STATS 415 Lab12

Weijing Tang

April 5, 2018

1 Today's objectives

1. Learn how to implement Hierarchical Clustering.
2. Learn how to implement k-means.
3. Clustering the observations of NCI60 data.

2 Hierarchical Clustering

The `hclust()` function implements hierarchical clustering in R. We begin with a simple simulated example in which there truly are two clusters in the data: the first 25 observations have a mean shift relative to the next 25 observations.

```
set.seed(2)
x=matrix(rnorm(50*2), ncol=2)
x[1:25,1]=x[1:25,1]+3
x[1:25,2]=x[1:25,2]-4
```

In this example we plot the hierarchical clustering dendrogram using complete, single, and average linkage clustering, with Euclidean distance as the dissimilarity measure. We begin by clustering observations using complete linkage. The `dist()` function is used to compute the 50×50 inter-observation Euclidean distance matrix.

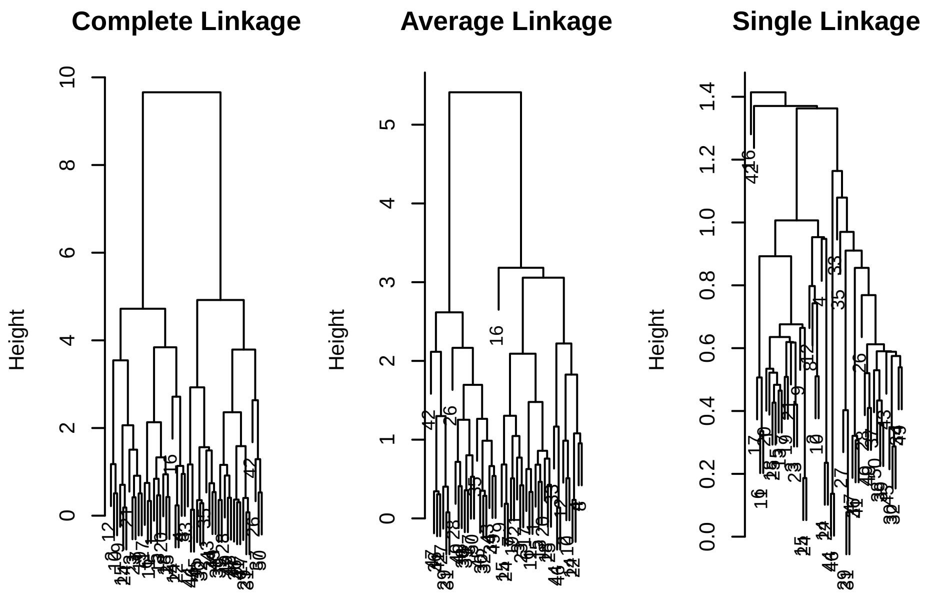
```
hc.complete=hclust(dist(x), method="complete")
```

We could just as easily perform hierarchical clustering with average or single linkage instead:

```
hc.average=hclust(dist(x), method="average")
hc.single=hclust(dist(x), method="single")
```

We can now plot the dendrograms obtained using the usual plot() function. The numbers at the bottom of the plot identify each observation.

```
par(mfrow=c(1,3))
plot(hc.complete,main="Complete Linkage", xlab="", sub="", cex=.9)
plot(hc.average, main="Average Linkage", xlab="", sub="", cex=.9)
plot(hc.single, main="Single Linkage", xlab="", sub="", cex=.9)
```



To determine the cluster labels for each observation associated with a given cut of the dendrogram, we can use the `cutree()` function:

```
cutree(hc.complete, 2)
```

```
##  [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 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
cutree(hc.average, 2)
```

```
##  [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 2 2 2 2 2 2 2 1 2 2
## [36] 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
```

```
cutree(hc.single, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Question: For this data we know the true clusters, how about the performance of three cluster dissimilarity measures?

For this data, complete and average linkage generally separate the observations into their correct groups. However, single linkage identifies one point as belonging to its own cluster. A more sensible answer is obtained when four clusters are selected, although there are still two singletons.

```r
cutree(hc.single, 4)
```

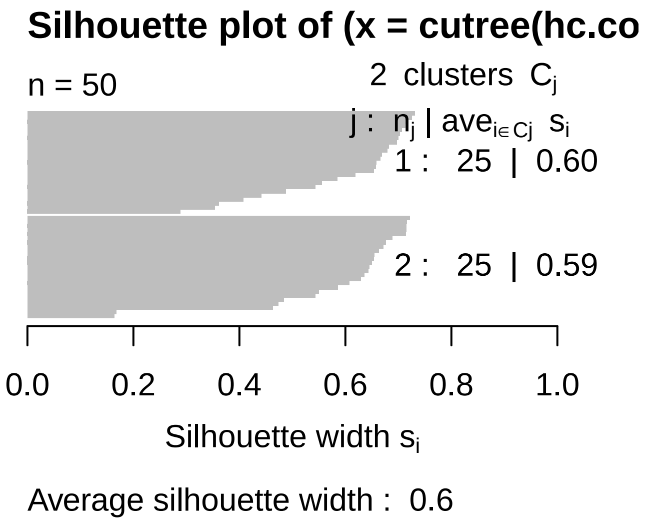
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3
## [36] 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3
```

To make silhouette coefficient plot we use `silhoutte()` function in the package `cluster`.

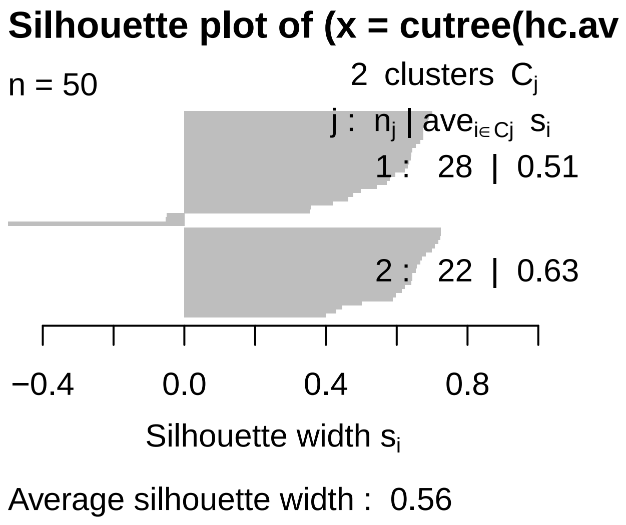
```r
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 3.3.3
```

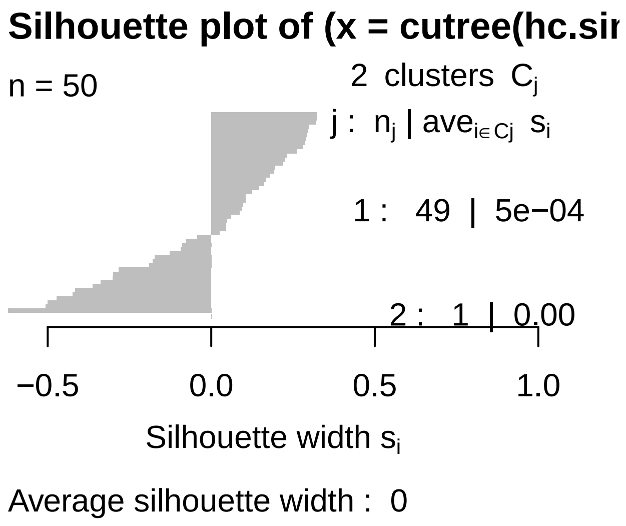
```r
sil.complete = silhouette(cutree(hc.complete,2),dist = dist(x))
sil.average = silhouette(cutree(hc.average,2),dist = dist(x))
sil.single = silhouette(cutree(hc.single,2),dist = dist(x))
par(mfrow=c(1,1))
plot(sil.complete)
```



```r
plot(sil.average)
```



```r
plot(sil.single)
```

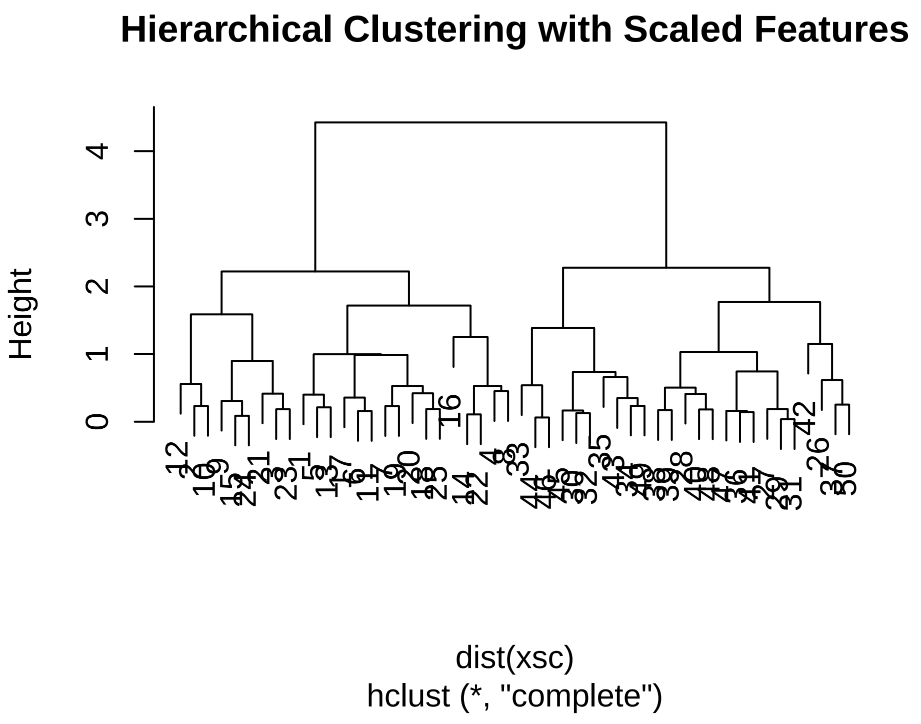


Question: What are the average silhouette coefficients of three measures? Which one is better?

Recall that the silhouette coefficient is always between -1 and 1. The closer to 1, the better the clustering result is.

To scale the variables before performing hierarchical clustering of the observations, we use the `scale()` function:

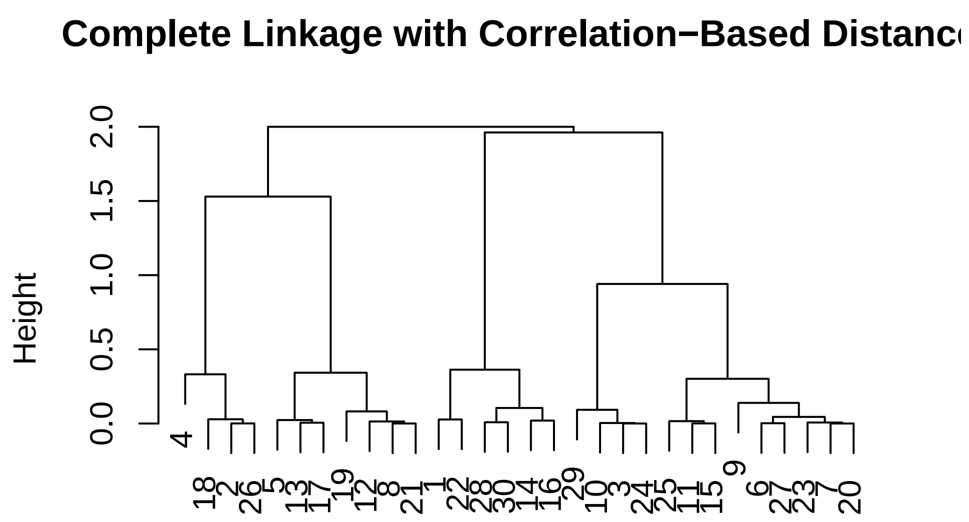
```r
xsc=scale(x)
plot(hclust(dist(xsc), method="complete"), main="Hierarchical Clustering with Scaled Features")
```



Note that standardization is optimal and do it if there is a special reason.

We can also replace the Euclidean distance by correlation. Correlation-based distance can be computed using the `cor` function and `as.dist()` function, which converts an arbitrary square symmetric matrix into a form that the `hclust()` function recognizes as a distance matrix. However, this only makes sense for data with at least three features since the absolute correlation between any two observations with measurements on two features is always 1. Hence, we will cluster a three-dimensional data set.

```
set.seed(2)
x=matrix(rnorm(30*3), ncol=3)
dd=as.dist(1-cor(t(x)))
plot(hclust(dd, method="complete"),
     main="Complete Linkage with Correlation-Based Distance", xlab="", sub="")
```



3 K-Means Clustering

The function `kmeans()` performs K-means clustering in R. Here we still use the 2-dimension data for better visualization.

```
set.seed(2)
x=matrix(rnorm(50*2), ncol=2)
x[1:25,1]=x[1:25,1]+3
x[1:25,2]=x[1:25,2]-4
```

We now perform K-means clustering with K = 2.

```
set.seed(2)
km.out=kmeans(x,2,nstart=20)
km.out
```

```
## K-means clustering with 2 clusters of sizes 25, 25
##
## Cluster means:
##         [,1]       [,2]
## 1  3.3339737 -4.0761910
## 2 -0.1956978 -0.1848774
##
## 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 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 63.20595 65.40068
##  (between_SS / total_SS =  72.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

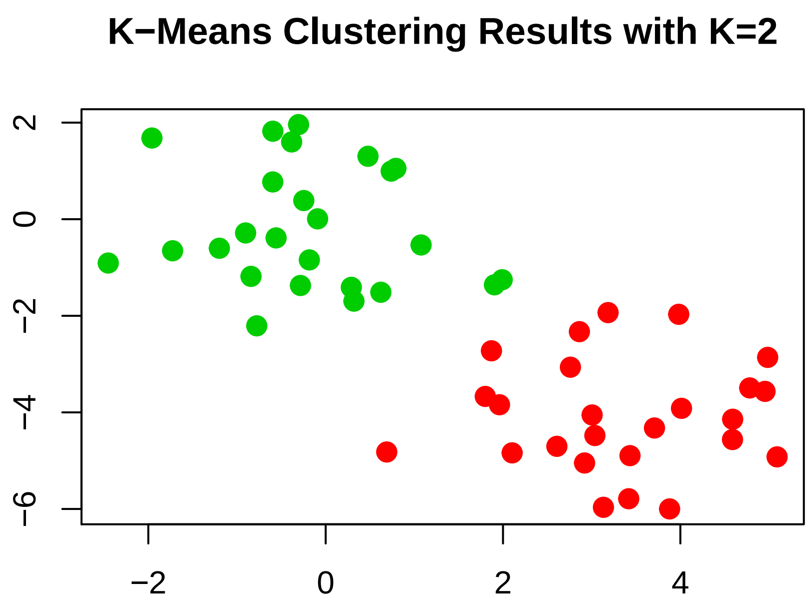
The cluster assignments of the 50 observations are contained in km.out$cluster.

```
km.out$cluster
```

```
##  [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 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

The K-means clustering perfectly separated the observations into two clusters even though we did not supply any group information to `kmeans()`. We can plot the data, with each observation colored according to its cluster assignment.

```
plot(x, col=(km.out$cluster+1), main="K-Means Clustering Results with K=2",
     xlab="", ylab="", pch=20, cex=2)
```



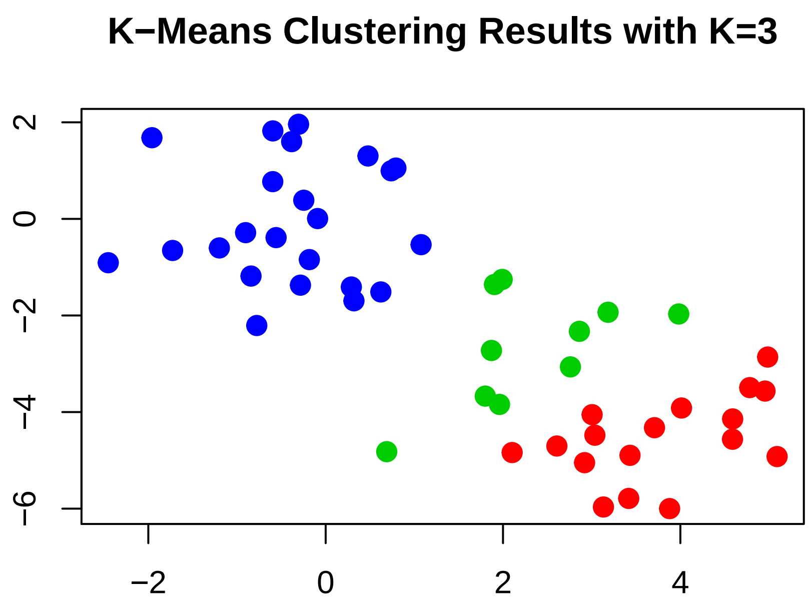
Note: Here the observations can be easily plotted because they are two-dimensional. If there were more than two variables then we could instead perform PCA and plot the first two principal components score vectors.

In this example, we know that there really were two clusters because we generated the data. However, for real data, in general we do not know the true number of clusters. We could instead have performed K-means clustering on this example with K = 3.

```
set.seed(2)
km.out=kmeans(x,3,nstart=20)
km.out
```

```
## K-means clustering with 3 clusters of sizes 17, 10, 23
##
## Cluster means:
##         [,1]        [,2]
## 1  3.7789567 -4.56200798
## 2  2.3001545 -2.69622023
## 3 -0.3820397 -0.08740753
##
## Clustering vector:
##  [1] 1 2 1 2 1 1 1 2 1 2 1 2 1 2 1 2 1 1 1 1 1 2 1 1 1 3 3 3 3 3 3 3 3 3 3
## [36] 3 3 3 3 3 3 3 3 2 3 2 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1] 25.74089 19.56137 52.67700
##  (between_SS / total_SS =  79.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
plot(x, col=(km.out$cluster+1), main="K-Means Clustering Results with K=3",
     xlab="", ylab="", pch=20, cex=2)
```



When K = 3, K-means clustering splits up the two clusters.

3.1 Attention to Initialization of K-means

1. multiple initial cluster assignments.

To run the `kmeans()` function in R with multiple initial cluster assignments, we use the `nstart` argument. If a value of nstart greater than one is used, then K-means clustering will be performed using multiple random assignments for initial cluster assignments, and the `kmeans()` function will report only the best results. Here we compare using nstart=1 to nstart=20.

```
set.seed(3)
km.out=kmeans(x,3,nstart=1)
km.out$tot.withinss
```

```
## [1] 104.3319
```

```
km.out=kmeans(x,3,nstart=20)
km.out$tot.withinss
```

```
## [1] 97.97927
```

Note that km.out$tot.withinss is the total within-cluster sum of squares, which we seek to minimize by performing K-means clustering. The individual within-cluster sum-of-squares are contained in the vector km.out$withinss. By using `nstart=20` we get a lower within-cluster sum of squares. **We strongly recommend always running K-means clustering with a large value of nstart, such as 20 or 50, since otherwise an undesirable local optimum may be obtained.**

When performing K-means clustering, in addition to using multiple initial cluster assignments, it is also important to set a random seed using the `set.seed()` function. This way, the initial cluster assignments can be replicated, and the K-means output will be fully reproducible.

2. use the solution from some hierarchical algorithm as initial value

```
cutree(hc.complete, 2)
```

```
##  [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 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
hcmean1 = colMeans(x[which(cutree(hc.complete, 2)==1),])
hcmean2 = colMeans(x[which(cutree(hc.complete, 2)==2),])
hcmean= rbind(hcmean1,hcmean2)
```

We calculate the centres of each cluster based on the solution of hierarchical clustering. If we want to specify the initial centres, we can use argument `centers`. It requires to be a $k \times p$ matrix, each row represents the initial mean of each cluster.

```
km.hc = kmeans(x,centers = hcmean)
km.hc$iter
```

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

The algorithm stops after only one iteration.

4 Clustering the Observations of the NCI60 Data

Up to now, we onlu use simulated data, let's see how clustering methods work on real data.

4.1 The NCI60 data

Unsupervised techniques are often used in the analysis of genomic data. In particular, PCA and hierarchical clustering are popular tools. We illustrate clustering techniques on the NCI60 cancer cell line microarray data, which consists of 6,830 gene expression measurements on 64 cancer cell lines.

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.3.3
```

```
nci.labs=NCI60$labs
nci.data=NCI60$data
```

Each cell line is labeled with a cancer type. We do not make use of the cancer types in performing clustering. But after performing clustering, we will check to see the extent to which these cancer types agree with the results of these unsupervised techniques.

The data has 64 rows and 6,830 columns.

```
dim(nci.data)
```

```
## [1]   64 6830
```

We begin by examining the cancer types for the cell lines.

```
unique(nci.labs)
```

```
##  [1] "CNS"          "RENAL"        "BREAST"       "NSCLC"        "UNKNOWN"
##  [6] "OVARIAN"      "MELANOMA"     "PROSTATE"     "LEUKEMIA"     "K562B-repro"
## [11] "K562A-repro" "COLON"        "MCF7A-repro" "MCF7D-repro"
```

```
table(nci.labs)
```

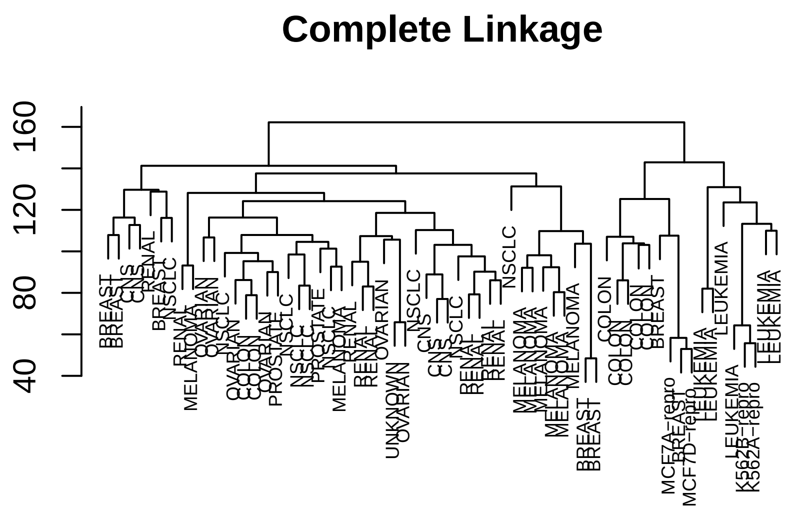
```
## nci.labs
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

To begin, we standardize the variables to have mean zero and standard deviation one. As mentioned earlier, this step is optional and should be performed only if we want each gene to be on the same scale.

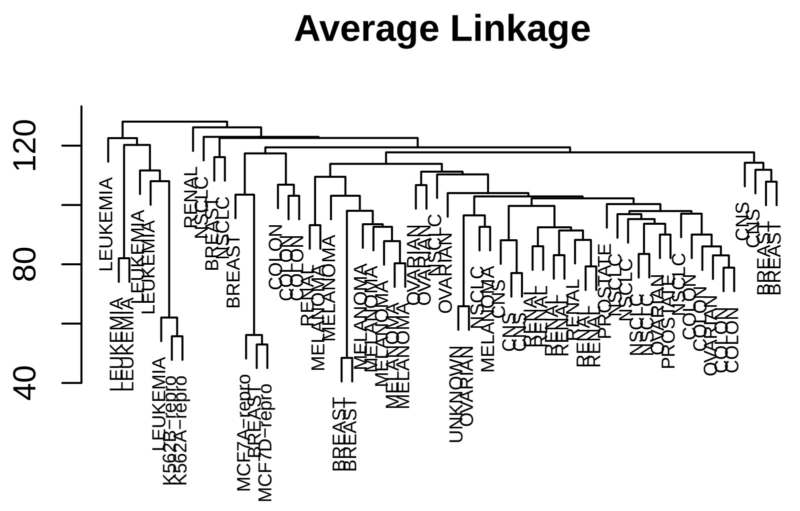
```
sd.data=scale(nci.data)
```

We now proceed to hierarchically cluster the cell lines in the NCI60 data, with the goal of finding out whether or not the observations cluster into distinct types of cancer. We now perform hierarchical clustering of the observations using complete, single, and average linkage. Euclidean distance is used as the dissimilarity measure.

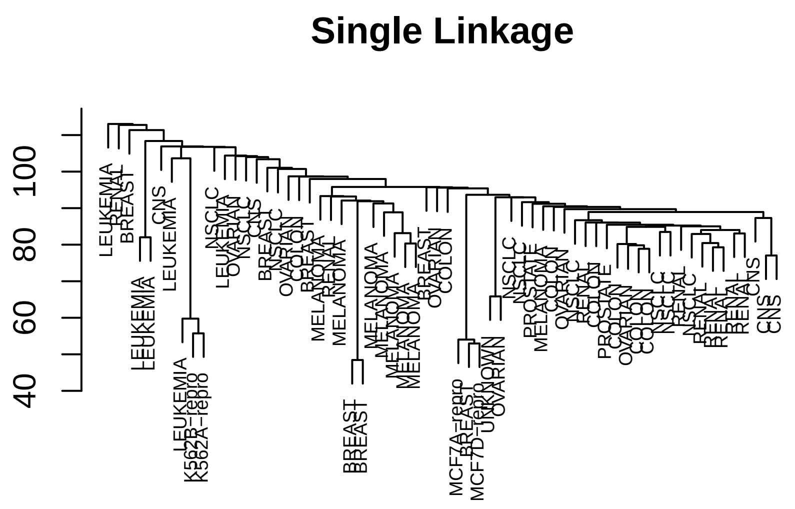
```
par(mfrow=c(1,1))
data.dist=dist(sd.data)
plot(hclust(data.dist), labels=nci.labs, main="Complete Linkage",
     xlab="", sub="",ylab="",cex = 0.6)
```



```r
plot(hclust(data.dist, method="average"), labels=nci.labs,
     main="Average Linkage", xlab="", sub="",ylab="",cex = 0.6)
```



```r
plot(hclust(data.dist, method="single"), labels=nci.labs,
     main="Single Linkage", xlab="", sub="",ylab="",cex = 0.6)
```



We see that the choice of linkage certainly does affect the results obtained. Typically, single linkage will tend to yield trailing clusters: very large clusters onto which individual observations attach one-by-one. On the other hand, complete and average linkage tend to yield more balanced, attractive clusters. For this reason, complete and average linkage are generally preferred to single linkage. Clearly cell lines within a single cancer type do tend to cluster together, although the clustering is not perfect. We will use complete linkage hierarchical clustering for the analysis that follows.

We can cut the dendrogram at the height that will yield a particular number of clusters, say four:

```r
hc.out=hclust(data.dist)
hc.clusters=cutree(hc.out,4)
table(hc.clusters,nci.labs)
```

```
##             nci.labs
## hc.clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
##           1      2   3     2           0           0        0           0
##           2      3   2     0           0           0        0           0
##           3      0   0     0           1           1        6           0
##           4      2   0     5           0           0        0           1
##             nci.labs
## hc.clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
##           1           0        8     8       6        2     8       1
##           2           0        0     1       0        0     1       0
##           3           0        0     0       0        0     0       0
##           4           1        0     0       0        0     0       0
```

Question: Any interesting patterns you find here?

For example, all the leukemia cell lines fall in cluster 3, while the breast cancer cell lines are spread out over three different clusters.

Let's try K-means!

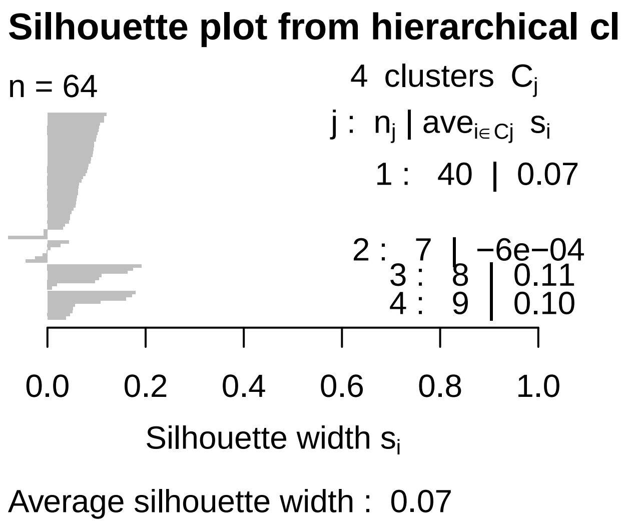
```
set.seed(2)
km.out=kmeans(sd.data, 4, nstart=20)
km.clusters=km.out$cluster
table(km.clusters,hc.clusters)
```

```
##            hc.clusters
## km.clusters  1  2  3  4
##           1 11  0  0  9
##           2  0  0  8  0
##           3  9  0  0  0
##           4 20  7  0  0
```

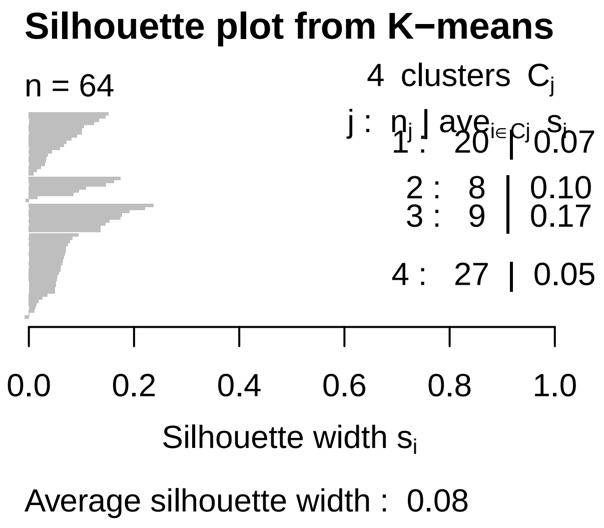
Question: How do these NCI60 hierarchical clustering results compare to what we get if we perform K-means clustering with K = 4?

K-means clustering and hierarchical clustering with the dendrogram cut to obtain the same number of clusters can yield very different results. We see that the four clusters obtained using hierarchical clustering and K-means clustering are somewhat different. Cluster 2 in K-means clustering is identical to cluster 3 in hierarchical clustering. However, the other clusters differ: for instance, cluster 4 in K-means clustering contains a portion of the observations assigned to cluster 1 by hierarchical clustering, as well as all of the observations assigned to cluster 2 by hierarchical clustering.

```
plot(silhouette(hc.clusters,data.dist),main="Silhouette plot from hierarchical clustering")
```



```
plot(silhouette(km.clusters,data.dist),main="Silhouette plot from K-means")
```



K-means has a slightly higher average silhouette width.