Chapter 8

Cluster Analysis

So far, the methods we mentioned such as Binary Logistic regression, Classification Tree and Neural Network can derive rules from a **training** data set. That is, the data set has known group members. For example, stocks with known *HSI=0* or *1*, customer with bad debt or not, patient with certain disease or not, etc. These methods are also known as **supervised learning**. However, we may have data set **without** knowing the group information or classification. Even worst, we may not know how many groups exist in the data set. For example, in marketing, we may want to classify the customers into several segments so that we may have different strategies to promote our products to different segments. This is known as **unsupervised learning**. The problems in unsupervised learning are much more difficult to deal with than that in supervised learning; for example, imagine we want to classify the Iris flowers into species without knowing which species each flower belongs to and how many species are in the data set.

8.1 Cluster Analysis and K-means Clustering

Generally speaking, unsupervised learning is more difficult than supervised learning. Furthermore, there is a basic difficulty in unsupervised learning. There is no unique objective criterion to cluster our data. For example, in a deck of 52 playing cards, we can group the cards according to their color (red or black) or according to their suits (spades, heart, club and diamond). Or we can even group the cards according to letters or numbers (A,K,Q,J; 1-10) etc. This problem arises simply because we do not have an objective measure of **similarity** or **dissimilarity**.

However, as long as we have a measure of similarity (or dissimilarity), we can classify the data into clusters such that observations within each cluster are as homogenous as possible while observations belong to different clusters are as heterogeneous as possible. There are two major types of methods in Cluster analysis, namely **hierarchical** and **non-hierarchical** method. Hierarchical method tries to link the two most similar observations in the data set to form a cluster, and then link up the next two most similar objects and so on. Finally, it will build up a tree-hierarchy structure with lines joining all the observations. However, it is not feasible when the number of observations is large. Here we introduce a very popular non-hierarchical clustering method called K-means clustering.

K-means clustering is fast, efficient and suitable for large data set in Data Mining. In K-means clustering, we want to cluster n observations into k homogenous groups. Note that the number of clusters, k, has to be specified in advance. Here is a brief description of the algorithm:

1. Randomly chose k data points as the initial seeds.
2. For each observation, calculate the distance of this point to these k seeds, and assign this point to group j if it is closest to the j^{th} seed.
3. We compute the mean of the k clusters in (2). Then we re-compute the distance of each observation to the k cluster means. Re-assign the observation to group j^{th} group if it is closest to group j.
4. Repeat (3) until converge.

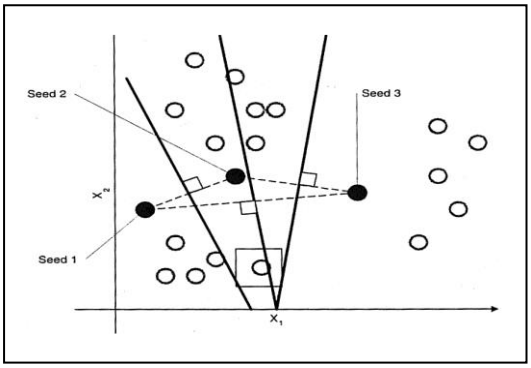


Fig. 1 Initial seeds determine the initial Cluster boundaries

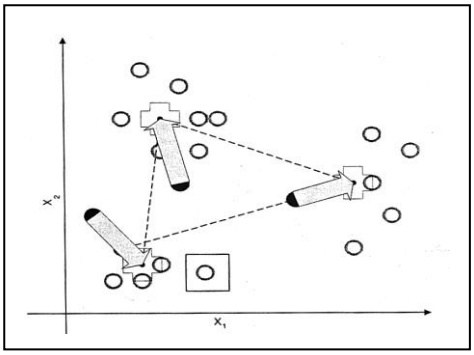


Fig. 2 Computing the centroids of the new clusters

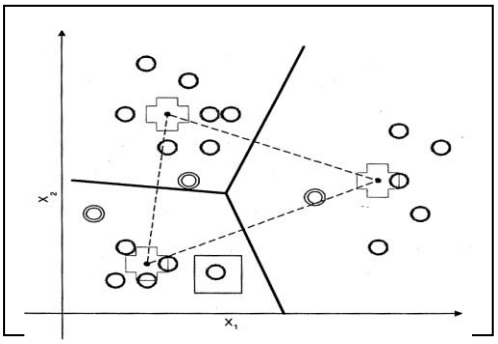


Fig 3. Re-assignment of records to new clusters

Mathematically speaking, k-means clustering is to find the class label *z* to minimize the within group sum of squares:

$$E(z) = \sum_{j=1}^{k} \sum_{i=1}^{n} z_{ij} (x_i - \bar{x}_j)^2 \text{ , where } z_{ij} = 1 \text{ if } x_i \text{ is in group } j; z_{ij} = 0 \text{ otherwise, } \quad (8.1)$$

and

$$\bar{x}_j = \frac{\sum_{i=1}^{n} x_i z_{ij}}{\sum_{i=1}^{n} z_{ij}}$$

8.2 K-means clustering using R

The K-means clustering method is implemented in R using a built-in function *kmeans*() in R. Let us illustrate this by Fisher's Iris data and our HSI example.

```
> d<-read.csv("iris.csv")    # read in data
> d1<-d[,1:4]                # save the the first 4 columns to d1
> km<-kmeans(d1,3)           # K-means clustering
```

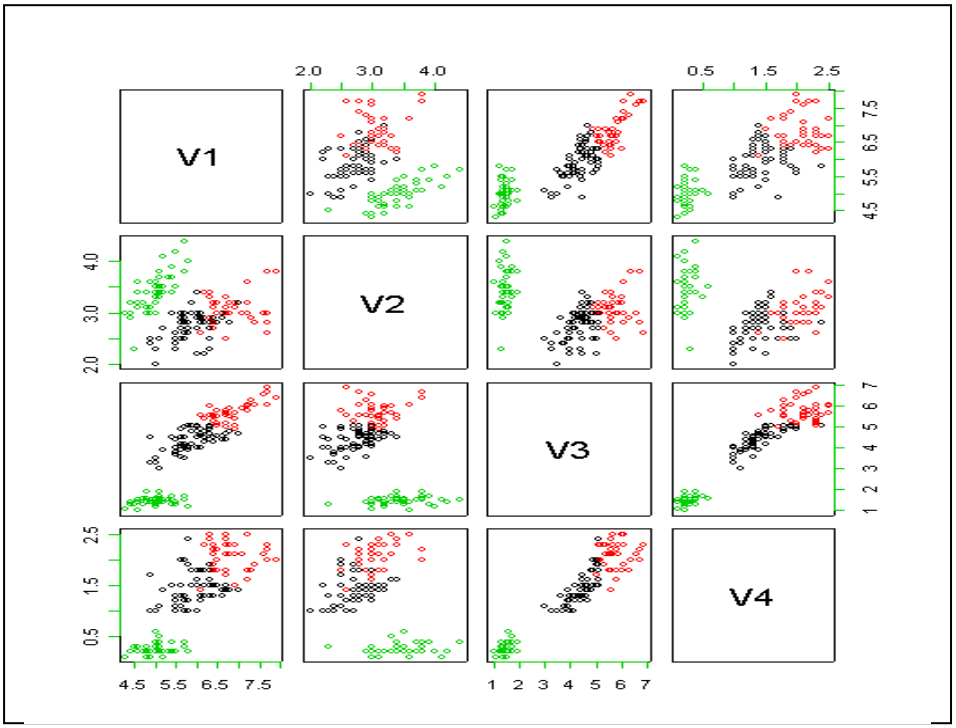
```
> print(km)                  # print result
K-means clustering with 3 clusters of sizes 50, 38, 62

Cluster means:
  Sepal_len Sepal_wid Petal_len Petal_wid
1  5.006000  3.428000  1.462000  0.246000
2  6.850000  3.073684  5.742105  2.071053
3  5.901613  2.748387  4.393548  1.433871

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [37] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [73] 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2 2 3 2
[109] 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 3 2 2 2 2 3 2 2 2 3 2
[145] 2 2 3 2 2 3

Within cluster sum of squares by cluster:
[1] 15.15100 23.87947 39.82097
 (between SS / total SS =  88.4 %)
```

The output gives 3 clusters of sizes 50, 38 and 62. Other items are:

1. *Cluster means* contains the mean vector of these groups.
2. *Clustering vector* is the group label of the cluster obtained from the k-means clustering. Note that the numbering of these group labels is arbitrary. We can re-labeled the first group as “1” instead of “3”.
3. *Within cluster ss* is the within group sum of squares, see (8.1).

We can also plot the observation with color for each group using

```
> plot(d1,col=km$cluster)
```



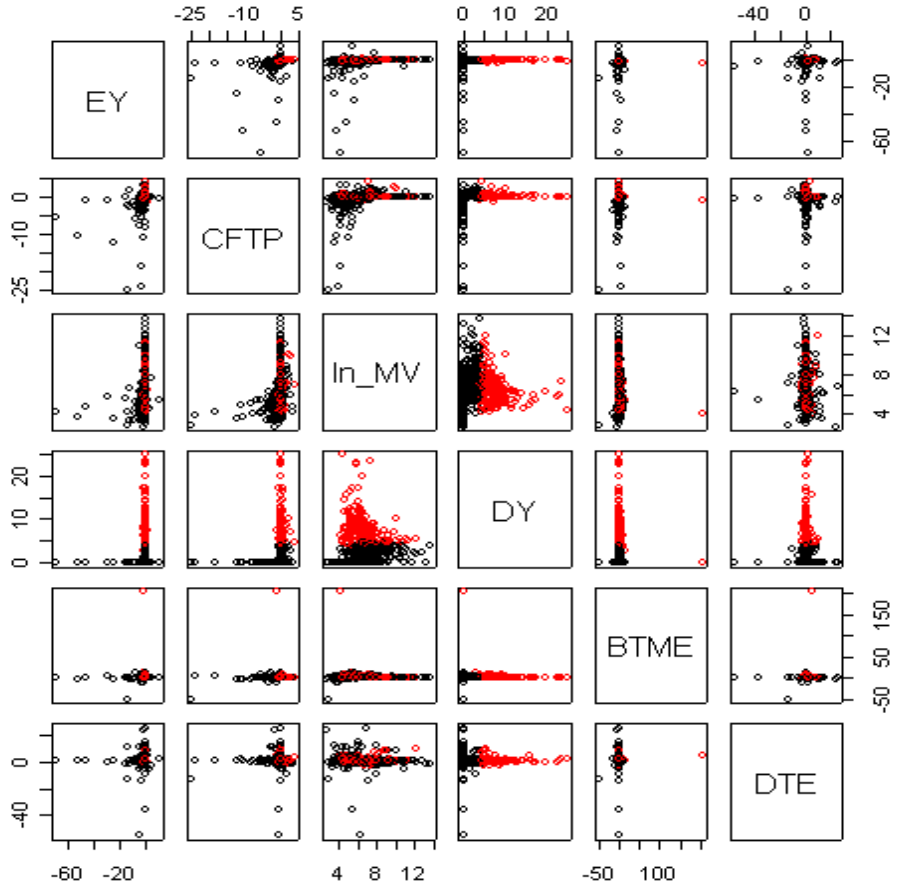
Finally, we can produce the classification table of the label from k-means clustering with the "true" species.

```
> table(km$cluster,d$V5)          # classification table with "true" species
    1  2  3
1  50  0  0
2   0  2 36
3   0 48 14
```

Note that the numbering of the label is arbitrary. Therefore the actual misclassification rate is only *(14+2)/150=10.67%*.

We can apply this K-means clustering to our HSI example.

```
> d<-read.csv("fin-ratio.csv")
> d1<-d[,1:6]                  # save the first 6 columns to d1
> km<-kmeans(d1,2)             # k-means clustering with k=2
> plot(d1,col=km$cluster)      # plot observation with color for each group
```



From this plot, *DY* is the most important and useful variable in this k-means clustering.

8.3 Choosing the suitable *k*

An obviously but difficult question is how to choose the suitable *k* in the k-means clustering. There is a useful statistic to help us choosing a suitable *k* for k-means clustering. Recall that our objective in clustering is to find groups in the data such that the records within the same group should be as homogenous as possible while records between different groups are as different as possible. Then the within group variation and between group variation will be useful statistics. First, we need to understand the output from k-means clustering, especially the within group sum of square statistic.

```
d<-read.csv("iris.csv")      # read in data
x<-d[,1:4]                   # save data to x
km<-kmeans(x,3)              # K-means with k=3
names(km)                    # display items in km

K-means clustering with 3 clusters of sizes 50, 62, 38
Cluster means:
  Sepal_len Sepal_wid Petal_len Petal_wid
1  5.006000  3.428000  1.462000  0.246000
2  5.901613  2.748387  4.393548  1.433871
3  6.850000  3.073684  5.742105  2.071053
Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [37] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [73] 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3
[109] 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 2 3 3 3 2 3
[145] 3 3 2 3 3 2
Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:
[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"

km$betweenss                 # display between group ss
[1] 602.5192
```

Let us find out how this *withinss* and *betweenss* are computed.

```
> d1<-x[km$cluster==1,]      # select group using cluster label
> d2<-x[km$cluster==2,]
> d3<-x[km$cluster==3,]
> n1<-dim(d1)[1]             # save group size
> n2<-dim(d2)[1]             # n1=50, n2=62, n3=38
> n3<-dim(d3)[1]

> apply(d1,2,mean) # compute group mean x̄1
Sepal_len Sepal_wid Petal_len Petal_wid
    5.006     3.428     1.462     0.246
> apply(d2,2,mean) # compute group mean x̄2
Sepal_len Sepal_wid Petal_len Petal_wid
 5.901613  2.748387  4.393548  1.433871
> apply(d3,2,mean) # compute group mean x̄3
Sepal_len Sepal_wid Petal_len Petal_wid
 6.850000  3.073684  5.742105  2.071053
```

```
> (n1-1)*var(d1)                        # compute SSCP matrix for group 1
             Sepal_len Sepal_wid Petal_len Petal_wid
Sepal_len       6.0882    4.8616    0.8014    0.5062
Sepal_wid       4.8616    7.0408    0.5732    0.4556
Petal_len       0.8014    0.5732    1.4778    0.2974
Petal_wid       0.5062    0.4556    0.2974    0.5442

> sum(diag((n1-1)*var(d1)))             # compute tr(SSCP) for each group
[1] 15.151                              sum_{i=1}^{n_j} tr[(x_i1 - x̄_1)(x_i1 - x̄_1)']
> sum(diag((n2-1)*var(d2)))
[1] 39.82097                            sum_{i=1}^{n_j} tr[(x_i1 - x̄_1)(x_i1 - x̄_1)']
> sum(diag((n3-1)*var(d3)))
[1] 23.87947                            sum_{i=1}^{n_j} tr[(x_i1 - x̄_1)(x_i1 - x̄_1)']


m<-apply(x,2,mean)                      # overall mean x̄
m<-matrix(rep(m,3),nrow=3,byrow=T)      # create a matrix whose row is m
dm<-(km$centers-m)^2                    # (group mean - overall mean)^2
colsum<-apply(dm,1,sum)                 # column sum
sum(km$size*colsum)                     # between group ss
[1] 602.5192                            sum_{j=1}^{k} tr[n_j (x̄_j - x̄)(x̄_j - x̄)']
```

$tr[SSW(k)] = \sum_{j=1}^{k}\sum_{i=1}^{n_j} tr[(x_{ij} - \bar{x}_j)(x_{ij} - \bar{x}_j)']$ is the within group sum of square

=sum(km$withinss),

$tr[SSB(k)] = \sum_{j=1}^{k} tr[n_j\,(\bar{x}_j - \bar{x})(\bar{x}_j - \bar{x})']$ is the between group ss = *km$betweenss*,

$\bar{x}_i$ and $\bar{x}$ are the sample mean vectors of group *j* and whole dataset respectively.

The statistic $R(k) = \frac{n-k}{k-1}\frac{tr[SSB(k)]}{tr[SSW(k)]}$ (proposed by Calinski and Harabasz) helps us to choose a suitable *k*, where *n* = total sample size (*=n1+n2+n3=150* in iris example). We want *tr(SSW)* to be as small as possible while *tr(SSB)* to be as large as possible. Therefore, we try several values of *k* so that this statistic *R* is maximized. More accurately, we shall show in the appendix that for univariate samples, it is approximately the normalized $R(k)$ with $F_{k-1,n-k}$ statistics that we are to maximize.

The following function *kmstat*() is to compute this statistic:

```
# function for computing stat in k-mean clustering
# input data matrix x
# try serveral k, choose k so that the stat. is maximum

kmstat<-function(x,k) {
  km<-kmeans(x,k)                                  # k-means clustering with k=3
  ng<-km$size                                      # the number of points of each cluster
  n<-dim(x)[1]                                     # sample size
  ssw<-sum(km$withinss)                            # compute total within group ss
  ssb<-km$betweenss                                # between group ss
  out<-list((n-k)*ssb/((k-1)*ssw),ng,km$cluster) # save output to a list (km$cluster is
                                                   # a vector of integers from 1 to k
                                                   # indicating the cluster to which
                                                   # each point is allocated)
  names(out)<-c("stat","size","cluster")           # apply names to list
  out                                              # output
}
```

Note that this function outputs three items: the R stat, cluster size and the cluster label. We then have to write another function *km*( ) will perform k-means several times and output the best (largest *R* ratio) trial.

```
# Try kmeans(x,k) several times and output the best (largest ratio) trial
# x is the matrix of input variable, k is the no. of clusters
# try is no. of trials

km<-function(x,k,try=5) {          # default no. of trial is 5
  res0<-kmstat(x,k)                # save the result of the first trial
  r0<-res0$stat                    # save the stat from the first trial

  for (i in 2:try) {
    res<-kmstat(x,k)                   # new trial
    if (res$stat>r0) {                 # if new trial is better
      r0<-res$stat                     # update r0 and res
      res0<-res
    }
  }
  cat("cluster size=",res0$size,"\n") # display cluster size
  cat("stat=",res0$stat,"\n")         # display stat
  res0$cluster                        # output cluster label
}
```

Now we read in these functions and try different k.

```
> source("km.r")              # read in km() function
> km2<-km(x,2)                # try k=2
cluster size= 53 97
stat= 513.9245
> km3<-km(x,3)                # try k=3
cluster size= 50 62 38
stat= 561.6278
> km4<-km(x,4)                # try k=4
cluster size= 32 50 28 40
stat= 530.7658
> km5<-km(x,5)                # try k=5
cluster size= 40 28 32 22 28
stat= 459.5058
```

Clearly, *k=3* gives the max. value (*=561.6278*) and hence *k=3* is a suitable choice. Since the cluster label is saved in *km3*, we can produce the following misclassification table:

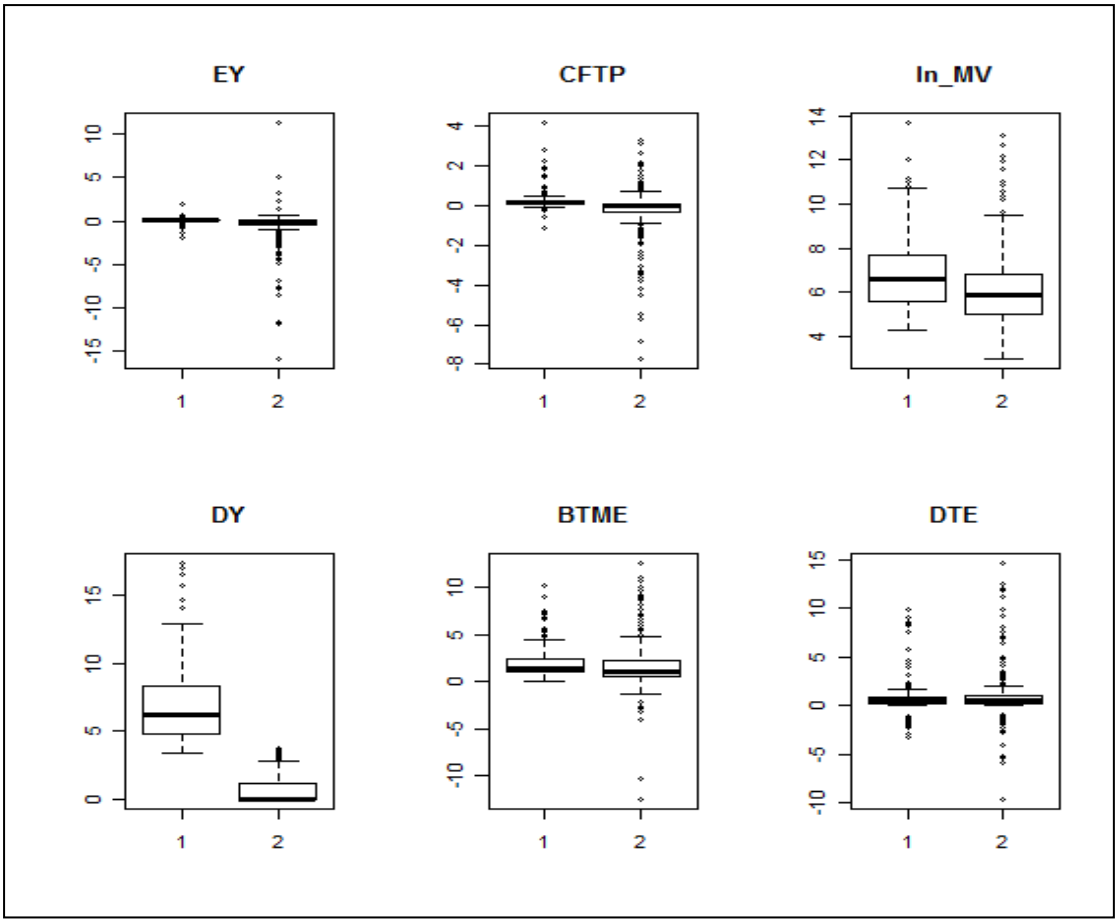
```
> table(km3,d[,5])      # classification table
km3  1  2  3
  1  0  2 36
  2  0 48 14
  3 50  0  0
```

Now we try *km()* using the cleaned financial ratio data "*fin-ratio1.csv*":

```
> d<-read.csv("fin-ratio1.csv")
> x<-d[,1:6]
> km2<-km(x,2)                    # try k=2
cluster size= 185 473
stat= 330.7535
> km3<-km(x,3)                    # try k=3
cluster size= 373 186 99
stat= 247.1816
> km4<-km(x,4)                    # try k=4
cluster size= 91 20 370 177
stat= 233.2413
> km5<-km(x,5)                    # try k=5
cluster size= 45 325 165 104 19
stat= 239.908
```

k=2 gives the max. value (*330. 7535*). Note that this classification needs not be according to *HSI*. One major objective is to find out the characteristics of these clusters. The following box-plots (named after George Box) are an effective graphical method to find out the characteristics of each cluster.

```
lab<-names(x)                     # save var. names
par(mfrow=c(2,3))                 # define 2x3 multi-frame graphic
boxplot(x[,1]~km2,main=lab[1])    # boxplots for each variable
boxplot(x[,2]~km2,main=lab[2])
boxplot(x[,3]~km2,main=lab[3])
boxplot(x[,4]~km2,main=lab[4])
boxplot(x[,5]~km2,main=lab[5])
boxplot(x[,6]~km2,main=lab[6])
```



From these boxplots, it is clear that *DY* in 2nd group is lower than that of 1st group while there is no significant different in other variables. Therefore we can simply describe 2nd group as the collection of stocks with lower *DY*.

8.4 Dis-similarity measures

Note that the K-mean clustering as well as other clustering methods is based on the distance (or dis-similarity) between observations. Therefore it is important to define distance between two observations. Since there may be many different types of variables in our data set, the distance or dissimilarity between objects (observations) should be defined differently according to their variable type. Suppose that we have p variables in our data set and the object *i* and *j* denoted by

object	*i*	x_{i1}	$\cdots$	x_{ip}
object	*j*	x_{j1}	$\cdots$	x_{jp}

- **Continuous variables**

If all the variables are continuous, we can use either one of the following as the distance measure:

Euclidean : $d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + \ldots + (x_{ip} - x_{jp})^2}$

City-Block: $d(i,j) = |x_{i1} - x_{j1}| + \ldots + |x_{ip} - x_{jp}|$

Minkowski: $d(i,j) = [|x_{i1} - x_{j1}|^q + \ldots + |x_{ip} - x_{jp}|^q]^{1/q}$

Ordinal variables

If all the variables are ordinal, we can rescale the observation to (0,1) by
$z_{ik} = (r_{ik} - 1)/(M_k - 1)$ *and* $z_{jk} = (r_{jk} - 1)/(M_k - 1)$.
where variable k has M_k levels with rank r_k. Then treat them like continuous variables.

Binary variables

If all the variables are binary, we can use either one of the following measures:

- Simple matching coefficient:
  d(i,j) = (r+s)/(q+r+s+t) = (r+s)/p
- Jaccard coefficient:
  d(i,j) = (r+s)/(q+r+s)

		Object j	
		1	0
Object i	1	*q*	*r*
	0	*s*	*t*

Nominal (Categorial) variables

If all the variables are nominal, we can use
d(i,j) = (p-m)/p where m is the number of matches among these p variables.

Mixed type, Missing values and scaling of data

In practice, data set usually contains missing values and variables of mixed type. The easiest way to handle the missing data is to delete all the records with missing values. R has a build-in function *complete.cases()* to select all the complete cases. However, using only the complete cases will throw away lots of information and more seriously may lead to a biased sub-sample. A better but more complicated way to handle missing values is to use available variables to compute the distance between two records.

Another issue is that the range of the variables may differ a lot. Using the original scale may put more weights on the variables with large range. Therefore, we better re-scale them into the [0,1] interval before performing clustering analysis. Define

$$d(i,j) = \sum_{k=1}^{p} w_{ij}^{(k)} d_{ij}^{(k)} \Big/ \sum_{k=1}^{p} w_{ij}^{(k)},$$

where $w_{ij}^{(k)} = 0$ if x_{ik} or x_{jk} is missing; $w_{ij}^{(k)} = 1$ otherwise. Also,
If variable *k* is binary or nominal, $d_{ij}^{(k)} = 0$ if $x_{ik} = x_{jk}$; otherwise $d_{ij}^{(k)} = 1$;

If variable *k* is continuous, $d_{ij}^{(k)} = | x_{ik} - x_{jk} | / (\max x_{uk} - \min x_{uk})$;

If variable *k* is ordinal, we first rescale r_{ik} to [0,1] by $z_{ik} = (r_{ik} - 1)/(M_k - 1)$ and compute $d_{ij}^{(k)} = | z_{ik} - z_{jk} |$.

8.5 K-means using EXCEL

Although the k-means clustering algorithm is iterative in nature, we can find the solution by using the *solver()* function to minimize $E(z)$ in equation (8.1). We use the iris data as our example.

1. The data is in cells B6 to F156.
2. The cluster center c1, c2 and c3 for group 1 to 3 are in B2 to E4. We may first use the 1st, 51st and 101st observation as the center for group1 to 3. That is, copy B7:E7 to B2:E2; B57:E57 to B3:E3; and B107:E107 to B4:E4.
3. We compute x-c1 by entering =B7-B$2 in H7 and copy it to K156. Similarly we compute x-c2 by entering =B7-B$3 in M7 and copy it to P156; and compute x-c3 by entering =B7-B$4 in R7 and copy it to U156.
4. We compute the squared Euclidean distance of the 1st observation to c1 by entering =SUMSQ(H7:K7) in W7. Similarly we compute the squared distance to c2 by entering =SUMSQ(M7:P7) in X7 and the squared distance to c3 by entering =SUMSQ(R7:U7) in Y7. Then we copy W7:Y7 down to W156:Y156.
5. Now we find the minimum distance by entering =MIN(W7:Y7) in AA7 and copy it down to AA156.
6. We obtain the class label by entering =1*(W7=AA7)+2*(X7=AA7)+3*(Y7=AA7) in Z7. This gives us the group label such that it is closest to the 1st observation. We copy it down to Z156.
7. We compute the sum of all these minimum squared distance by entering =SUM(AA7:AA156) in AA158. This the objective function value we need to minimize using solver().
8. We use solver and enter $AA$158 as the objective cell to minimize and $B$2:$E$4 as the variable cells. Click solve to obtain the clustering result.
9. Finally we can obtain the cluster size for group 1 by entering =COUNTIF($Z$7:$Z$156,"=1"). We can find cluster size for other groups similarly.

Reference:

Sections 12.1, 12.2, 12.4 and 12.8 of Applied Multivariate Statistical Analysis, 5th ed., Richard Johnson and Dean Wichern, Prentice Hall.

Appendix: To relate *R*(*k*) statistics to the F-distribution for univariate samples

We show that for univariate samples x_i, $i = 1, 2, ..., n$,

$$R(k) = \frac{n-k}{k-1} \frac{tr[SSB(k)]}{tr[SSW(k)]} \sim F_{k-1,n-k}$$

Let $\mu_j := \mathrm{E}(x_{ij})$, while $\bar{x}_j = \frac{1}{n_j}\sum_{i=1}^{n_j} x_{ij}$

$$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\mu_j)^2$$

$$=\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\bar{x}_j+\bar{x}_j-\mu_j)^2$$

$$=\sum_{j=1}^{k}\sum_{i=1}^{n_j}[(x_{ij}-\bar{x}_j)^2+2(x_{ij}-\bar{x}_j)(\bar{x}_j-\mu_j)+(\bar{x}_j-\mu_j)^2]$$

$$=\sum_{j=1}^{k}\left[\sum_{i=1}^{n_j}(x_{ij}-\bar{x}_j)^2+n_j(\bar{x}_j-\mu_j)^2\right]$$

$$=tr[SSW(k)]+\sum_{j=1}^{k}\left[n_j(\bar{x}_j-\mu_j)^2\right]$$

Since $\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\mu_j)^2 \sim \chi^2_n$, $\sum_{j=1}^{k}\left[n_j(\bar{x}_j-\mu_j)^2\right]\sim \chi^2_k$ and they are independent,

hence $tr[SSW(k)] \sim \chi^2_{n-k}$.

Using $\mu=\mathrm{E}(x_i)$, $\bar{\bar{x}}=\frac{1}{n}\sum_{j=1}^{k} n_j\overline{x_j}$ and $n=\sum_{j=1}^{k} n_j$, we can prove that

$$\sum_{j=1}^{k} n_j(\bar{x}_j-\mu)^2$$

$$=\sum_{j=1}^{k} n_j(\bar{x}_j-\bar{x}+\bar{x}-\mu)^2$$

$$=\sum_{j=1}^{k} n_j\left[(\bar{x}_j-\bar{x})^2+2(\bar{x}_j-\bar{x})(\bar{x}-\mu)+(\bar{x}-\mu)^2\right]$$

$$=\left[\sum_{j=1}^{k} n_j(\bar{x}_j-\bar{x})^2\right]+n(\bar{x}-\mu)^2$$

$$=tr[SSB(k)]+n(\bar{x}-\mu)^2$$

Since $\sum_{j=1}^{k} n_j(\bar{x}_j-\mu)^2 \sim \chi^2_k$, $n(\bar{x}-\mu)^2 \sim \chi^2_1$ and they are independent, hence

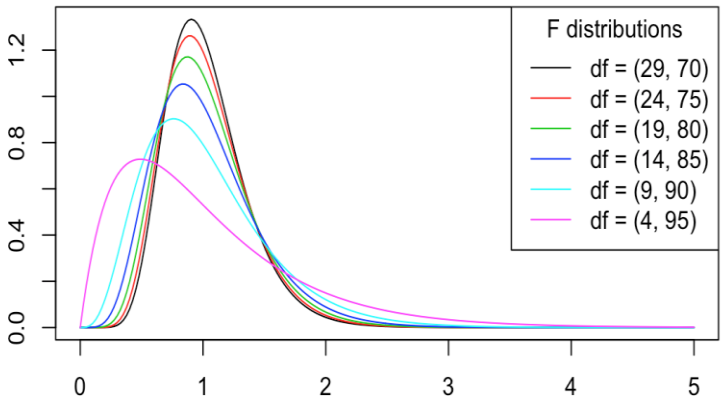
$tr[SSB(k)] \sim \chi^2_{k-1}$.

While the $tr[SSB(k)]$ and $tr[SSW(k)]$ are also independent,

$$R(k)=\frac{tr[SSB(k)]/(k-1)}{tr[SSW(k)]/(n-k)} \sim F_{k-1,n-k},$$

for samples drawn from normal distribution for each *k*-clusters, and the centroids of all clusters are also normally distributed. For multivariate sample, the Wishart distribution should be used instead of the chi-square distribution.

In determining the optimal number of clusters *k*, it is more accurate to maximize the normalized statistics $R(k) / F_{k-1,n-k}$, than the un-normalized $R(k)$.



F distributions	Upper 5% quantile
df=(29, 70)	1.6294
df=(24,75)	1.6633
df=(19,80)	1.7180
df=(14,85)	1.8099
df=(9,90)	1.9856
df=(4,95)	2.4675