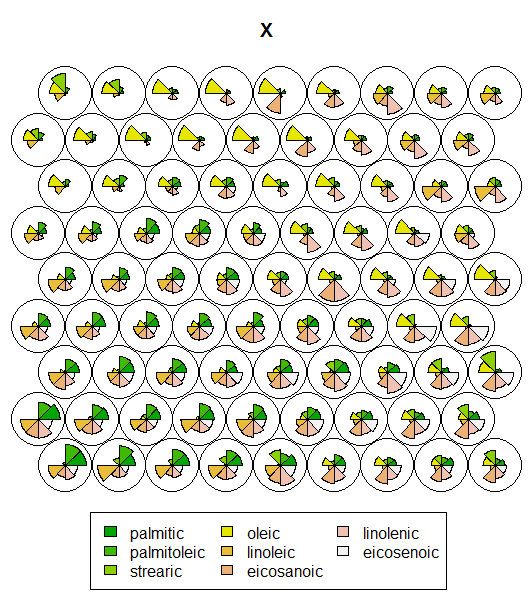
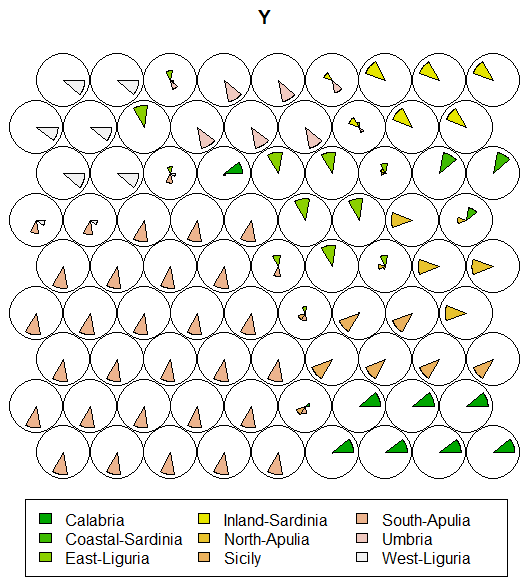


> plot(kohmap)

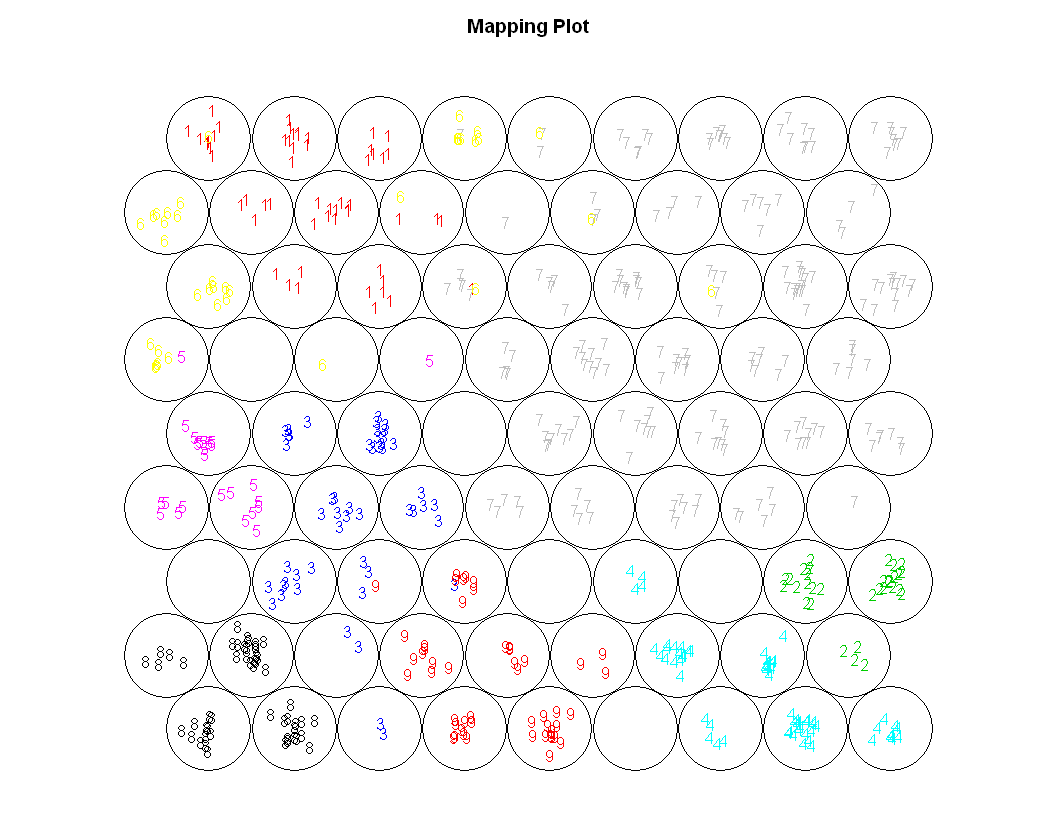


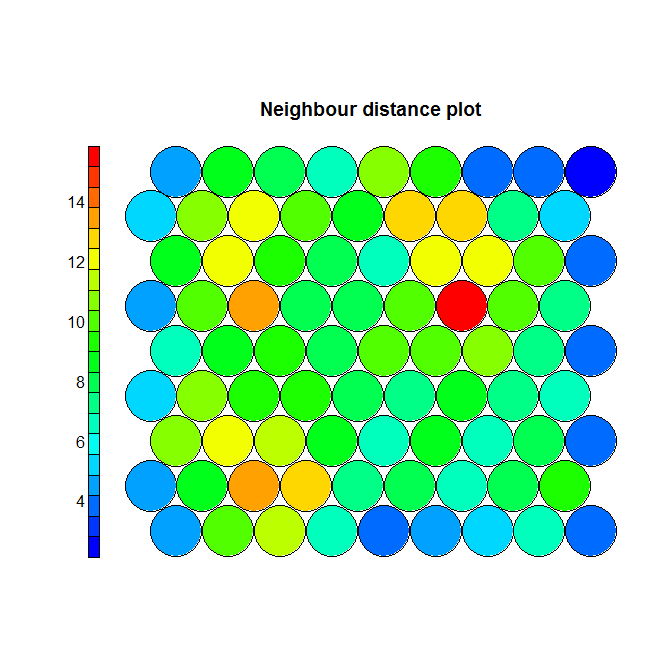
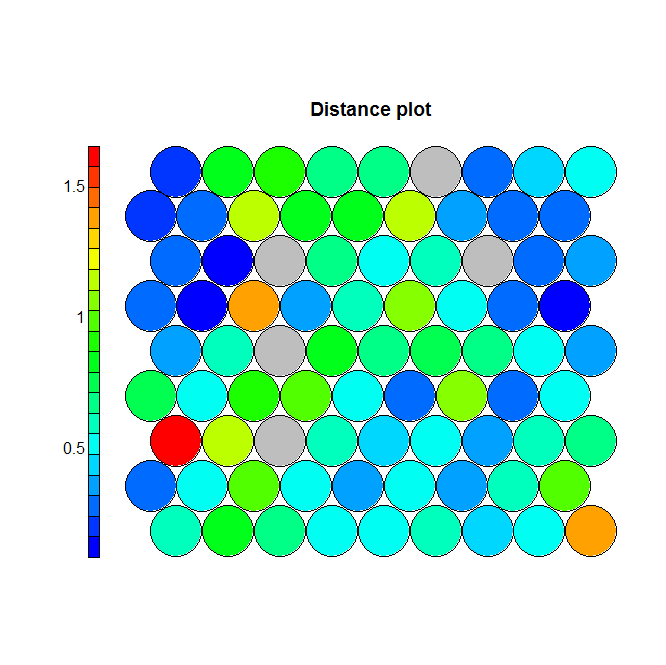
Using a 9 X 9 grid instead.



> plot(kohmap,type="mapping",labels=as.numeric(Olives$Area.name),  
col=as.numeric(Olives$Area.name)+2)



> plot(kohmap,"dist.neighbours",palette=coolBlueHotRed)  
> plot(sommap,"quality",palette=coolBlueHotRed)  
 

Example 4.10: Clustering Brazilian Facial Images

Brazil Faces Cluster 1



Brazil Faces Cluster 2



Brazil Faces Cluster 3



Brazil Faces Cluster 4



Brazil Faces Cluster 5



