K-means and Hierarchical Clustering

MSBA7002: Business Statistics

Contents

1. Synthetic Data

1.1 K-Means Clustering

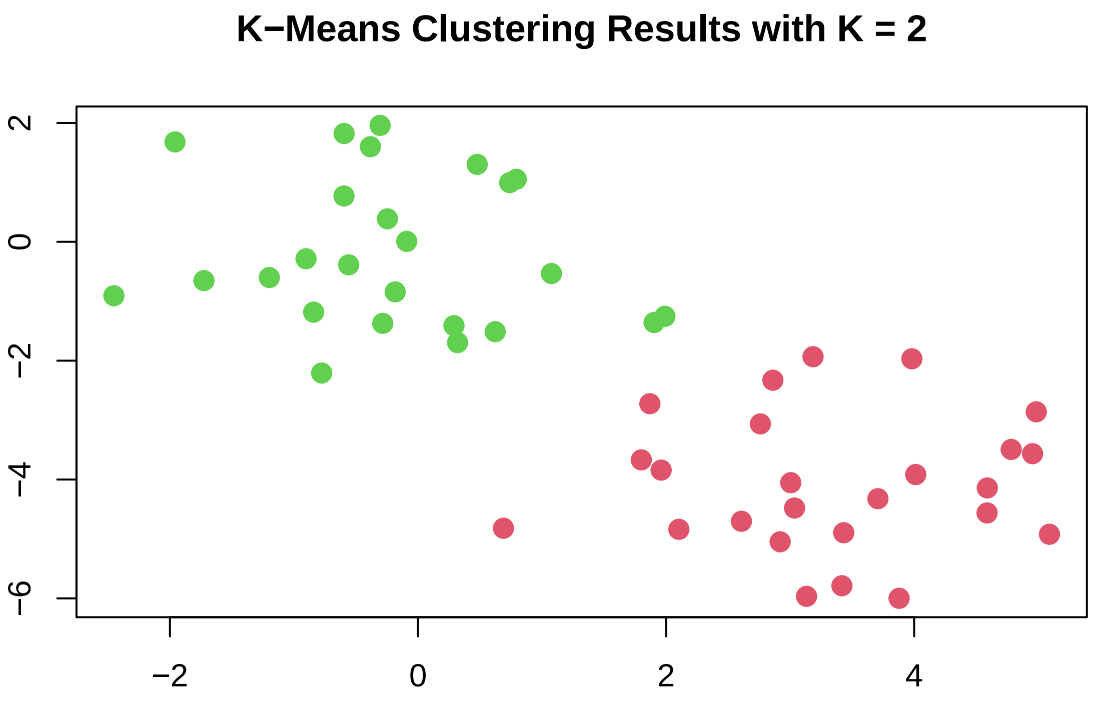
The function kmeans() performs K-means clustering in R. We first generate some simulated data and perform K-means clustering with K = 2.

```
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

km.out <- kmeans(x, 2, nstart = 20)
km.out$cluster  # The cluster assignments are contained in km.out$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 2 2
## [39] 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)
```



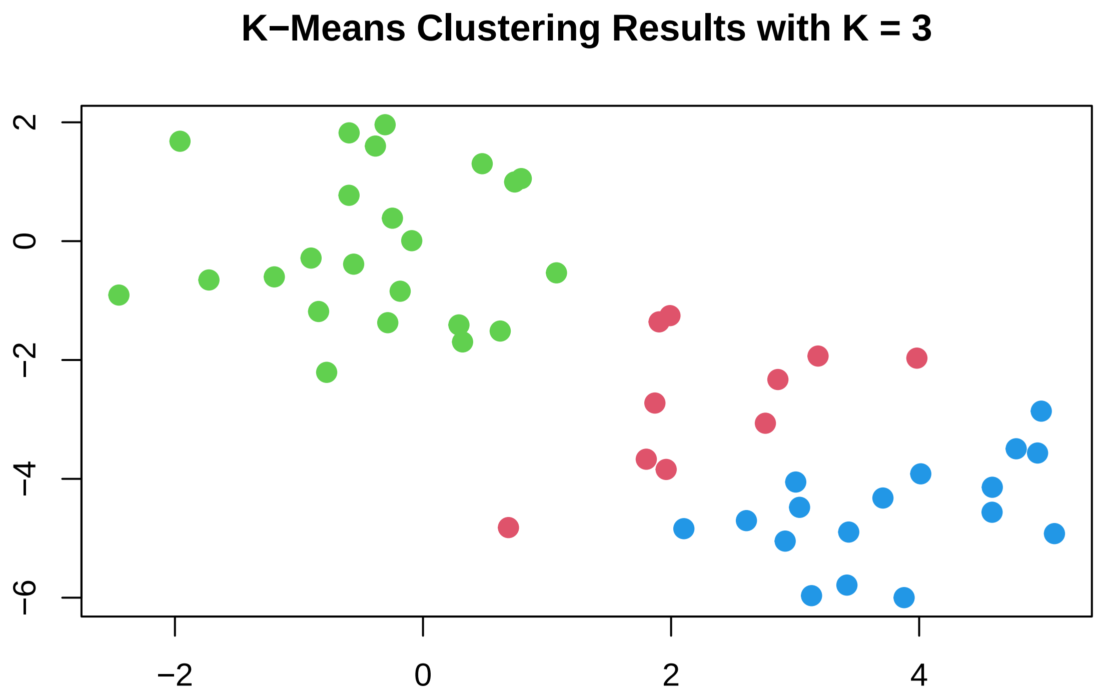
In this example, we knew that there really were two clusters because we generated the data. However, for real data, in general we do not know the true number of clusters. We could instead have performed K-means clustering on this example with K = 3.

```
set.seed(4)
km.out <- kmeans(x, 3, nstart = 20)
km.out
```

```
## K-means clustering with 3 clusters of sizes 10, 23, 17
##
## Cluster means:
```

```
##          [,1]        [,2]
## 1  2.3001545 -2.69622023
## 2 -0.3820397 -0.08740753
## 3  3.7789567 -4.56200798
##
## Clustering vector:
##  [1] 3 1 3 1 3 3 3 1 3 1 3 1 3 1 3 1 3 3 3 3 3 1 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 1 2 1 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 19.56137 52.67700 25.74089
##  (between_SS / total_SS =  79.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

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



To run the kmeans() function in R with multiple initial cluster assign- ments, 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, and the kmeans() function will report only the best results. Here we compare using nstart = 1 to nstart = 20.

```
set.seed(4)
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.

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.

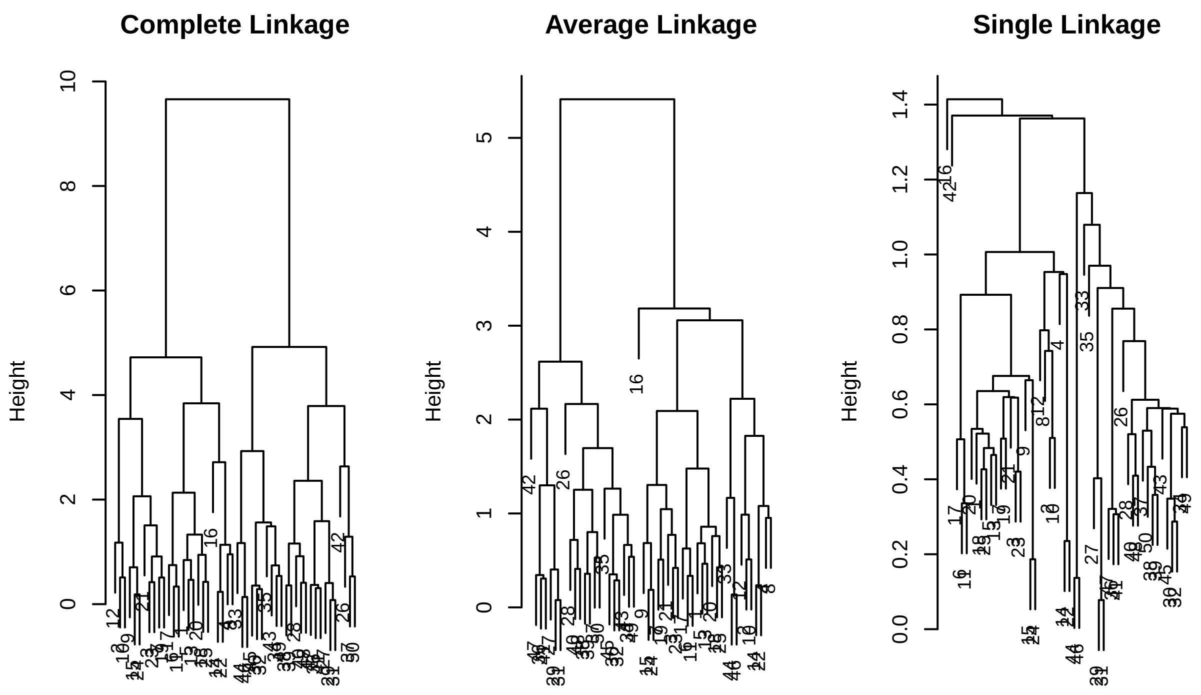
1.2 Hierarchical Clustering

The hclust() function implements hierarchical clustering in R. In the following, we generate hierarchical clustering dendrogram using complete, single, and average linkage clustering, with Euclidean distance as the dissimilarity measure. The dist() function is used to compute the inter-observation Euclidean distance matrix.

```
hc.complete <- hclust(dist(x), method = "complete")
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 2 2
## [39] 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 2 2
## [39] 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 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1
```

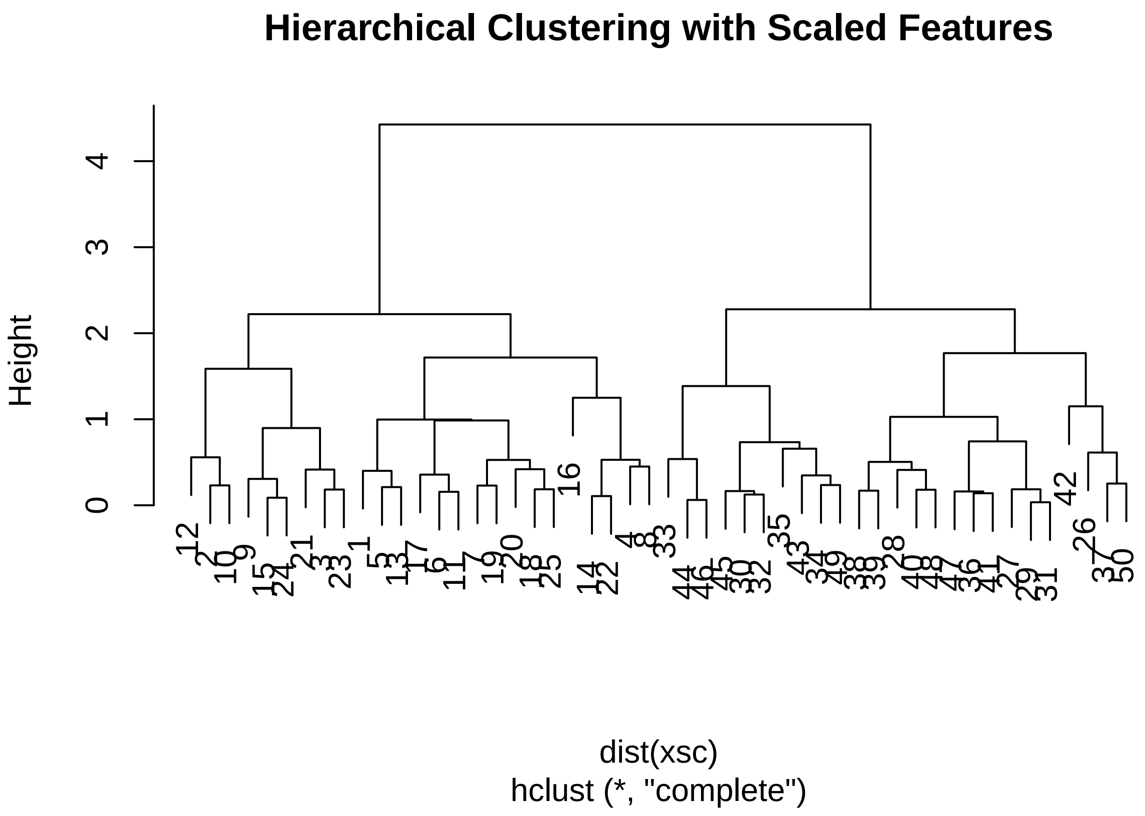
The second argument to cutree() is the number of clusters we wish to obtain. For this data, complete and average linkage generally separate the observations into their correct groups. However, single linkage identifies one point as belonging to its own cluster. A more sensible answer is obtained when 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 3 3 3
## [39] 3 3 3 4 3 3 3 3 3 3 3 3
```

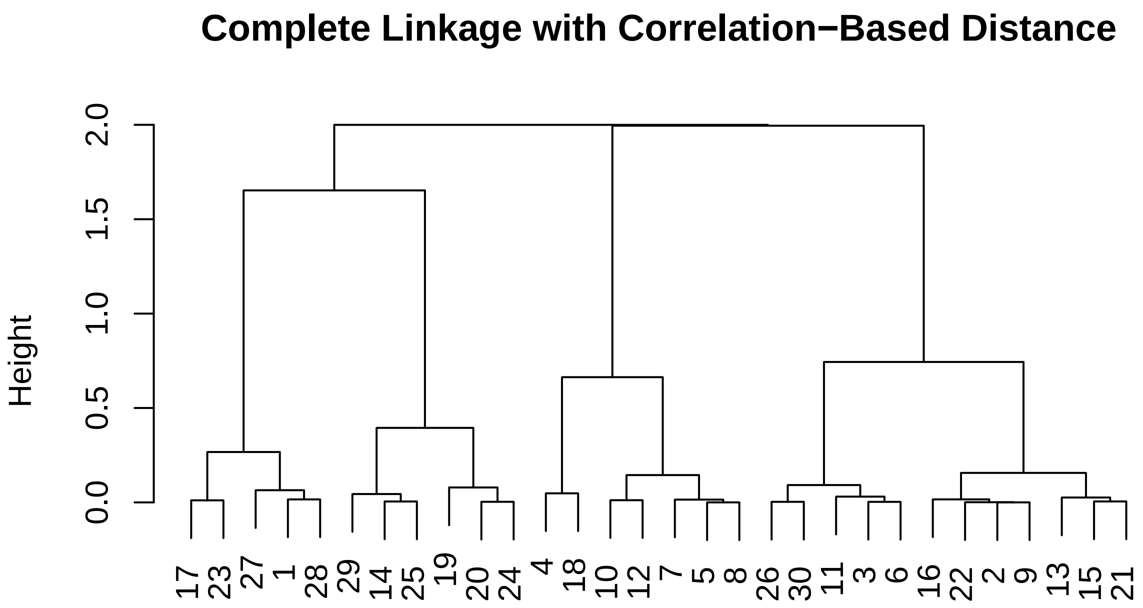
To scale the variables before performing hierarchical clustering of the observations, we use the scale() function.

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



Correlation-based distance can be computed using the 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. This data set does not contain any true clusters.

```r
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 = "")
```


Complete Linkage with Correlation−Based Distance
Height
0.0
0.5
1.0
1.5
2.0
17
23
27
1
28
29
14
25
19
20
24
4
18
10
12
7
5
8
26
30
11
3
6
16
22
2
9
13
15
21

2. NCI60 Data

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

```
library(ISLR2)
nci.labs <- NCI60$labs
nci.data <- NCI60$data
```

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

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

```
nci.labs[1:4]
```

```
## [1] "CNS"   "CNS"   "CNS"   "RENAL"
```

```
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
```

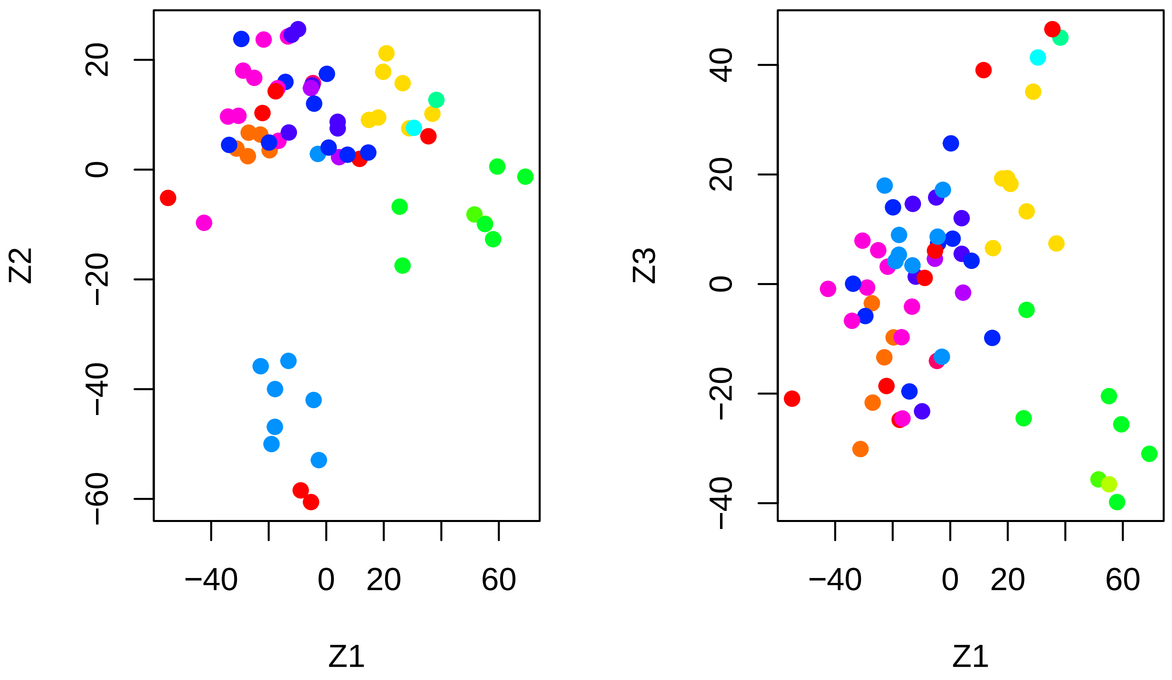
2.1 PCA

We first perform PCA on the data after scaling the variables (genes) to have standard deviation one, although one could reasonably argue that it is better not to scale the genes.

```
pr.out <- prcomp(nci.data, scale = TRUE)
```

We now plot the first few PC score vectors, in order to visualize the data. The observations (cell lines) corresponding to a given cancer type will be plotted in the same color, so that we can see to what extent the observations within a cancer type are similar to each other.

```
Cols <- function(vec) {
  cols <- rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])
}

par(mfrow = c(1, 2))
plot(pr.out$x[, 1:2], col = Cols(nci.labs), pch = 19, xlab = "Z1", ylab = "Z2")
plot(pr.out$x[, c(1, 3)], col = Cols(nci.labs), pch = 19, xlab = "Z1", ylab = "Z3")
```



On the whole, cell lines corresponding to a single cancer type do tend to have similar values on the first few principal component score vectors. This indicates that cell lines from the same cancer type tend to have pretty similar gene expression levels.

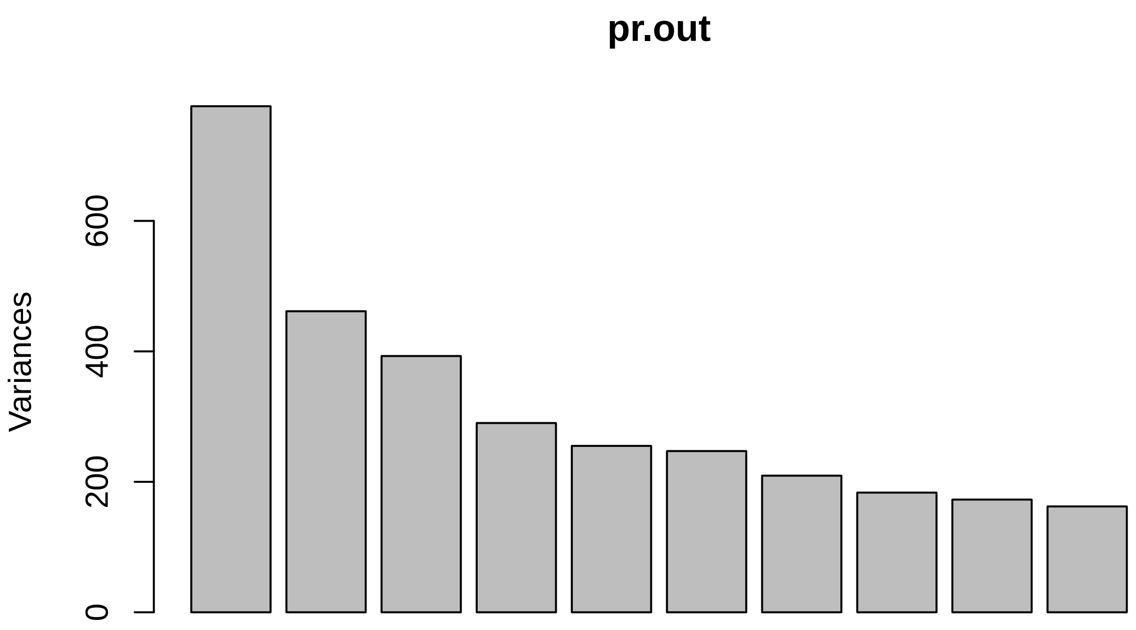
2.2 Proportion of Variance Explained

We can obtain a summary of the proportion of variance explained (PVE) of the first few principal components using the summary() method for a prcomp object.

```
summary(pr.out)
```

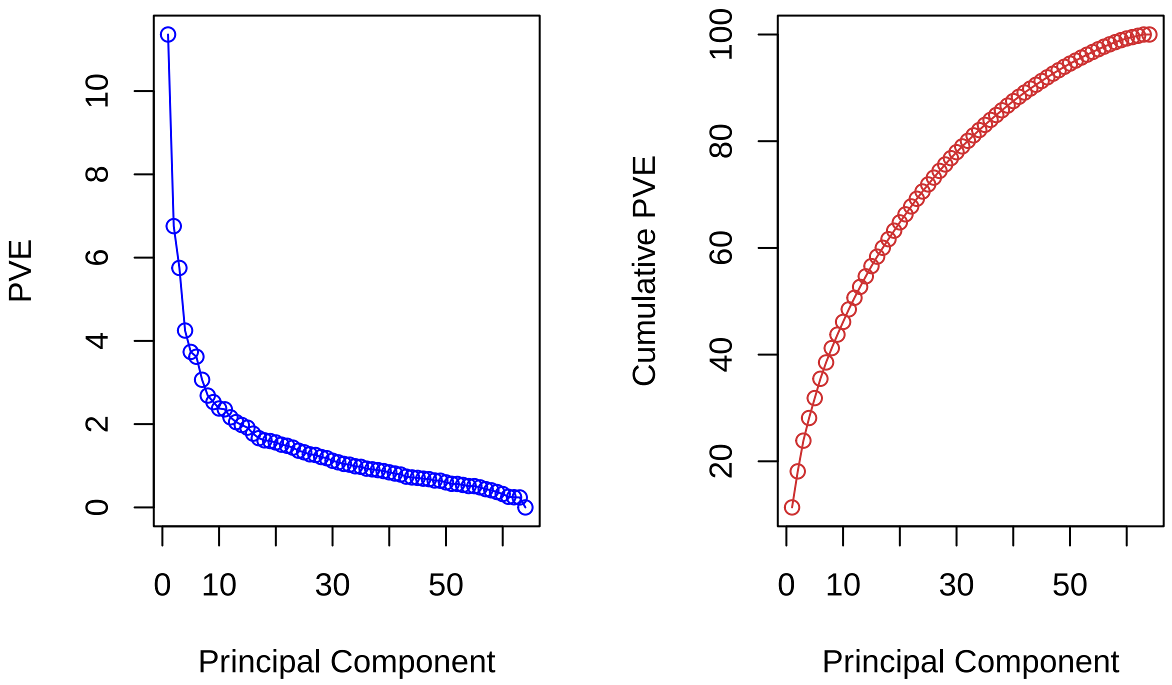
```
## Importance of components:
##                            PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     27.8535 21.48136 19.82046 17.03256 15.97181 15.72108
## Proportion of Variance  0.1136  0.06756  0.05752  0.04248  0.03735  0.03619
## Cumulative Proportion   0.1136  0.18115  0.23867  0.28115  0.31850  0.35468
##                             PC7      PC8      PC9     PC10     PC11     PC12
## Standard deviation     14.47145 13.54427 13.14400 12.73860 12.68672 12.15769
## Proportion of Variance  0.03066  0.02686  0.02529  0.02376  0.02357  0.02164
## Cumulative Proportion   0.38534  0.41220  0.43750  0.46126  0.48482  0.50646
##                            PC13     PC14     PC15     PC16     PC17     PC18
## Standard deviation     11.83019 11.62554 11.43779 11.00051 10.65666 10.48880
## Proportion of Variance  0.02049  0.01979  0.01915  0.01772  0.01663  0.01611
## Cumulative Proportion   0.52695  0.54674  0.56590  0.58361  0.60024  0.61635
##                            PC19    PC20     PC21    PC22    PC23    PC24
## Standard deviation     10.43518 10.3219 10.14608 10.0544 9.90265 9.64766
## Proportion of Variance  0.01594  0.0156  0.01507  0.0148 0.01436 0.01363
## Cumulative Proportion   0.63229  0.6479  0.66296  0.6778 0.69212 0.70575
##                           PC25    PC26    PC27   PC28    PC29    PC30    PC31
## Standard deviation     9.50764 9.33253 9.27320 9.0900 8.98117 8.75003 8.59962
## Proportion of Variance 0.01324 0.01275 0.01259 0.0121 0.01181 0.01121 0.01083
## Cumulative Proportion  0.71899 0.73174 0.74433 0.7564 0.76824 0.77945 0.79027
##                           PC32    PC33    PC34    PC35    PC36    PC37    PC38
## Standard deviation     8.44738 8.37305 8.21579 8.15731 7.97465 7.90446 7.82127
## Proportion of Variance 0.01045 0.01026 0.00988 0.00974 0.00931 0.00915 0.00896
## Cumulative Proportion  0.80072 0.81099 0.82087 0.83061 0.83992 0.84907 0.85803
##                           PC39    PC40    PC41   PC42    PC43   PC44    PC45
## Standard deviation     7.72156 7.58603 7.45619 7.3444 7.10449 7.0131 6.95839
## Proportion of Variance 0.00873 0.00843 0.00814 0.0079 0.00739 0.0072 0.00709
## Cumulative Proportion  0.86676 0.87518 0.88332 0.8912 0.89861 0.9058 0.91290
##                          PC46    PC47    PC48    PC49    PC50    PC51    PC52
## Standard deviation     6.8663 6.80744 6.64763 6.61607 6.40793 6.21984 6.20326
## Proportion of Variance 0.0069 0.00678 0.00647 0.00641 0.00601 0.00566 0.00563
## Cumulative Proportion  0.9198 0.92659 0.93306 0.93947 0.94548 0.95114 0.95678
##                           PC53    PC54    PC55    PC56    PC57   PC58    PC59
## Standard deviation     6.06706 5.91805 5.91233 5.73539 5.47261 5.2921 5.02117
## Proportion of Variance 0.00539 0.00513 0.00512 0.00482 0.00438 0.0041 0.00369
## Cumulative Proportion  0.96216 0.96729 0.97241 0.97723 0.98161 0.9857 0.98940
##                           PC60    PC61    PC62    PC63      PC64
## Standard deviation     4.68398 4.17567 4.08212 4.04124 2.122e-14
## Proportion of Variance 0.00321 0.00255 0.00244 0.00239 0.000e+00
## Cumulative Proportion  0.99262 0.99517 0.99761 1.00000 1.000e+00
```

```r
plot(pr.out)
```



Note that the height of each bar in the bar plot is given by squaring the corresponding element of pr.out$sdev. However, it is more informative to plot the PVE of each principal component (i.e. a scree plot) and the cumulative PVE of each principal component. This can be done with just a little work.

```r
pve <- 100 * pr.out$sdev^2 / sum(pr.out$sdev^2)

par(mfrow = c(1, 2))
plot(pve, type = "o", ylab = "PVE", xlab = "Principal Component", col = "blue")
plot(cumsum(pve), type = "o", ylab = "Cumulative PVE",
     xlab = "Principal Component", col = "brown3")
```

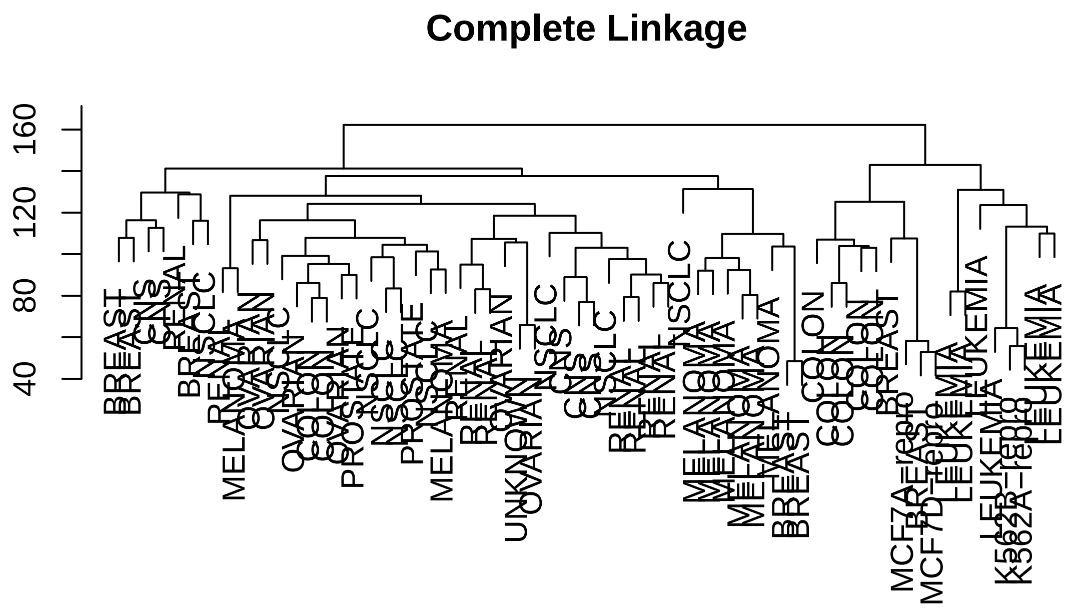


We see that together, the first seven principal components explain around 40 % of the variance in the data. This is not a huge amount of the variance. However, looking at the scree plot, we see that while each of the first seven principal components explain a substantial amount of variance, there is a marked decrease in the variance explained by further principal components. That is, there is an elbow in the plot after approximately the seventh principal component. This suggests that there may be little benefit to examining more than seven or so principal components.

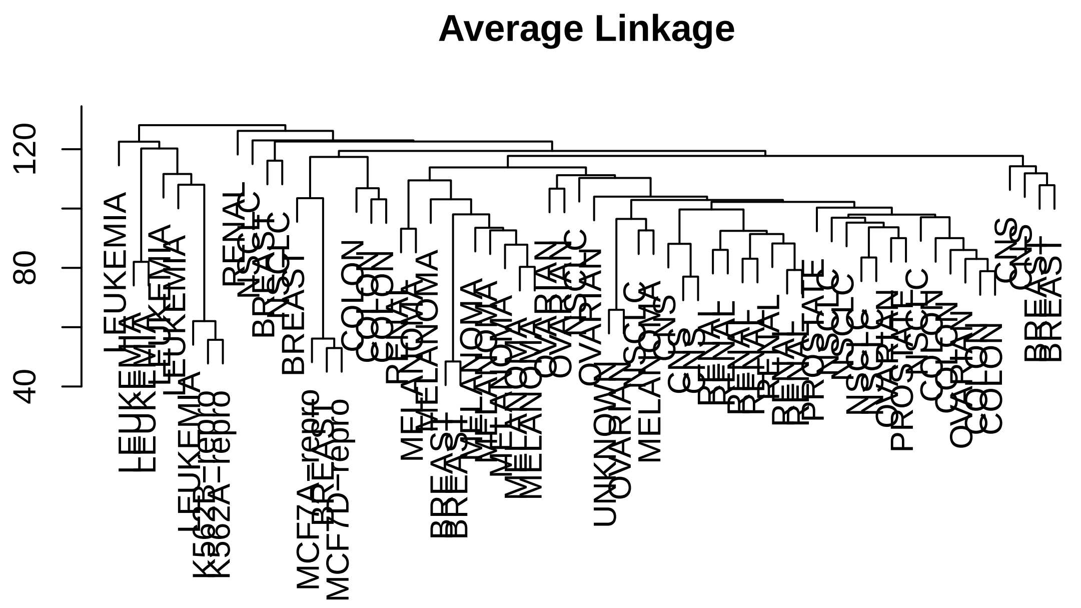
2.3 Hierarchical Clustering on all data

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. To begin, we standardize the variables to have mean zero and standard deviation one. We then perform hierarchical clustering of the observations using complete, single, and average linkage. Euclidean distance is used as the dissimilarity measure.

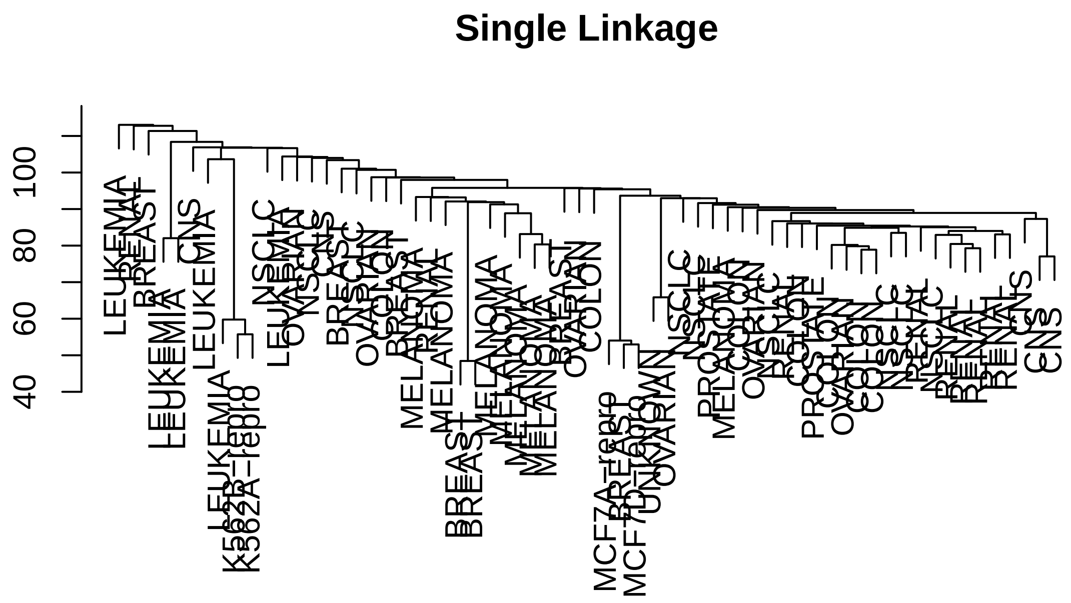
```
sd.data <- scale(nci.data)

data.dist <- dist(sd.data)
plot(hclust(data.dist), xlab = "", sub = "", ylab = "",
     labels = nci.labs, main = "Complete Linkage")
```



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



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



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.

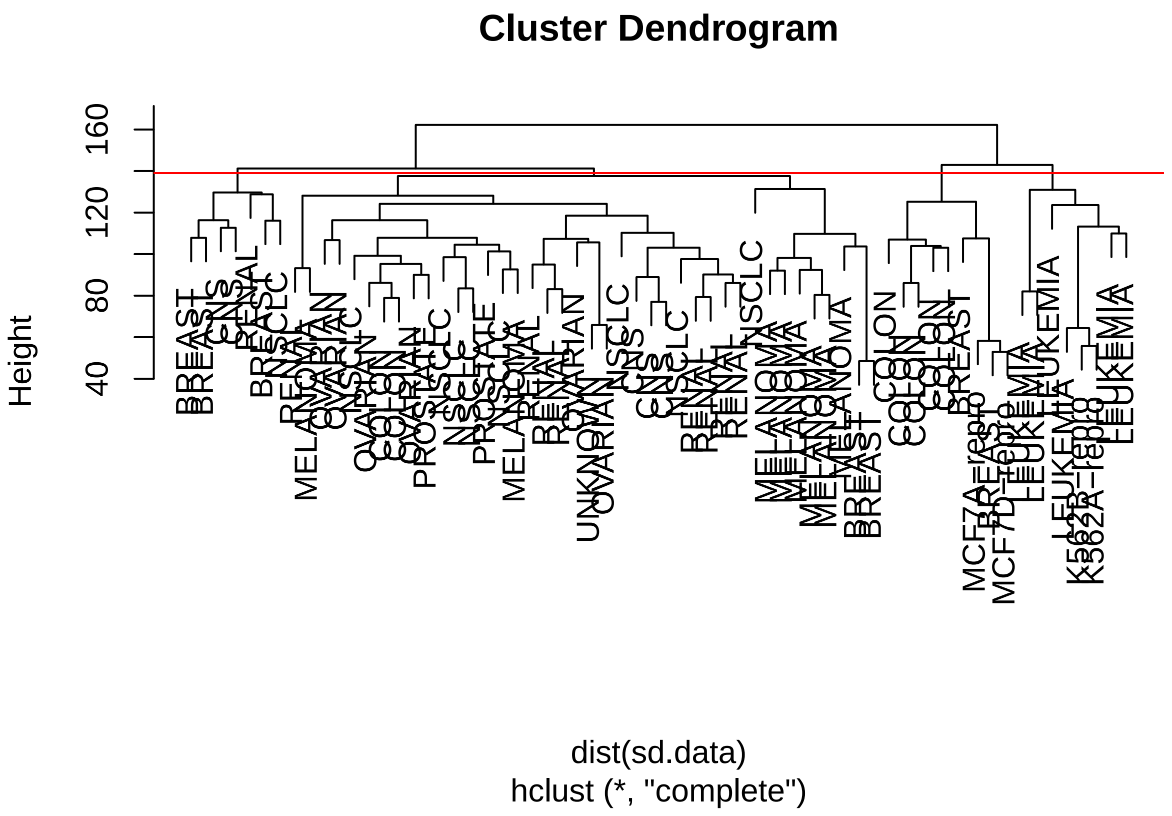
We can cut the dendrogram at the height that will yield a particular number of clusters, say four.

```
hc.out <- hclust(dist(sd.data))
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
```

There are some clear patterns. All the leukemia cell lines fall in cluster 3, while the breast cancer cell lines are spread out over three different clusters. We can plot the cut on the dendrogram that produces these four clusters.

```
par(mfrow = c(1, 1))
plot(hc.out, labels = nci.labs)
abline(h = 139, col = "red")
```



The argument h = 139 plots a horizontal line at height 139 on the dendrogram; this is the height that results in four distinct clusters. It is easy to verify that the resulting clusters are the same as the ones we obtained using "cutree(hc.out, 4)' '.

2.4 K-means Clustering on all data

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

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

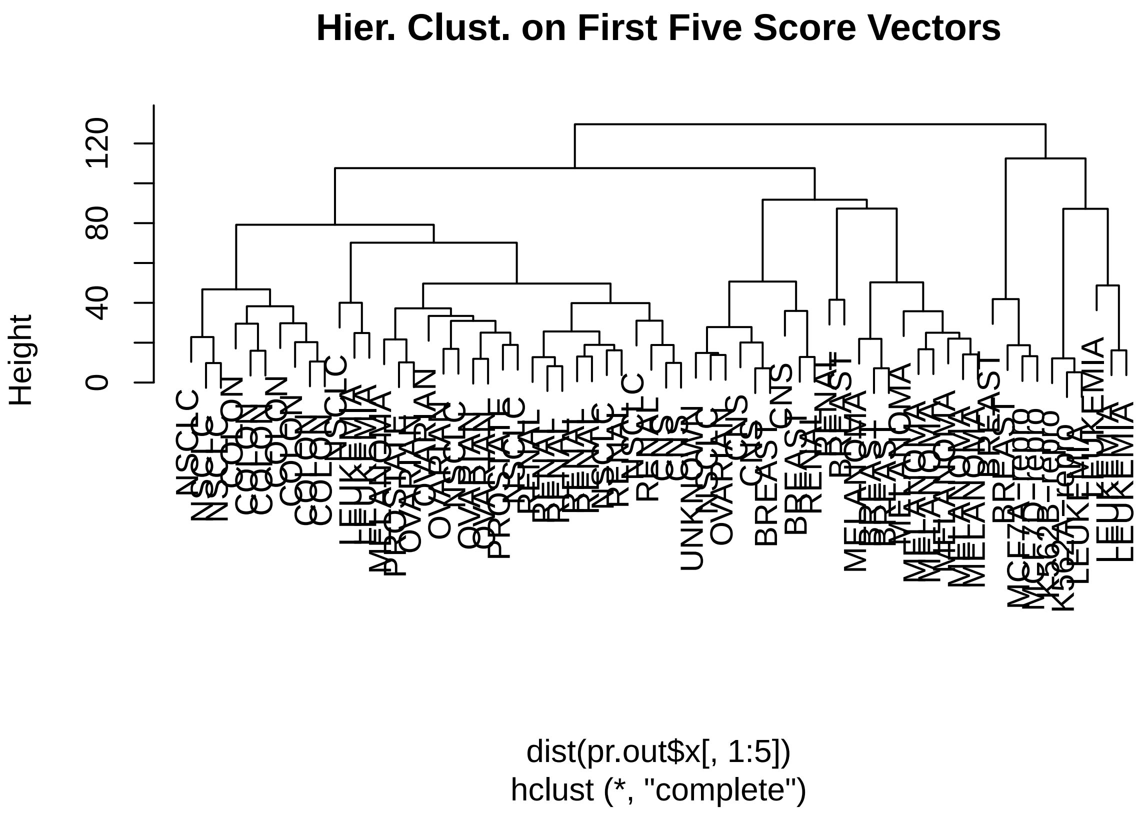
```
##            hc.clusters
## km.clusters  1  2  3  4
##           1  0  0  8  0
##           2 11  0  0  9
##           3 20  7  0  0
##           4  9  0  0  0
```

We see that the four clusters obtained using hierarchical clustering and K-means clustering are somewhat different. Cluster 2,3,4 in hierarchical clustering is identical to cluster 3,1,2 in K-means clustering. However, Cluster 1 in hierarchical clustering distributes into cluster 2,3,4 in K-means clustering.

2.5 Clustering on PCs

Rather than performing hierarchical clustering on the entire data matrix, we can simply perform hierarchical clustering on the first few principal component score vectors.

```r
hc.out <- hclust(dist(pr.out$x[, 1:5]))
plot(hc.out, labels = nci.labs, main = "Hier. Clust. on First Five Score Vectors")
```



```r
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      0   2     7           0           0        2           0
##           2      5   3     0           0           0        0           0
##           3      0   0     0           1           1        4           0
##           4      2   0     0           0           0        0           1
##            nci.labs
## hc.clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
##           1           0        1     8       5        2     7       0
##           2           0        7     1       1        0     2       1
##           3           0        0     0       0        0     0       0
##           4           1        0     0       0        0     0       0
```

Not surprisingly, these results are different from the ones that we obtained when we performed hierarchical clustering on the full data set. Sometimes performing clustering on the first few principal component score vectors can give better results than performing clustering on the full data. In this situation, we might view the principal component step as one of denoising the data.

We could also perform K-means clustering on the first few principal component score vectors rather than the full data set.

```r
set.seed(2)
km.out <- kmeans(pr.out$x[, 1:5], 4, nstart = 20)
km.clusters <- km.out$cluster
```

```
table(km.clusters, nci.labs)
```

```
##            nci.labs
## km.clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
##           1      3   5     0           0           0        0           0
##           2      2   0     0           0           0        0           0
##           3      0   0     0           1           1        5           0
##           4      2   0     7           0           0        1           1
##            nci.labs
## km.clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
##           1           0        1     4       3        1     9       1
##           2           0        7     0       0        0     0       0
##           3           0        0     0       0        0     0       0
##           4           1        0     5       3        1     0       0
```

```
table(km.clusters, hc.clusters)
```

```
##            hc.clusters
## km.clusters  1  2  3  4
##           1 16 11  0  0
##           2  0  9  0  0
##           3  1  0  6  0
##           4 17  0  0  4
```