Module 11. Clustering Analysis

Overview:

In this module we will learn clustering analysis. Clustering analysis is what is known as an "unsupervised learning technique" grouping data into similar groups. We will learn the objective and several standard methods of clustering analysis.

We will start with the basic concepts of clustering analysis. We describe the two approaches of clustering analysis: partitional and hierarchical. Then we will discuss several different distance measures.

We will then describe a hierarchical clustering method in more detail, and how different linkages can be used. After that, we will describe the partitional clustering methods: K-Means and K-Medoids. We also will show the effect of using different distance measures, and consider the choice of the number of clusters.

By the end of this module, you should be able to conduct the hierarchical clustering (e.g., single linkage clustering), K-Means and K-Medoids clustering in R. You should be able to use and interpret the R outputs. You should also know how to use different distance measures in the clustering analysis. You should be able to use clustering analysis on microarray and other data to discover interesting groups.

Learning Objectives

1. Apply **clustering analysis** to group data with various **distance measures**
2. Conduct **hierarchical** clustering with various **linkage** functions
3. Conduct **K-Means** and **K-Medoids** clustering, and choose the appropriate number of clusters

Readings:

Statistics Using R with Biological Examples pages 310-318.

Applied Statistics for Bioinformatics using R pages 117-130.

Lesson 1: Clustering Analysis and Distance Measures

Objectives

By the end of this lesson you will have had the opportunity to:

- Compare and contrast two types of **clustering** approaches: **partitional** and **hierarchical**
- Evaluate several **distance measures** in clustering

Overview

In this lesson, we introduce the basic concepts of clustering analysis. Two basic approaches: partitional clustering and hierarchical clustering are illustrated on an example data set. These two approaches will be studied in detail in the next two lessons. The results of clustering analysis depends on the distance measure used, thus we will learn the definitions of several commonly used distance measures.

Clustering Analysis

The aim of cluster analysis is to divide observations into groups (clusters) that are more similar within the group. Biologists had been earlier users of clustering analysis to find classification structures in all living things. In bioinformatics, it is natural to use clustering to find groups of genes that have similar functions. By finding genes with similar expression patterns, the clusters provide interesting candidate classes for potential new scientific discovery of genetic functions.

The clustering analysis belongs to a category of so-called unsupervised learning techniques. To understand what *supervised learnings* versus *unsupervised learnings* are, it is helpful to think about the regression analysis in an earlier module. For a random sample of n $(X_1, Y_1), \ldots, (X_n, Y_n)$, the objective is to find a relationship f(X) that best predicts the Y values. We use Y to supervise our learning for f(X), also to check if f(X) really predicts Y well. In clustering analysis, in contrast, we only have observations $X_1, \ldots, X_n$ without a response variable Y. In this case, we are *unsupervised* since we lack a response variable to supervise our analysis. As a result, unsupervised learning is more challenging because of the lack of a clear quantitative objective. We may think that we separated the data into interesting groups, but the real group memberships are unknown. Therefore, we cannot check if our assigned group memberships are really good or an illusion, due to the lack of supervision.

We illustrate the clustering analysis on the iris data set we used previously.

K-means Clustering on Iris Data

The iris data set comes with R installation. It contains five variables: sepal length, sepal width, petal length, petal width, and species of the flowers. Without knowing the species beforehand, we use clustering analysis to group the observed flowers into three groups, based on the characteristics contained in the first four variables. We used the K-means clustering which will be covered in a later lesson. The R script is below:

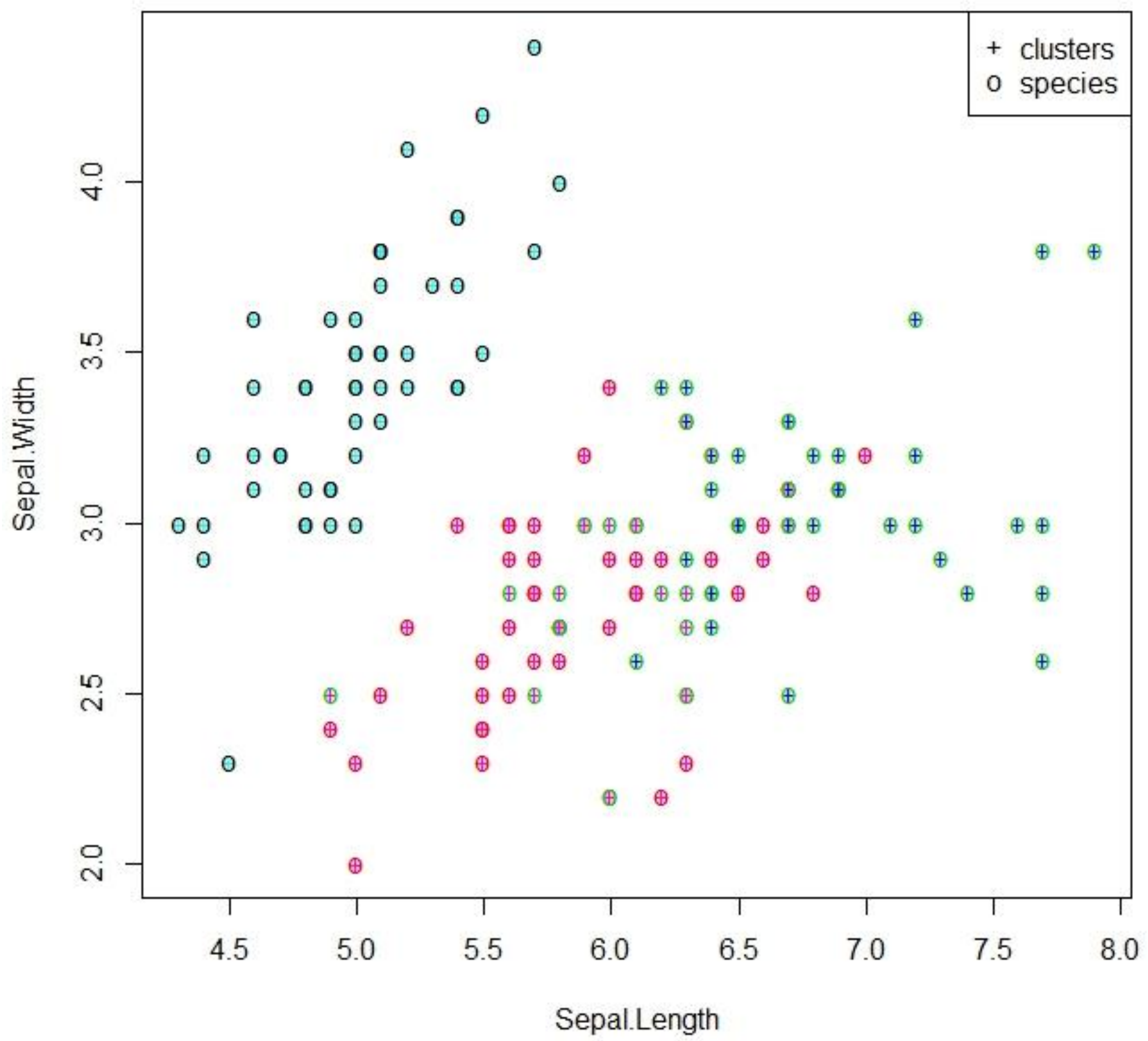
```
> iris2s2 <- iris; iris2$Species <- NULL
> clusters.km <- kmeans(iris2, 3)
> table(iris$Species, clusters.km$cluster)
                       1      2      3
   setosa              0     50      0
   versicolor          2      0     48
   virginica          36      0     14
```

We can see that the K-means clustering separate data into three groups roughly corresponding to the three species. This shows what we would like our unsupervised learning to do: not knowing the true group memberships (species), we find the groups simply through data.

We plot the clustering results, and overlay with the species information on a scatterplot of the first two variables on the next page.

K-means Clustering on Iris Data, Graphic Display

```
plot(iris2[c("Sepal.Length", "Sepal.Width")], col = clusters.km$cluster,
pch="+")

points(iris[c("Sepal.Length", "Sepal.Width")], col = iris$Species, pch="o")
```



We can see that the data was partitioned into three groups (approximately) according to the flower species. This type of clustering analysis is called **partitional clustering analysis**. Sometimes, the species may be further divided into subspecies. To reflect those types of structures, hierarchical (HC) clustering is needed. We illustrate the HC clustering on the iris data set next.

Hierarchical (HC) Clustering on Iris Data

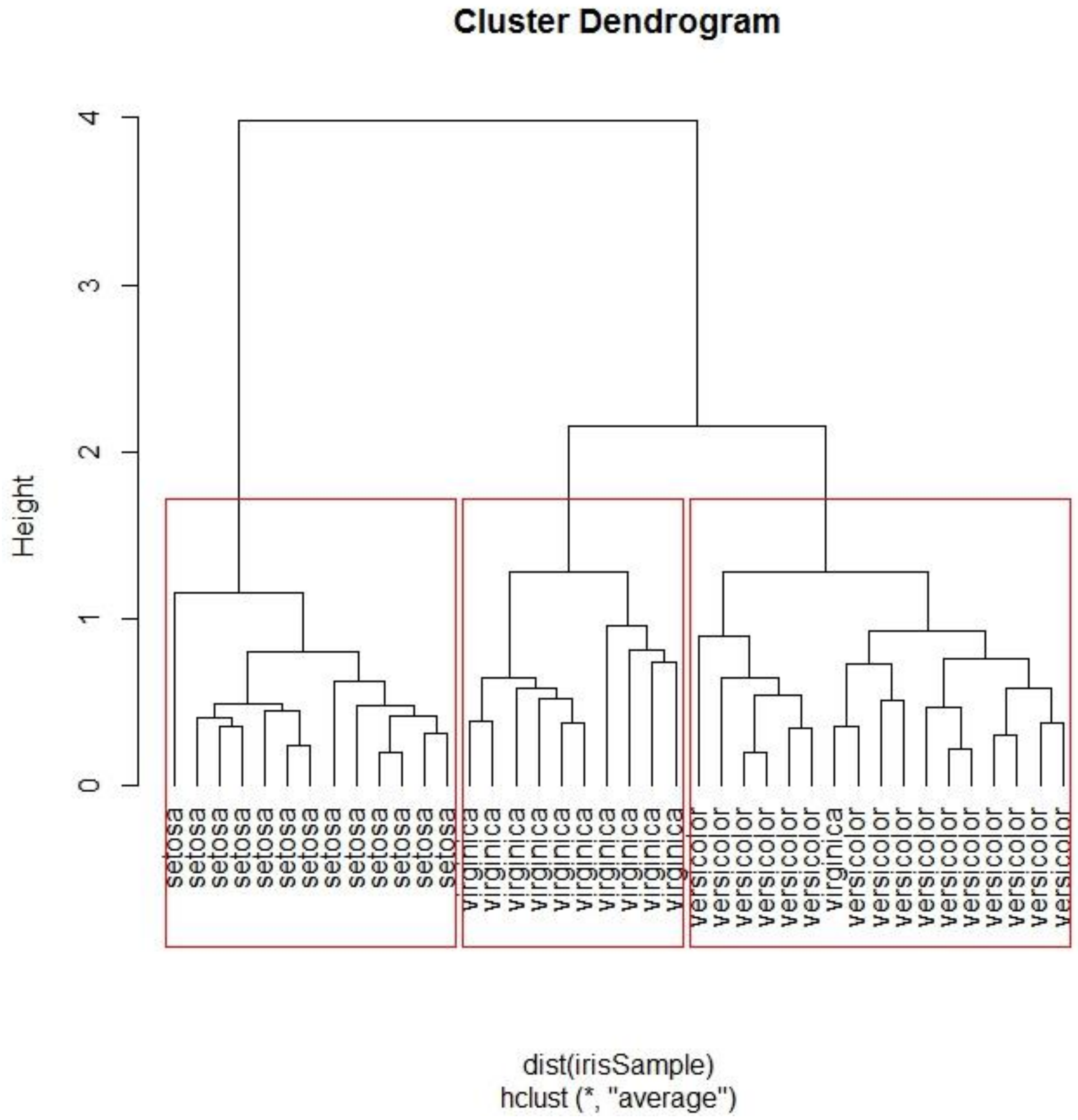
We clustered the data into K = 3 groups in the previous analysis. Since we do not know the appropriate number of groups beforehand in practice (information such as the species will not be available in a real clustering analysis situation), different number of K groups need to be tried. However, there may be no clear relationship between the K = 3 groups and the K = 4 groups we get at the end.

Hierarchical (HC) clustering provides a sequence of partitions from the finest to the coarsest. The finer partitioned groups are always subsets of the coarser partitions. For example, two of the K = 3 groups will remain the same at the K = 4 partition level, with the last group further divided into two subgroups. This nested structure provides easier interpretable results. The following R commands are used to produce HC clustering on the iris data.

```
idx <- sample(1:dim(iris)[1], 40)

irisSample <- iris[idx,]; irisSample$Species <- NULL

hc <- hclust(dist(irisSample), method="ave")

plot(hc, hang = -1, labels=iris$Species[idx])

rect.hclust(hc, k=3)

groups <- cutree(hc, k=3)    # cut tree into 3 clusters
```

The HC clustering results in a tree structure of the partitions that is often graphically displayed as a **cluster dendrogram**. We display the dendrogram in the next page. We only took a random subset of 40 flowers in the iris data set to get a better display in the dendrogram. We also cut the tree at K = 3 level to see its relationship to the species.

HC Clustering on Iris Data, Graphic Display



We can see that “setosa” iris are clustered into one group, the other two clusters have some overlap in the two remaining species.

The tree structure provides more information than a simple partition at K = 3 level. For example, we can see that virginica and versicolor species are more similar with each other than with setosa.

Distance Measures

The clustering results depend very much on the notation of distance: when are two data points considered close. The data points are generally represented as vectors. For example, in the iris data example, each flower is represented as a four dimensional vector (containing the four features used in clustering: sepal length, sepal width, petal length, petal width). Here we describe several commonly used distance measures.

We consider the distance between two vectors $\vec{x} = (x_1, ..., x_n)$ and $\vec{y} = (y_1, ..., y_n)$.

- The **Euclidean distance** (L_2 norm) is the square root of sum of squared differences between corresponding elements: $d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$.
- The **Manhattan distance** (L_1 norm) is the sum of absolute differences between corresponding elements: $d(\vec{x}, \vec{y}) = \sum_{i=1}^{n} | x_i - y_i |$.
- The **Max component distance** (L_∞ norm) is the maximum absolute difference between corresponding elements: $d(\vec{x}, \vec{y}) = \max_{1 \le i \le n} | x_i - y_i |$.

The distance measures are also often referred as the dissimilarity measures in the clustering literature. We use both terms interchangeably in this class.

Distance (Dissimilarity) Measures Calculation

Example: The distances between two vectors $\vec{x} = (1,6,5,9)$ and $\vec{y} = (11,3,2,17)$.

The Euclidean distance is $d(\vec{x},\vec{y}) = \sqrt{(1-11)^2 + (6-3)^2 + (5-2)^2 + (9-17)^2} = 13.49$, calculated in R by

```
x<-c(1,6,5,9); y<-c(11,3,2,17)
sqrt(sum((x-y)^2))
```

The Manhattan distance is $d(\vec{x},\vec{y}) = |1-11| + |6-3| + |5-2| + |9-17| = 24$, calculated in R by

```
sum(abs(x-y))
```

The Max component distance is $d(\vec{x},\vec{y}) = \max(|1-11|,|6-3|,|5-2|,|9-17|) = 10$, calculated in R by

```
max(abs(x-y))
```

In R, the `dist()` function calculates the pairwise distances between rows of a matrix. For microarray expression data, the rows correspond to genes. So `dist()` will calculate the pairwise distances between genes, which can then be used for gene clustering analysis.

Demonstration: Genes Distances in Golub Data

We can use dist() to find the Euclidian distances among the first five genes in the Golub et al. (1999) data as

```
> data(golub, package="multtest")
> dist(golub[1:5,], method="eucl")
          1         2         3         4
2  2.834725
3  9.208276  7.189650
4 16.706824 15.556355 12.083020
5 14.854179 13.831896 11.368185  4.650899
```

So the distance between the first gene and the second gene is 2.835. The distance between the 3rd and 5th genes is 11.37.

We can similarly get the Manhattan distances as

```
> dist(golub[1:5,], method="manh")
         1        2        3        4
2 11.76841
3 52.81313 42.08696
4 79.78477 76.85826 62.76704
5 66.95679 65.22418 61.00142 19.67256
```

Demonstration: Find Closest Genes in ALL Data

We often want to select genes close to a target gene. This can be done with the genefinder() function in R. We find three genes from the ALL data (Chiaretti et al., 2004) closest to the gene with identifier 1389_at as shown here:

```
library("genefilter"); library("ALL"); data(ALL)
> closest <- genefinder(ALL, "1389_at", numResults=3, method =
"euclidean")

> featureNames(ALL)[closest[[1]]$indices]

[1] "32629_f_at" "1988_at"      "36571_at"
```

We see that the 32629_f_at gene has smallest Euclidean distance from the 1389_at gene. Similarly we can use other distances to find the closest genes.

```
> closest.max <- genefinder(ALL, "1389_at",    numResults=3, method =
"maximum")

> featureNames(ALL)[closest.max[[1]]$indices]

[1] "1988_at"      "31396_r_at" "36986_at"

> closest.manh <- genefinder(ALL, "1389_at",    numResults=3, method =
"manhattan")

> featureNames(ALL)[closest.manh[[1]]$indices]

[1] "39756_g_at" "32629_f_at" "39168_at"
```

Notice that the closest genes do change with the distance measures used. So the clustering analysis results can also be affected by the choice of distance measures.

Lesson Summary

This lesson introduced the basic concepts of clustering analysis. We learned the meaning of two types of clustering analysis: hierarchical clustering and partitional clustering. We also learned several distance definitions, and how to calculate these distances in R.

Next lesson covers hierarchical clustering methods.

Lesson 2: Hierarchical Clustering

Objectives

By the end of this lesson you will have had the opportunity to:

- Perform **single linkage clustering**
- Perform **complete, average, and Ward** linkage clustering

Overview

In this lesson, we focus further on the hierarchical clustering analysis. Particularly, we will start with single linkage clustering. Then we introduce three other linkage functions for hierarchical clustering. These methods will be illustrated on the ALL data set.

Single Linkage Clustering

The hierarchical clustering provides the clusters partition of the data for K = 1, 2, …, N groups, with N as the total number of the data points. *The clusters of a finer partition are always the subclusters of those of a coarser partition.* This restriction makes the results easier to interpret without specifying the number of clusters K beforehand.

There are two ways to do hierarchical clustering: **bottom-up** or **agglomerative clustering** and **top-down** or **divisive clustering**. Let us look at them in more detail:

(1) Agglomerative clustering starts with each data point as a cluster of single elements. Then the clusters are successively merged until all data points become one cluster.

(2) The divisive clustering goes the opposite way, starting with all data points as one cluster, and divides the clusters into subclusters, until eventually each data point forms a separate one-point cluster. The agglomerative clustering is the most common type used, and we will focus on the algorithms of this type.

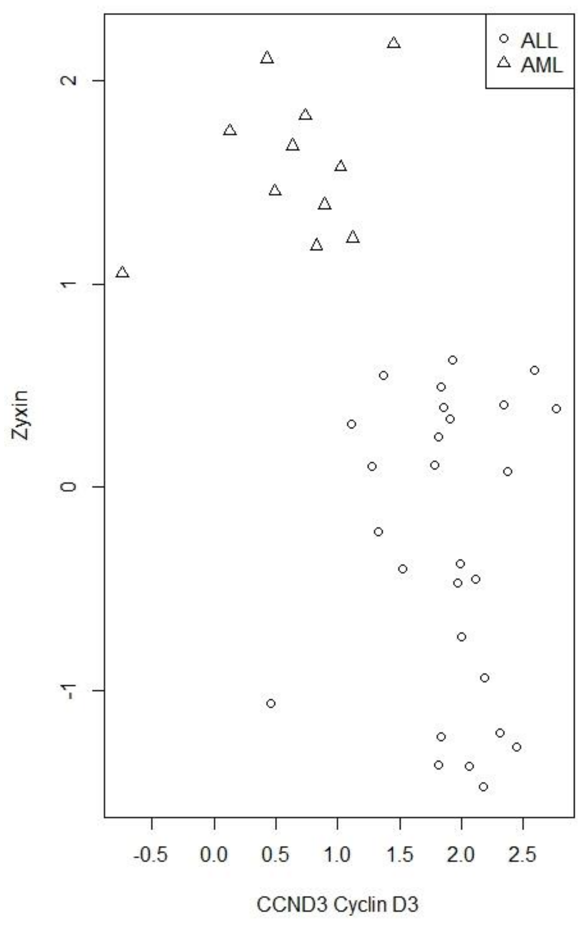
We first describe one agglomerative clustering method: the single linkage clustering. As all agglomerative clustering procedures, we start with each data point as a cluster of single elements, and proceed to combine the two nearest (most similar) clusters in successive steps. At beginning, each cluster *I* is simply a single point { x_i }. The distance (dissimilarity) between two clusters are simply the distance between the two points $dist(I,J) = dist(x_i, x_j)$. When there are more points in the clusters, the single linkage clustering define the distance (dissimilarity) of two clusters as the distance between the nearest neighbors in those clusters. That is, $dist(I,J) = \min_{i,j}\{dist(x_i, x_j) : x_i \ in \ I; x_j \ in \ J\}$. Then at each step, the two clusters closest under this distance definition are merged. The process is continued until all points are joined in one cluster.

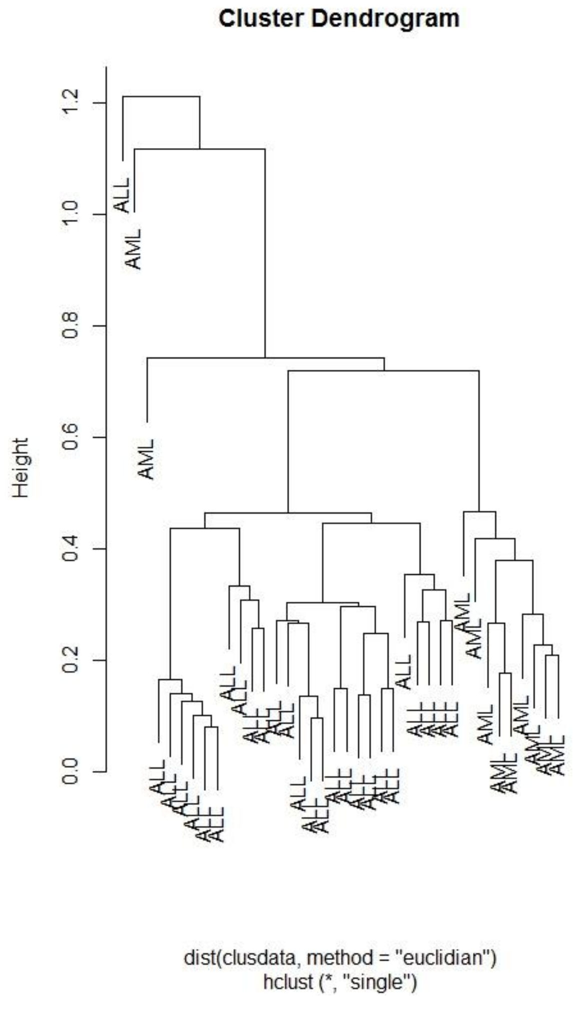
We illustrate this approach on the ALL data.

Demonstration: Single Linkage Clustering on Golub Data

We now apply single linkage cluster analysis to the Golub (1999) data. This is the Example 3 on page 124 of Applied Statistics for Bioinformatics using R.

Example: Recall that the first twenty seven patients belong to ALL and the remaining eleven to AML. Also we found in earlier modules that the expression values of two genes "CCND3 Cyclin D3" and "Zyxin" differ between the patient groups ALL and AML. The left figure below shows how the patient groups differ with respect to these two gene expression values. A single linkage cluster analysis dendrogram is shown by the right figure below. We can see that, except three cases each as a single cluster, the AML and ALL patients are separated into two clusters.





R Script of Single Linkage Clustering on Golub Data

The following R script produces the two figures in last page.

```
data(golub, package="multtest")
clusdata <- data.frame(golub[1042,],golub[2124,])
colnames(clusdata)<-c("CCND3 Cyclin D3","Zyxin")
gol.fac <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
par(mfrow=c(1,2))
plot(clusdata, pch=as.numeric(gol.fac))
legend("topright",legend=c("ALL","AML"),pch=1:2)
plot(hclust(dist(clusdata, method="euclidian"), method="single"),
labels=gol.fac)
```

More Hierarchical Clustering Methods

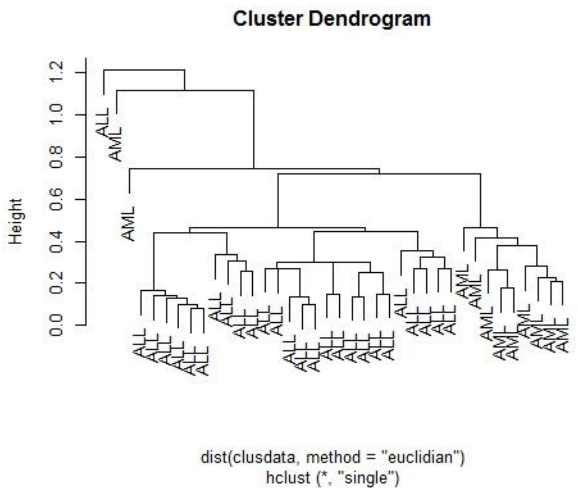
Besides the single linkage, there are other commonly used agglomerative hierarchical clustering methods. In the **complete linkage clustering**, the distance between two clusters are defined by the largest distance between a point in the first cluster and a point in the second cluster. This is in contrast to the minimum distance used in the **single linkage clustering**. The **average linkage clustering** uses the average distance between the points in the first cluster and the points in the second cluster. See Figure 16.16 on page 317 of Statistics Using R with Biological Examples for an illustration of these three types of cluster distance. Another commonly used method is the **Ward linkage clustering**, where in each step we choose the two clusters to merge so as to minimize SSE -- the within clusters sums of squared deviations from the cluster means. This is easiest to see under the Euclidean distance. Ward method aims to minimize $SSE = \sum_{k=1}^{K} \sum_{i \in I_k} (x_i - \bar{x}_k)^2$ where $\bar{x}_k$ denotes the mean of the k-th cluster I_k, and $(x_i - \bar{x}_k)^2$ is the squared Euclidean distance.

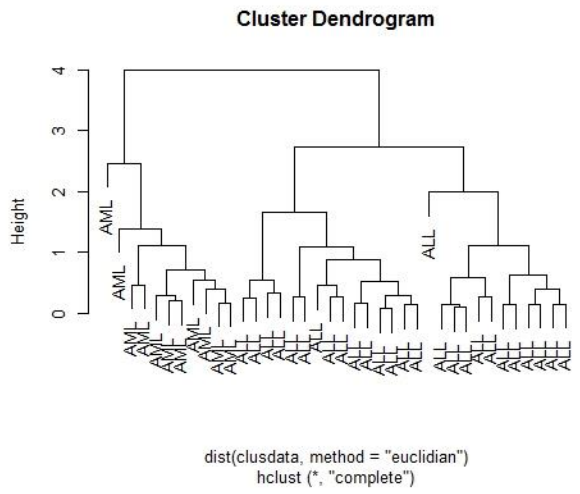
The **single linkage**, using nearest neighbors, will string the data points to form clusters. Hence, the single linkage clustering tends to form clusters of long “chains”. The **complete linkage clustering** works well when the data points form naturally distinct "clumps", and not so well when the clusters are somehow elongated or of a "chain" type nature. The **average linkage clustering** is in between those two methods, and can work in both “clumps” and “chains” types of clusters. The **Ward linkage** mathematically is equivalent to a special form of average linkage which define the distance between two clusters using the average of *squares* of the inter-cluster pairwise distances.

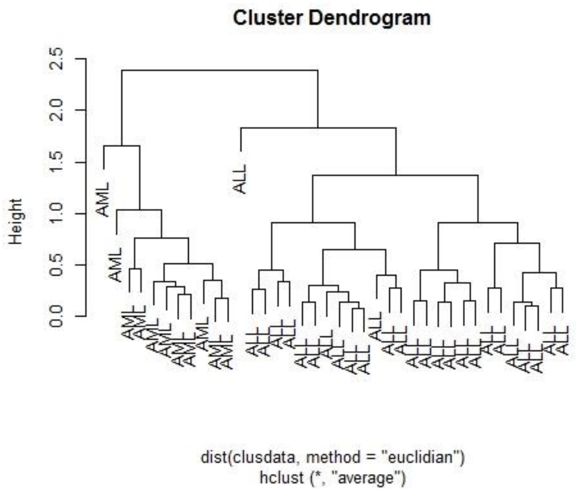
Next, we illustrate these four types of hierarchical clustering on the Golub data.

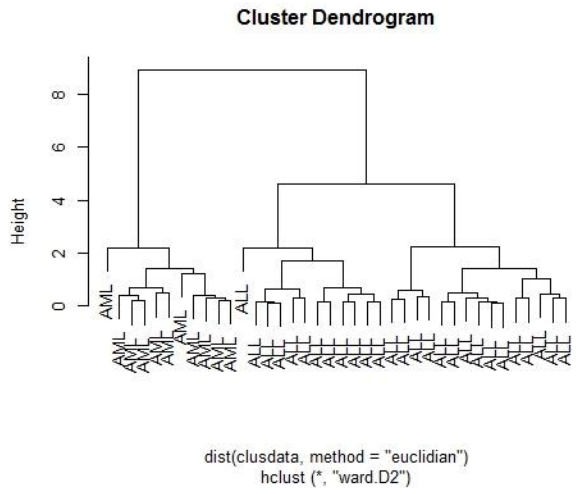
Comparison of the Four Linkage Clustering Methods on Golub Data

Recall that the ALL and AML patients form two natural clumps in the scatterplot. Hence the other three methods should all work better than the single linkage clustering before. The following dendrograms confirm this intuition. All three clustering methods other than the single linkage separates the AML and ALL patients into two distinct clusters.









R Script for Clustering Methods Comparison on Golub Data

The plots on previous page are produced by the following R code

```
data(golub, package="multtest")
clusdata <- data.frame(golub[1042,],golub[2124,])
gol.fac <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
hcALL.sing<-hclust(dist(clusdata,method="euclidian"),method="single")
hcALL.comp<-hclust(dist(clusdata,method="euclidian"),method="complete
")
hcALL.ave<-hclust(dist(clusdata,method="euclidian"),method="average")
hcALL.ward<-hclust(dist(clusdata,method="euclidian"),method="ward.D2"
)
par(mfrow=c(2,2))
plot(hcALL.sing, labels=gol.fac)
plot(hcALL.comp, labels=gol.fac)
plot(hcALL.ave, labels=gol.fac)
plot(hcALL.ward, labels=gol.fac)
```

Using Different Distance Measures in Clustering

We can use distance measures other than Euclidean distance. In the previous code, replace dist(clusdata,method="euclidian") by dist(clusdata,method="manhattan") and dist(clusdata,method=" maximum") respectively to use the Manhattan distance and Max component distance.

You can run those analysis on the ALL data yourself. The results are about the same for all distance measures under the average linkage, complete linkage and Ward linkage clustering. There is some small change in single linkage clustering.

In this case, the two patient groups (ALL and AML) are well separated in these two genes expressions. So the choice of distance measures do not matter, while the choice of linkage functions has more effect on the clustering results.

Summary

This lesson introduces four common agglomerative hierarchical clustering methods. We covered the basic definitions of each approach, and briefly discussed the situations where each method works well. You should be able to carry out these clustering analyses in R.

In next lesson, we cover the K-means clustering method.

Lesson 3: K-Means and K-Medoids Clustering

Objectives

By the end of this lesson you will have had the opportunity to:

- Fit **K-Means clustering** in R
- **Select the number of clusters** with SSE plot
- Fit **K-Medoids clustering** in R
- Using different **distance measures** for clustering in R

Overview

In this lesson, we focus on the partitional clustering analysis. We first introduce the most commonly used K-means clustering method. We illustrate the method on two data sets. The selection of clusters is briefly discussed. We then consider a robust alternative method of K-medoids clustering. Also, we show the effect of using different distance measures on the iris data set.

K-means Clustering

As mentioned in the first lesson of this module, K-means is a partitional clustering method. In contrast to the hierarchical clustering, we do not care about the hierarchical structure. That is, we do not care about how a finer partition relates to a coarser partition. Rather, for a fixed number K clusters, we aim to find the **optimal partition**.

For the optimal partition, we need to define a criterion to be optimized. For K-means clustering, we want to find the K clusters that minimize the $SSE = \sum_{k=1}^{K} \sum_{i \in I_k} (x_i - \bar{x}_k)^2$. This is the same criterion used by Ward's method in the hierarchical clustering.

Unfortunately, it is very hard computationally to find the best possible K clusters that minimizes SEE (To be exact, the problem is NP-hard in computer science language). The commonly used implementation uses an algorithm to find local optimum solution. The algorithm starts from a random partition of the data points into K clusters.

Step (1): For each of the K clusters, compute the cluster mean $\bar{x}_k = \frac{1}{n_k} \sum_{i \in I_k} x_i$. Note that the means $\bar{x}_k$ and data points x_i are vectors here. The mean $\bar{x}_k$ is called the *centroid* in clustering.

Step (2): For each x_i of the n data points, reassign its cluster membership to the k-th cluster whose centroid is closest (that is, $(x_i - \bar{x}_k)^2$ is smallest among $\bar{x}_k$ s).

Repeat steps (1) and (2) until there are no changes in the cluster memberships.

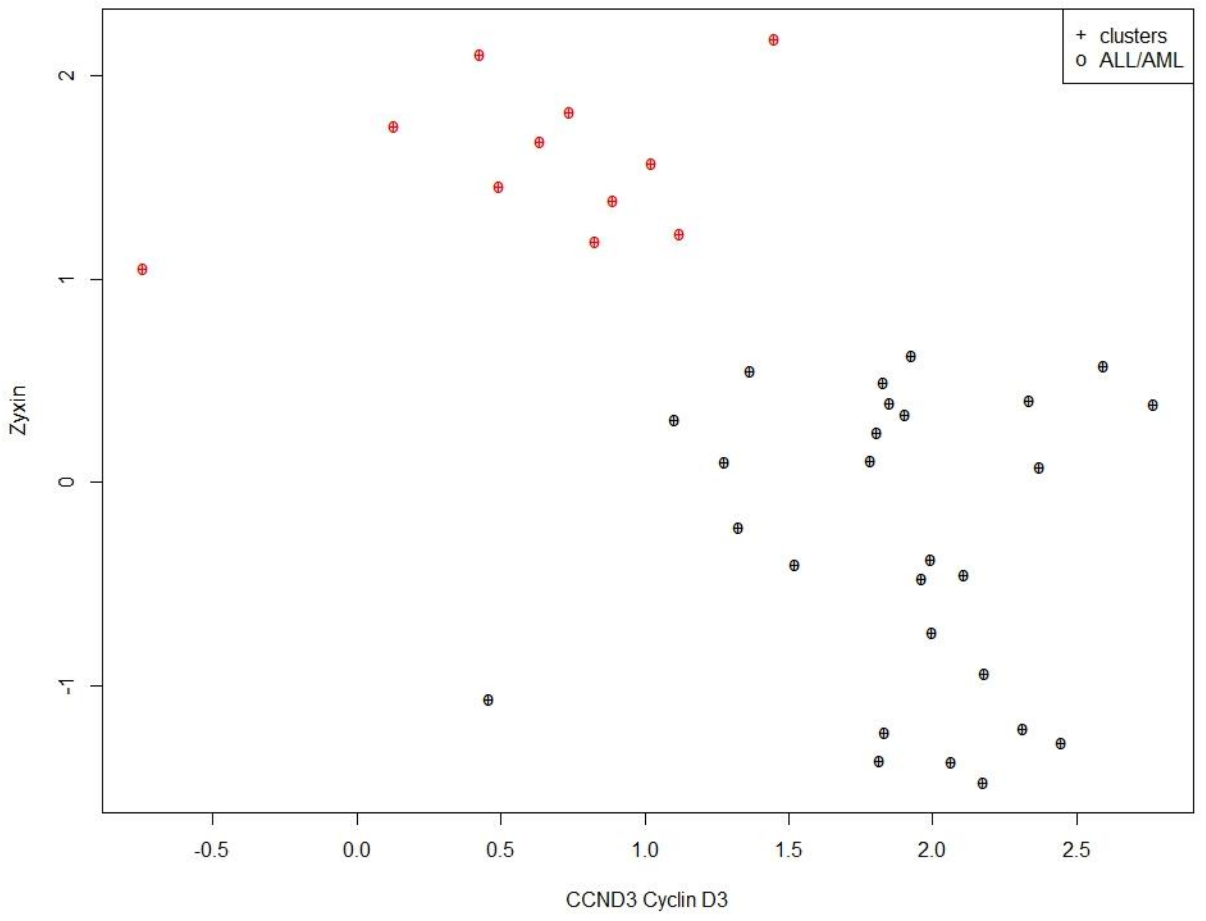
Since SSE is non-increasing in each iteration of steps (1) and (2), this guarantees the result as a local minimum. Generally several random starting partitions are tried to get closer to the global minimum.

In R, the algorithm is implemented in kmeans(). We illustrate its application on the ALL data set next.

Demonstration: K-means Clustering on Golub Data

Example; For the Golub data, we do a 2-means cluster analysis using the expression values of two genes "CCND3 Cyclin D3" and "Zyxin" as in the last lesson. The analysis tried 10 random starting partitions of two clusters.

```
> clusdata <- data.frame(golub[1042,], golub[2124,])

> colnames(clusdata)<- c("CCND3 Cyclin D3", "Zyxin")

> gol.fac <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))

> cl.2mean <- kmeans(clusdata, centers=2, nstart = 10)
```



We can see that the two clusters discriminate exactly the ALL patients (in black) from the AML patients (in red).

K-means Clustering on Golub Data (Cont’d)

Let us look at the output of the 2-means clustering analysis in more detail

```
> cl.2mean
K-means clustering with 2 clusters of sizes 11, 27

Cluster means:
  CCND3 Cyclin D3       Zyxin
1       0.6355909  1.5866682
2       1.8938826 -0.2947926

Clustering vector:
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1]  4.733248 19.842225
 (between_SS / total_SS =  62.0 %)

Available components:

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

We can see that the R output contains the two clusters’ means, the cluster label for each patient, within cluster sum of squares, etc. To study the stableness of the two predicted clusters’ means, we can use bootstrap analysis. We do the bootstrap on next page.

K-means Clustering on Golub Data (Cont'd)

As in the textbook, we do 1000 bootstrap resampling and fit the 2-means clustering using the previous two cluster means as initial partitions.

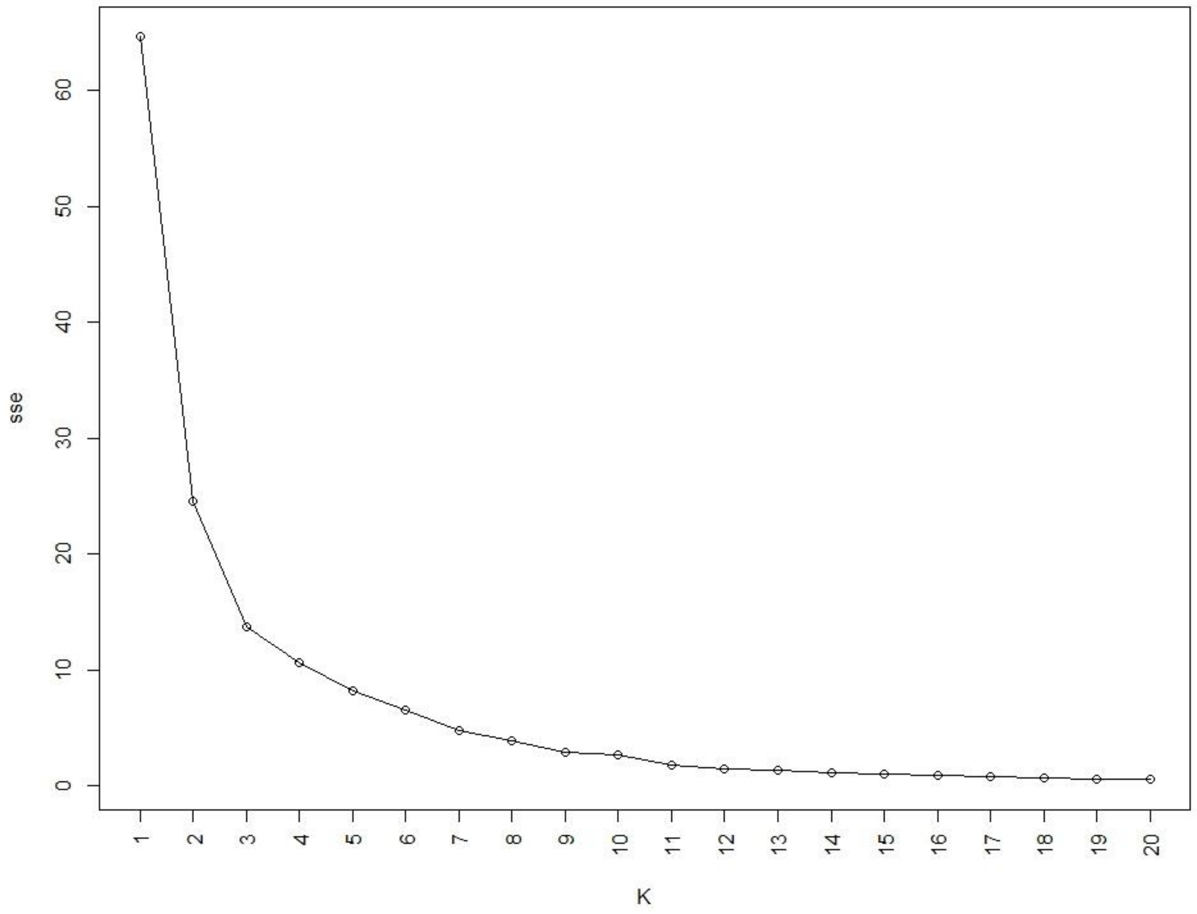
```
initial <-cl.2mean$centers
n <- dim(clusdata)[1]; nboot<-1000
boot.cl <- matrix(NA,nrow=nboot,ncol = 4)
for (i in 1:nboot){
    dat.star <- clusdata[sample(1:n,replace=TRUE),]
    cl <- kmeans(dat.star, initial, nstart = 10)
    boot.cl[i,] <- c(cl$centers[,1], cl$centers[,2])
}
> apply(boot.cl,2,mean)
[1]  0.6394810  1.8941089  1.5715604 -0.3098556
> quantile(boot.cl[,1],c(0.025,0.975))
     2.5%     97.5%
0.2671617 0.9756043
> quantile(boot.cl[,2],c(0.025,0.975))
    2.5%    97.5%
1.707684 2.077394
> quantile(boot.cl[,3],c(0.025,0.975))
    2.5%    97.5%
1.272734 1.809685
> quantile(boot.cl[,4],c(0.025,0.975))
       2.5%       97.5%
-0.61976929 -0.03681523
```

We can see that the two clusters remain stable over resampling. The bootstrap means are almost same as the 2-means from the original clustering. Hence, the estimation bias is small. The estimations of cluster means are quite precise because the 95% bootstrap confidence intervals are fairly small.

Deciding the Number of Clusters K on Golub Data

A crucial decision in the K-means clustering analysis is the choice of K, the number of clusters to use. A common method is to look at the SSE values for different choice of K. The SSE for the K-means clustering is also called the within clusters sum of squares. And we can see it is stored in the field “tot.withinss” in the R output earlier. Let us plot the SSE values versus K values.

```
K<-(1:20); sse<-rep(NA,length(K))
for (k in K) {
     sse[k]<-kmeans(clusdata, centers=k,nstart = 10)$tot.withinss
}
plot(K, sse, type='o', xaxt='n'); axis(1, at = K, las=2)
```

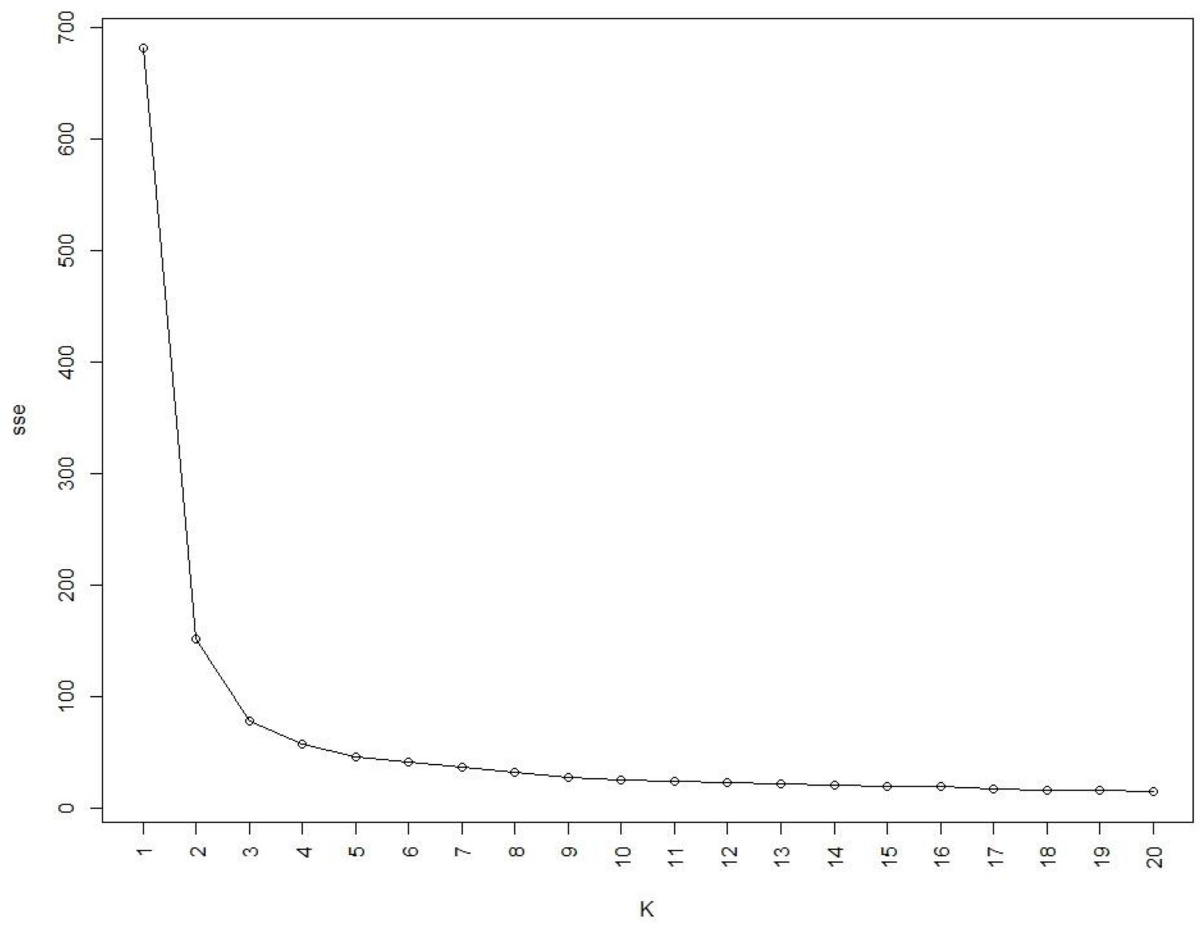


We can see a big drop off in SSE from K=1 to K=2. There is a further drop off of SSE to K = 3. Afterwards, the decrease in SSE starts to level off. So two or three clusters seems best in the data.

Deciding the Number of Clusters K on Iris Data

Let us revisit the iris data set used in the example in lesson 1 of this module. We again plot the SSE versus the number of clusters K.

```
iris2<- iris; iris2$Species <- NULL

K<-(1:20); sse<-rep(NA,length(K))
for (k in K) {
    sse[k]<-kmeans(iris2, centers=k,nstart = 10)$tot.withinss
}
plot(K, sse, type='o', xaxt='n'); axis(1, at = K, las=2)
```

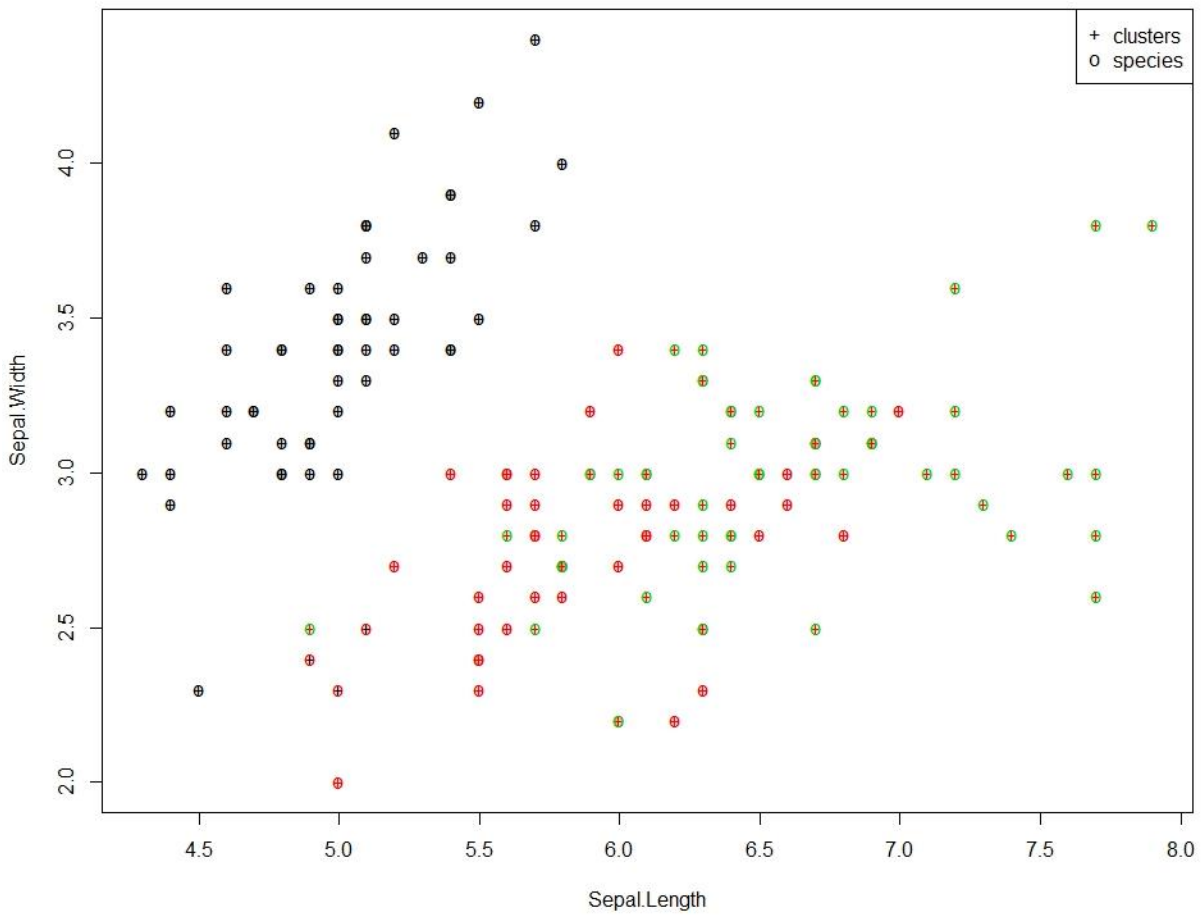


This again shows a big drop at K=2, and quickly levels off after K=3. We check the K-means clustering with K=2 next.

K-Means Clustering with K=2 on Iris Data

We reanalyze the iris data set with 2-means clustering.

```
clusters.2km <- kmeans(iris2, centers=2)
plot(iris2[c("Sepal.Length", "Sepal.Width")], col = clusters.2km$cluster,
pch="+")
points(iris[c("Sepal.Length", "Sepal.Width")], col = iris$Species, pch="o")
legend("topright",c("clusters","species"), pch= c("+","o"))
```



We can see that the two clusters (different colored “+”s) put all ‘setosa’ flowers plus three other flowers in one cluster, and the rest of the flowers in another.

The iris data and the Golub data example show that the SSE curve gives us some indication of natural clusters number in the data set. But it is really hard to decide the exact value of K.

K-Medoid Clustering

In univariate statistics the sample median, or the middle observation, is often considered as a robust alternative to the sample mean. The median is insensitive, or robust, to outliers, no matter how extreme an outlier. The sample mean minimizes the sum of the squared distances (i.e. squared Euclidean distance) to the data points, while the median minimizes the sum of the absolute differences (L1 metric) to the data points.

In p-dimensional data, the sample mean is also called the centroid, which we used in the K-Means clustering. The centroid is also sensitive to outliers. The p-dimensional sample median is the so-called medoids: the data point for which the sum of the distance with all other data points in the cluster is at a minimum. Correspondingly, we can use K-Medoid clustering which is a more robust version of K-Means clustering.

The K-Medoid is implemented by the pam() function in the “cluster” package. The command pam stands for “partition around medoids”.

We can do the K-Medoid clustering on the iris data, and plot the fit as before.

```
library(cluster)
cl.pam<-pam(iris2, k=3)
plot(iris2[c("Sepal.Length", "Sepal.Width")], col = cl.pam$cluster, pch="+")
points(iris[c("Sepal.Length", "Sepal.Width")], col = iris$Species, pch="o")
legend("topright",c("clusters","species"), pch= c("+","o"))
```

The resulting clusters are very similar to the K-Means clustering on this data set. That is because the ‘setosa’ flowers are well separated from the other two species which mix a little with each other. There are not many outliers to show the difference between the K-Means and K-Medoids approach on this data set.

Different Distance Measures in K-Medoid Clustering

As we mentioned in the first lesson, Euclidean distance is not the only distance measure for clustering. The pam() function has an option of specifying metric= ‘euclidean’ or ‘manhattan’. More generally, we can provide the distance matrix (pairwise distance of all data points) instead of the data set to pam() function. In the following we fit the iris data with Euclidean, Manhattan and Max component distances respectively, and compare the resulting clusters with the species.

```
> table(iris$Species, pam(dist(iris2, method='eucl'), k=3)$cluster)
              1    2    3
  setosa     50    0    0
  versicolor  0   48    2
  virginica   0   14   36
> table(iris$Species, pam(dist(iris2, method='manh'), k=3)$cluster)
              1    2    3
  setosa     50    0    0
  versicolor  0   13   37
  virginica   0   48    2
> table(iris$Species, pam(dist(iris2, method='max'), k=3)$cluster)
              1    2    3
  setosa     50    0    0
  versicolor  0   14   36
  virginica   0   48    2
```

We see that there are 16 cases of misclassification for Euclidean and Max component distances, while Manhattan distance results in 15 misclassification. We can also use some nonstandard distance (dissimilarity) measures. One such dissimilarity measure is one minus the correlation coefficient $d(\vec{x},\vec{y}) = 1 - cor(\vec{x},\vec{y})$.

```
> table(iris$Species, pam(as.dist(1-cor(t(iris2))), k=3)$cluster)
              1    2    3
  setosa     50    0    0
  versicolor  0   47    3
  virginica   0    3   47
```

For the iris data, the correlation dissimilarity works better than the other distances, resulting in only 6 misclassifications.

The correlation dissimilarity measures the closeness between the patterns of change among the features of the two data points, rather than the closeness in the absolute values of each feature of the two data points. The correlation dissimilarity can be particularly useful when we cluster the genes, since the genes has similar patterns of varying expression across patients may have similar functions, even if their absolute expression levels differ.

Syntax for Distance Measures in K-means, K-Medoid and Hierarchical Clustering

We have seen that different distance measures can be used in the K-medoid and hierarchical clustering. A nature question is, what if we use different distance measures in the K-means clustering? However, K-means clustering can only be done under the Euclidean distance. The reason is that the "mean" is calculated from the Euclidean coordinates system. The "mean" can be a point other than the n data points in the data set. We will not be able to calculate the mean under a different distance measure, with only pairwise distances among the n data points. The K-medoid always take the center as one of the n data points, and the pairwise distances will allow calculation of the medoid (even without the coordinates of the original n data points).

In summary, the K-means only work on a data matrix with Euclidean coordinates for the n data points. So the syntax is kmeans(x, centers) where x is a numeric data matrix whose rows are the data points to cluster.

The hierarchical clustering hclust() only takes the distance structure (containing pairwise distances) as an input. That is, hclust(d), where d can be calculated from the data matrix x by dist(x).

The K-medoid pam() can take either the data matrix x or the distance structure d as input. That is, we can use either pam(x, k) or pam(d, k). For the data matrix x, pam(x, k) gives the same clustering results as pam(dist(x), k) which use the default Euclidean distance.

Comments on Evaluation of Clustering Methods

As introduced at the beginning of this module, clustering analysis is a type of unsupervised learning. This makes the evaluation of the clustering analysis harder than standard statistical analysis such as ANOVA or regression. *The results of the clustering analysis needs to be interpreted with caution.*

We used two example data sets with well separated categories to illustrate the clustering methods. Strictly speaking, these two data sets belong to the classification problem since we know the categories which are not available for real clustering analysis application. Also, in many applications the categories are not as well separated. (It is more typical for two categories to be mingled somewhat in practice, as the two species 'virginica' and 'versicolor' in the iris data set.) Generally, the clustering analysis need careful considering of three questions:

1. What distance (dissimilarity) measures to use?
2. What type of linkage to use?
3. How many clusters should be used?

The choice of answers to these three questions affect the results of clustering analysis. We have seen some effects of these choices in the examples. There is no best answer on the choices, but we did discuss some ideas on the choices.

As an unsupervised learning question, the clustering analysis generally do not have *THE correct answer*. It is used in practice for exploration. That is, to generate interesting hypothesis for further scientific study. Hence, we should follow the general ideas of doing reasonable clustering, and can try several approaches together. The results should not be taken as the absolute truth about the data: K=3 or K=5 clusters may both be correct for a data set, reflecting different levels of interesting groupings.

In the microarray example of ALL data set, we have illustrated the clustering of the patients (column vectors). This is for illustration purpose as we have two clearly defined patient groups with relative few sample size. In practice, clustering the genes (row vectors) are often done to generate interesting hypothesis for further study. The same R procedures can be easily applied for clustering in the rows.

Summary

In this lesson we were introduced to two partitional clustering methods: K-means and K-Medoids. You should also know how to do K-Medoids with other distance measures. You should also know to plot SSE for choosing the number of clusters. Finally, we discussed the three important questions when doing clustering analysis, and why we need be cautious in interpreting clustering results.

Module Summary

This module covered clustering analysis. You should understand the purpose of clustering analysis, and two approaches generally used. You should be able to conduct the hierarchical clustering in R with various linkage functions and various distance measures. You should be able to do K-Means clustering in R, and to do K-Medoids clustering in R with various distance measures. You should know how to choose the number of clusters.

In the next module, we will study multivariate distributions. And we will learn another unsupervised learning method, the principal components analysis.