Lab 5: Cluster Analysis Using R and Bioconductor

June 4, 2003

Introduction

In this lab we introduce you to various notions of distance and to some of the clustering algorithms that are available in R. The lab comes in two parts, in the first we consider different distance measures while in the second part we consider the clustering methods. It is worth mentioning that all clustering and machine learning methods rely on some measure of distance. Almost all implementations have some measure that they use as a default (this is usually Euclidean). As a user of this software you need to decide what distance measures are of interest or are relevant for your situation and to subsequently ensure that the clustering methods use the distance metric that **you** believe is appropriate.

The examples in this section will rely on the data reported in **?**. The basic comparisons here are between B-cell derived ALL, T-cell derived ALL and AML. In some cases we will just compare ALL to AML while in others comparisons between the three groups will be considered.

```
> library(Biobase)
> library(annotate)
> library(golubEsets)
> library(genefilter)
> library(ellipse)
> library(lattice)

Loading required package: grid

> library(cluster)
> library(mva)
```

We transform the data in much the same way that **?** did. The sequence of commands needed are given below.

```
> data(golubTrain)
> X <- exprs(golubTrain)
> X[X < 100] <- 100
> X[X > 16000] <- 16000
> mmfilt <- function(r = 5, d = 500, na.rm = TRUE) {
+     function(x) {
+         minval <- min(x, na.rm = na.rm)
+         maxval <- max(x, na.rm = na.rm)
+         (maxval/minval > r) && (maxval - minval > d)
+     }
+ }
> mmfun <- mmfilt()
> ffun <- filterfun(mmfun)
> sub <- genefilter(X, ffun)
> sum(sub)

[1] 3051

> X <- X[sub, ]
> X <- log2(X)
> golubTrainSub <- golubTrain[sub, ]
> golubTrainSub@exprs <- X
> Y <- golubTrainSub$ALL.AML
> Y <- paste(golubTrain$ALL.AML, golubTrain$T.B.cell)
> Y <- sub("NA", "", Y)
```

Now that the data are set up, we are ready to compute distances between tumor samples and apply clustering procedures to these samples.

1 Distances

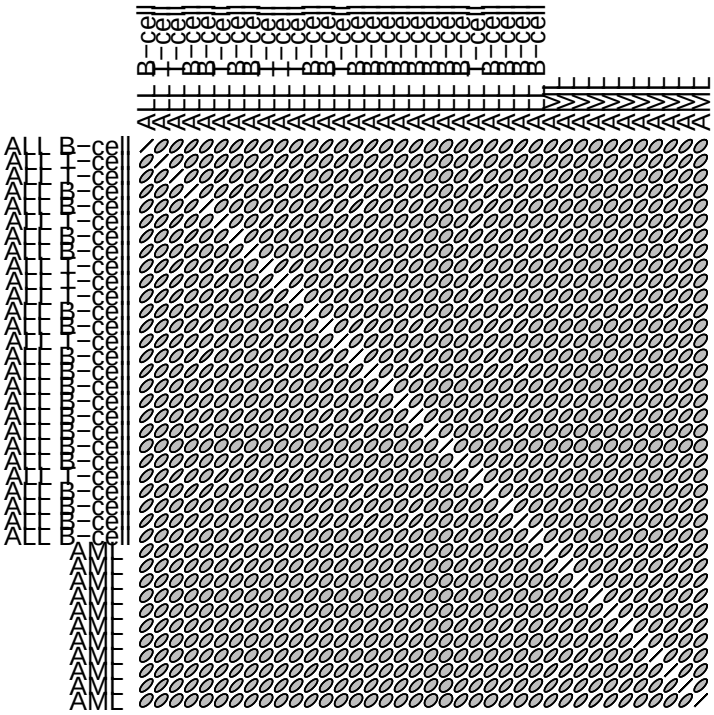
We first compute the correlation distance between the 38 tumor samples from the training set, using all 3,051 genes retained after the above filtering procedure.

```
> r <- cor(X)
> dimnames(r) <- list(as.vector(Y), as.vector(Y))
> d <- 1 - r
```

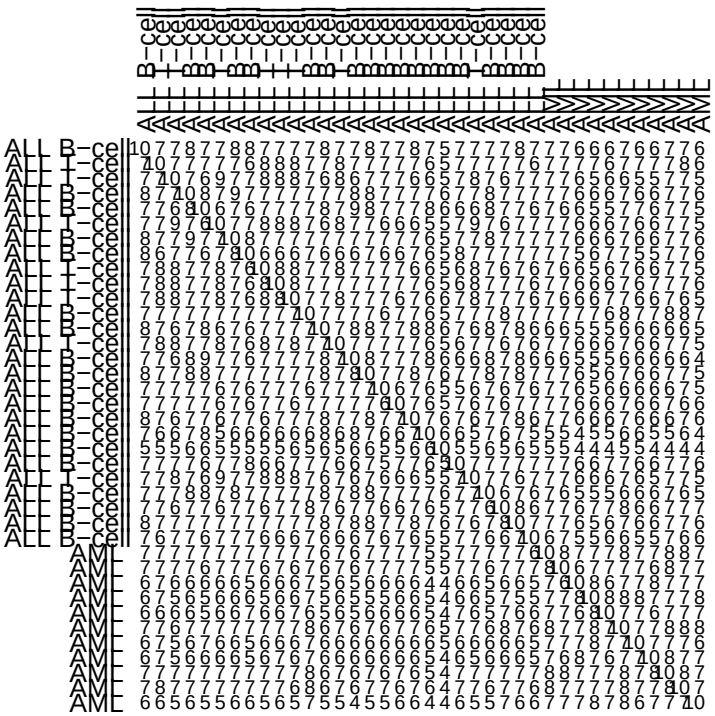
We rely on the `plotcorr` function from the *ellipse* package and the `levelplot` function from *lattice* for displaying this 38 $\times$38 distance matrix.

```
> plotcorr(r, main = "Leukemia data: Correlation matrix for 38 mRNA samples\n All 3,0
```

Leukemia data: Correlation matrix for 38 mRNA samples
All 3,051 genes




```
> plotcorr(r, numbers = TRUE, main = "Leukemia data: Correlation matrix for 38 mRNA s
```




```
> levelplot(r, col.region = heat.colors(50), main = "Leukemia data: Correlation matri
```

We can see in this plots that there is not a strong grouping of the ALL and AML samples. We now consider selecting genes that are good differentiators between the two groups. We use a t-test filter and require that the p-value be below 0.01 for selection (no adjustment has been made).

```
> gtt <- ttest(golubTrainSub$ALL, p = 0.01)
> gf1 <- filterfun(gtt)
> whT <- genefilter(golubTrainSub, gf1)
> gTrT <- golubTrainSub[whT, ]
```

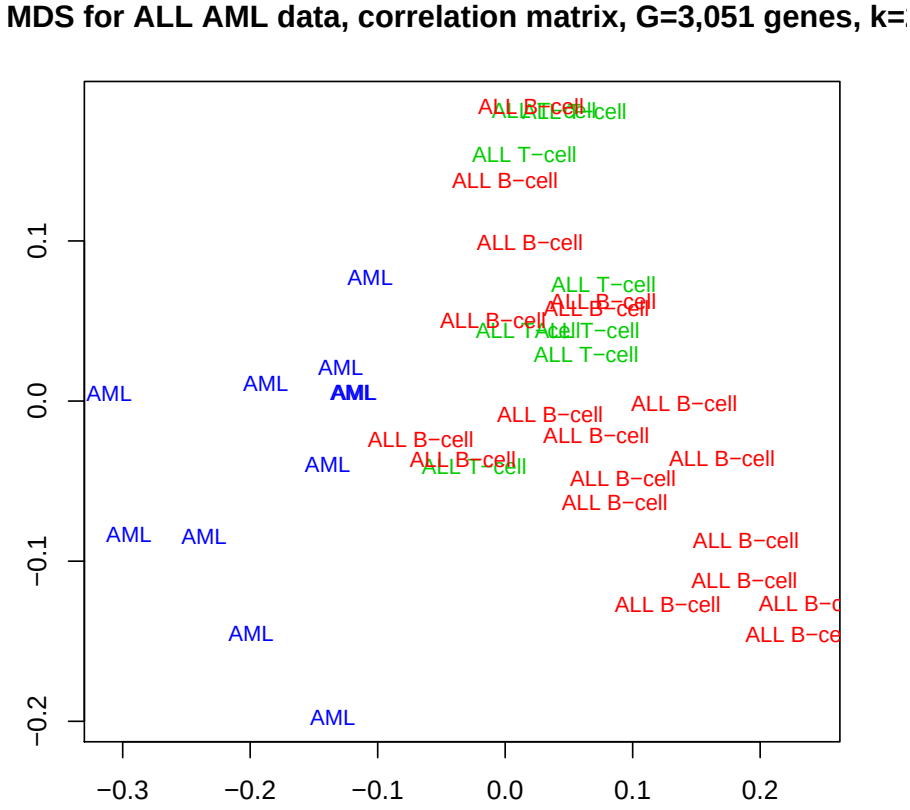
Now we can repeat the above computations and plots using the data in `gTrT`. There were 609 genes selected using the t-test criterion above.

```
> rS <- cor(exprs(gTrT))
> dimnames(rS) <- list(gTrT$ALL, gTrT$ALL)
> dS <- 1 - rS
> plotcorr(r, main = "Leukemia data: Correlation matrix for 38 mRNA samples\n 609 gen
```

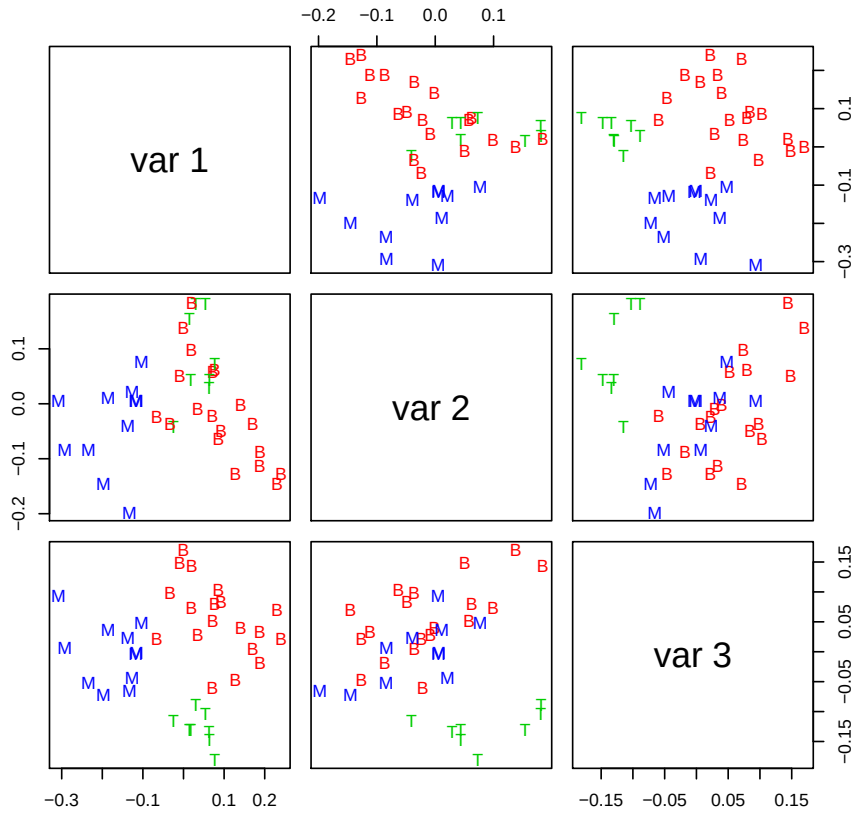
2 Multidimensional Scaling

We now turn to *multidimensional scaling* (MDS) for representing the tumor sample distance matrix. MDS is a data reduction method that is appropriate for general distance matrices. It is much like principal component analysis (PCA), but it is not the same – except for Euclidean distances. Given an $n \times n$ distance matrix, MDS is concerned with identifying n points in Euclidean space with a *similar* distance structure. The purpose is to provide a lower dimensional representation of the distances which can better convey the information on the relationships between objects, such as the existence of clusters or one-dimensional structure in the data (e.g., seriation). In most cases we would like to find a two or three dimensional reduction that could easily be visualized.

```
> mds <- cmdscale(d, k = 2, eig = TRUE)
> plot(mds$points, type = "n", xlab = "", ylab = "", main = "MDS for ALL AML data, co
> text(mds$points[, 1], mds$points[, 2], Y, col = as.integer(factor(Y)) +
+     1, cex = 0.8)
```



```
> mds3 <- cmdscale(d, k = 3, eig = TRUE)
> pairs(mds3$points, main = "MDS for ALL AML data, correlation matrix, G=3,051 genes,
+     pch = c("B", "T", "M")[as.integer(factor(Y))], col = as.integer(factor(Y)) +
+         1)
```



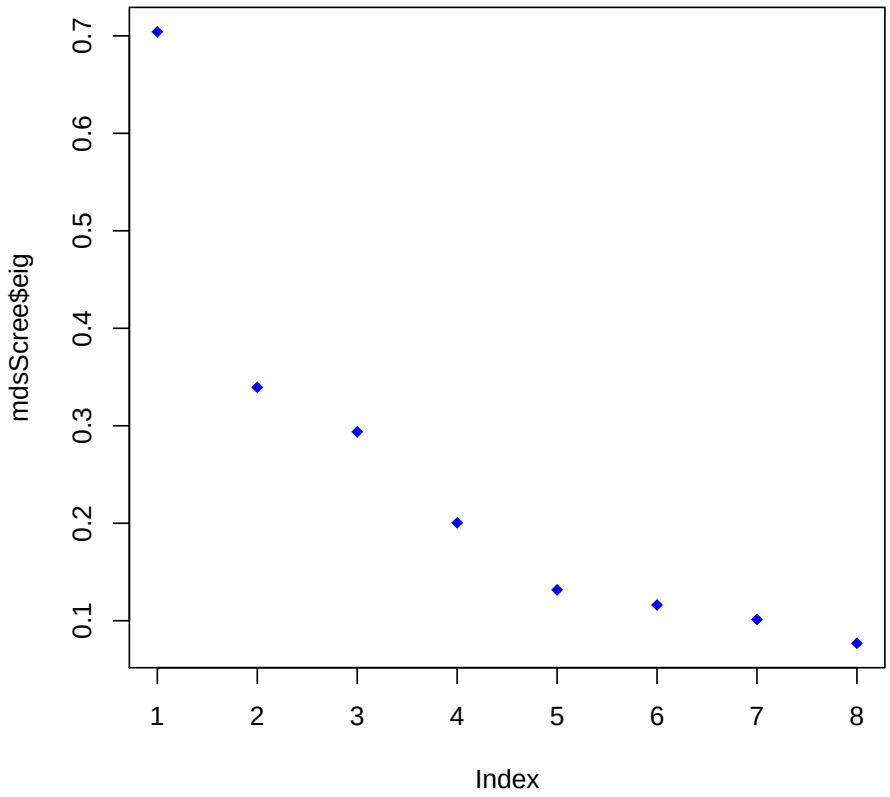

```
> if (require(rgl) && interactive()) {
+     rgl.spheres(x = mds3$points[, 1], mds3$points[, 2], mds3$points[,
+         3], col = as.integer(factor(Y)) + 1, radius = 0.01)
+ }
```

```
Loading required package: rgl
```

Note, if we are measuring distances between samples on the basis of $G = 3,051$ probes then we are essentially looking at points in 3,051 dimensional space. As with PCA, the quality of the representation will depend on the magnitude of the first k eigenvalues. To assess how many components there are, a *scree* plot similar to that used for principal components can be created. This plot of the eigenvalues suggests that much of the

information is contained in the first component. One might consider using either three or four components as well.

```
> mdsScree <- cmdscale(d, k = 8, eig = TRUE)
> plot(mdsScree$eig, pch = 18, col = "blue")
```



EDD and ROC

In this section we consider a couple of other tools for visualizing genomic data. The first is *edd* which stands for expression density diagnostics. The idea behind edd is to look for genes whose patterns of expression are quite different in two groups of interest.

To assess whether two genes have different patterns of expression edd transforms all expression values to the range 0,1 and then performs density estimation on them. Thus it is looking for differences in shape and ignores differences in scale or location. In some situations location and scale are more important (however there are many different tools that can be used in those situations).

In this case we will use *edd* to compare those patients with AML to those with B-cell derived ALL. We will rely on the merged data set so that we have a number of samples from each group (38 ALLs and 25 AMLs).

Once we have subset the data to the appropriate cases we next perform some non-specific filtering to remove any genes that show little variation across samples. This is an important step when using *edd*. Since *edd* standardizes the genes (across samples) to give approximately common centering and scaling non-informative genes will not be distinguishable from informative ones. Therefore, we must explicitly remove the non-informative ones before applying *edd*.

```
> gMs <- golubMerge[, (golubMerge$T.B != "T-cell" | is.na(golubMerge$T.B))]
> fF1 <- gapFilter(100, 200, 0.1)
> ff <- filterfun(fF1)
> whgMs <- genefilter(gMs, ff)
> sum(whgMs)
```

```
[1] 3082
```

```
> gMs <- gMs[whgMs, ]
> library(edd)
```

```
Loading required package: nnet
Loading required package: class
Loading required package: xtable
```

We next split the data into two expression sets, one with the AML and the other with the B-cell ALL samples. For each set we use `edd` to classify samples into one of a set of predetermined shapes.

We will use `edd` to perform the classification and will use the default values of most of the parameters. That means that we will be using the `multiCand` for candidate comparison and `knn` for classification.

When `multicand` is selected edd selects 100 samples from each of the eight candidate densities. Each of sample is scaled to have median 0 and mad 1. These constitute the reference samples and for each gene we will perform the same transformation and then compare the resulting data to the reference samples. Any machine learning technique can in principle be used to make this comparison. In this example we will use the default classifier which is `knn`.

The density, across samples, for each gene is found and scaled to have median 0 and mad 1 (as for the reference samples).

```
> gMsALL <- gMs[, gMs$ALL == "ALL"]
> gMsAML <- gMs[, gMs$ALL == "AML"]
> set.seed(1234)
> edd.ALL <- edd(gMsALL, method = "knn", k = 4, l = 2)
> sum(is.na(edd.ALL))
```

```
[1] 29
```

```
> edd.AML <- edd(gMsAML, method = "knn", k = 4, l = 2)
> sum(is.na(edd.AML))
```

```
[1] 71
```

```
> table(edd.ALL, edd.AML)
```

```
                     edd.AML
edd.ALL               .25N(0,1)+.75N(4,1) .75N(0,1)+.25N(4,1) B(2,8) B(8,2)
  .25N(0,1)+.75N(4,1)   0                   4                   3      3
  .75N(0,1)+.25N(4,1)   2                  28                  69     10
  B(2,8)                3                  88                 243     40
  B(8,2)                1                   5                   8      9
  logN(0,1)             1                  45                 102     14
  N(0,1)                4                  27                  94     34
  t(3)                 13                  63                 103     50
  U(0,1)                0                   6                   4      2
  X^2(1)                1                  42                  48      5
                     edd.AML
edd.ALL               logN(0,1) N(0,1) t(3) U(0,1) X^2(1)
  .25N(0,1)+.75N(4,1)   3         5     10    0      1
  .75N(0,1)+.25N(4,1)  20        40     40   17     15
  B(2,8)               66       167    150   58     46
  B(8,2)                3        20     11    4      1
  logN(0,1)            79        60     70   15     58
  N(0,1)               20       118     74   30     17
  t(3)                 60       116    153   38     28
  U(0,1)                3         9      5    3      0
  X^2(1)               68        25     29   11     45
```

For 29 of the ALL patients no classification was made. So properly they should be classified as *doubt*. For the AML group 71 were classified as *doubt*. We can see from the resulting table that there are a number of genes for which the ALL classification is a mixture (`mix1`) while the AML classification is as a Normal random variable. To further explore these will need to make use of some plots.

```
> diff1 <- edd.AML == "N(0,1)" & edd.ALL == ".75N(0,1)+.25N(4,1)"
> table(diff1)
```

```
diff1
FALSE  TRUE
 3021    40
```

```
> diff1[is.na(diff1)] <- FALSE
> AMLexprs <- exprs(gMsAML)[diff1, ]
> ALLexprs <- exprs(gMsALL)[diff1, ]
```

Now we have the two sets of expression data and we can see what the differences are.

```
> tAML <- fq.matrows(t(apply(AMLexprs, 1, centerScale)))
> tALL <- fq.matrows(t(apply(ALLexprs, 1, centerScale)))
> par(mfrow = c(2, 2))
> hist(AMLexprs[1, ])
> hist(ALLexprs[1, ])
> hist(AMLexprs[2, ])
> hist(ALLexprs[2, ])
> par(mfrow = c(1, 1))
```

ROC

Receiver operator curves are a tool that can be used to assess the classificaton capabilities of a covariate. In situations where there are two classes (for example, AML versus ALL) then one can consider splitting the data into two groups depending on the value of the covariate. The relative proportions of AML and ALL in the two resulting groups can be used to assess the capability of that covariate for splitting the data.

Hierarchical Clustering

We can use the `hclust` function in the *mva* package for *agglomerative hierarchical clustering*. We consider the following agglomeration methods, i.e., methods for computing *between cluster distances*:

Single linkage The distance between two clusters is the minimum distance between any two objects, one from each cluster.

Average linkage The distance between two clusters is the average of all pairwise distances between the members of both clusters.

Complete linkage The distance between two clusters is the maximum distance between two objects, one from each cluster.

The nested sequence of clusters resulting from hierarchical clustering methods can be represented graphically using a dendrogram. A *dendrogram* is a binary tree diagram in which the terminal nodes, or leaves, represent individual observations and in which the height of the nodes reflects the dissimilarity between the two clusters that are joined.

While dendrograms are quite appealing because of their apparent ease of interpretation, they can be misleading.

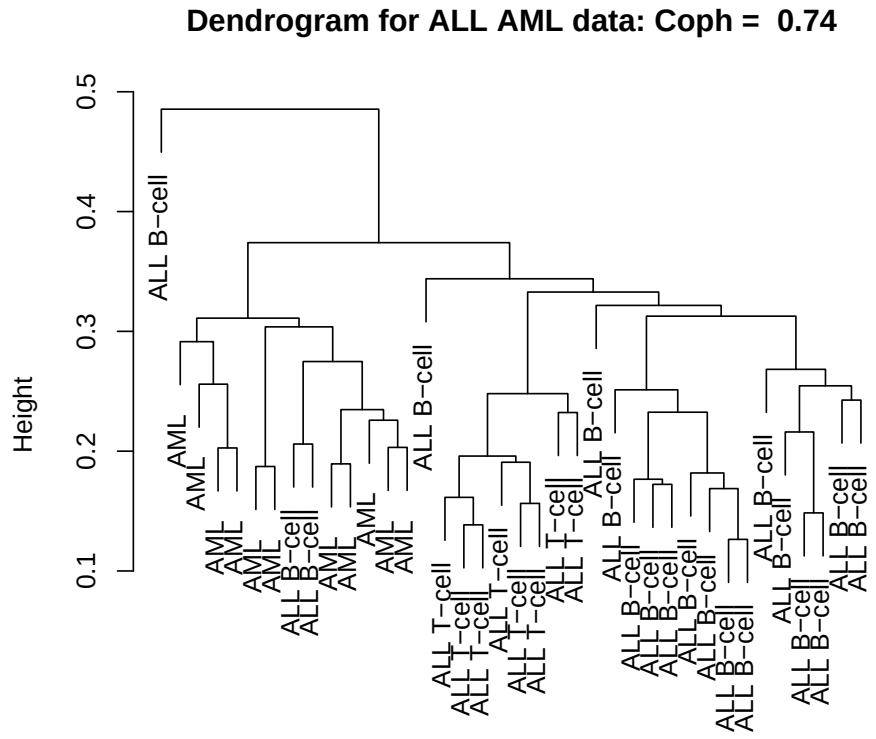
First, the dendrogram corresponding to a given hierarchical clustering is not unique, since for each merge one needs to specify which subtree should go on the left and which on the right. For n observations, there are $n-1$ merges and hence $2^{(n-1)}$ possible orderings of the terminal nodes. Therefore, $2^{(n-1)}$ different dendrograms are consistent with any given sequence of hierarchical clustering operations. The ordering is of practical importance because it will result in different genes being placed next to each other in the heat-maps and possibly different interpretations of the same data. A number of heuristics have been suggested for ordering the terminal nodes in a dendrogram. The default in the R function `hclust` from the *cluster* package is to order the subtrees so that the tighter cluster is on the left (i.e., the most recent merge of the left subtree is at a lower value than the last merge of the right subtree).

A second, and perhaps less recognized shortcoming of dendrograms, is that they *impose* structure on that data, instead of *revealing* structure in these data. Indeed, dendrograms are often viewed as graphical summaries of the data, rather than of the results of the clustering algorithm. Such a representation will be valid only to the extent that the pairwise dissimilarities possess the hierarchical structure imposed by the clustering algorithm. Note that in particular, the same matrix of pairwise distances between observations will be represented by a different dendrogram depending on the distance function that is used to compute between-cluster distances (e.g., complete or average linkage). The *cophenetic correlation coefficient* can be used to measure how well the hierarchical structure from the dendrogram represents the actual distances. This measure is defined as the correlation between the $n(n-1)/2$ pairwise dissimilarities between observations and their *cophenetic* dissimilarities from the dendrogram, i.e., the between cluster dissimilarities at which two observations are first joined together in the same cluster.

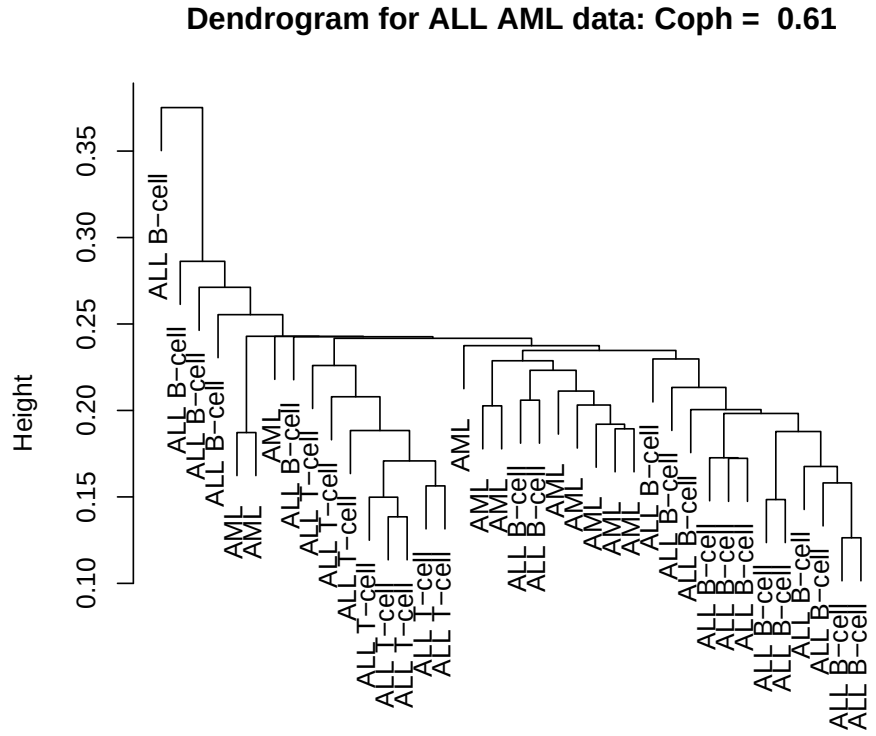
Next, we apply three agglomerative hierarchical clustering methods to cluster tumor samples. For each methods, we plot a dendrogram, compute the corresponding cophenetic correlation, and compare the three clusters obtained from cutting the tree into three branches to the actual tumor labels (ALL B-cell, ALL T-cell, and AML).

```
> hc1 <- hclust(as.dist(d), method = "average")
> coph1 <- cor(cophenetic(hc1), as.dist(d))
> plot(hc1, main = paste("Dendrogram for ALL AML data: Coph = ",
+     round(coph1, 2)), sub = "Average linkage, correlation matrix, G=3,051 genes")
> cthc1 <- cutree(hc1, 3)
> table(Y, cthc1)

            cthc1
Y            1  2  3
  ALL B-cell 16  2  1
```

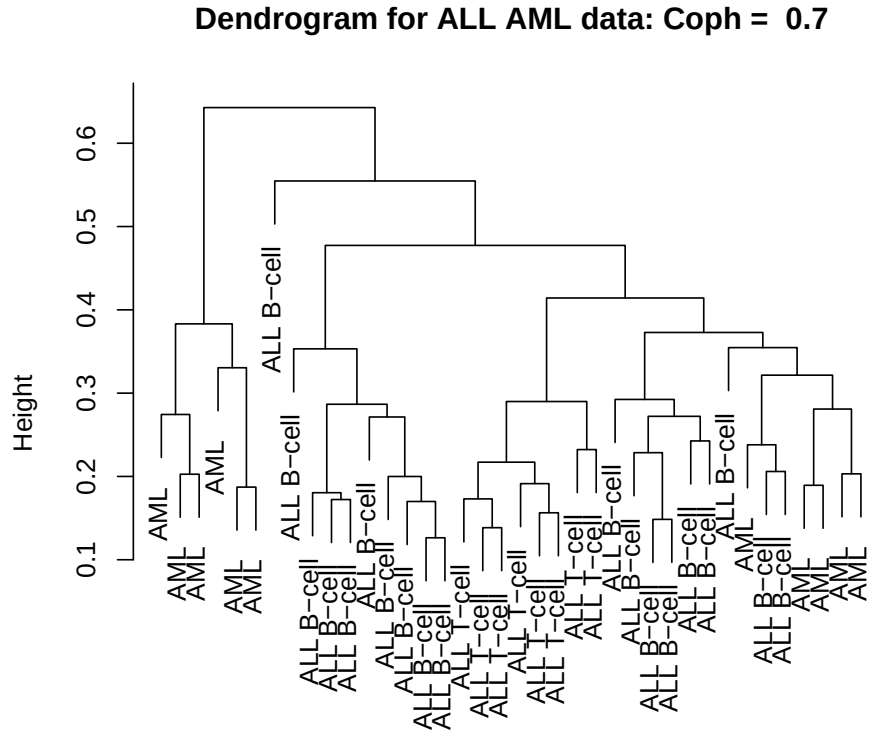
```
  ALL T-cell  8  0  0
  AML         0 11  0
```




```
> hc2 <- hclust(as.dist(d), method = "single")
> coph2 <- cor(cophenetic(hc2), as.dist(d))
> plot(hc2, main = paste("Dendrogram for ALL AML data: Coph = ",
+     round(coph2, 2)), sub = "Single linkage, correlation matrix, G= 3,051 genes")
> cthc2 <- cutree(hc2, 3)
> table(Y, cthc2)

            cthc2
Y            1  2  3
  ALL B-cell 17  1  1
  ALL T-cell  8  0  0
  AML        11  0  0
```

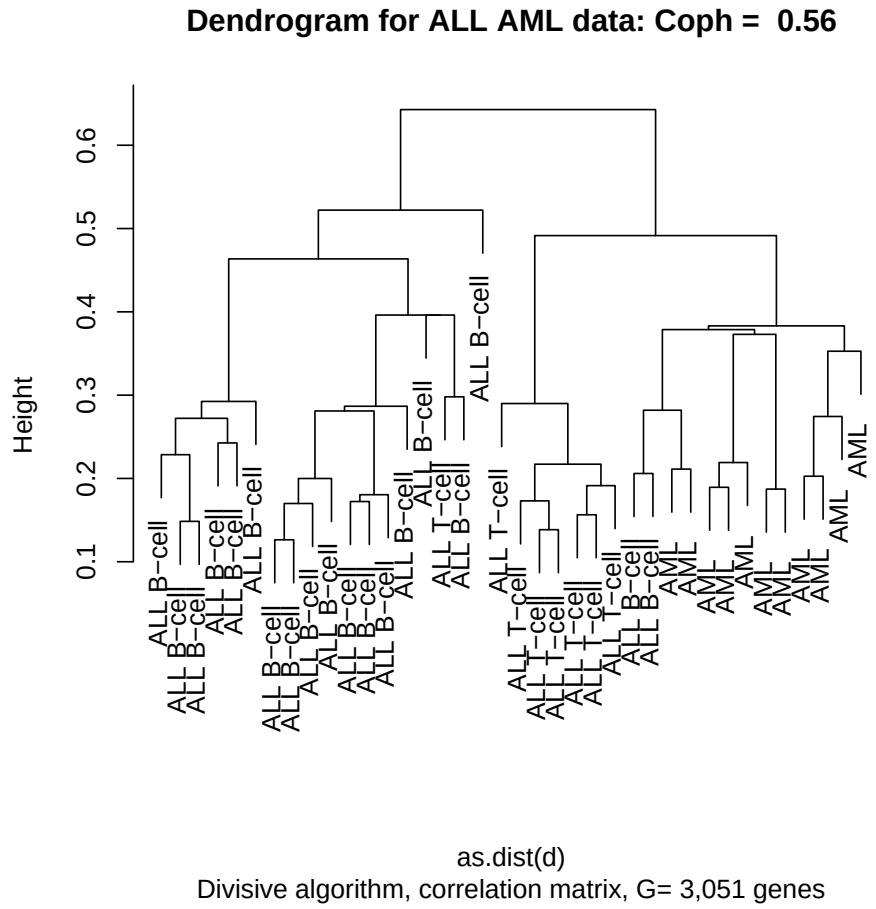



```
> hc3 <- hclust(as.dist(d), method = "complete")
> coph3 <- cor(cophenetic(hc3), as.dist(d))
> plot(hc3, main = paste("Dendrogram for ALL AML data: Coph = ",
+     round(coph3, 2)), sub = "Complete linkage, correlation matrix, G= 3,051 genes")
> cthc3 <- cutree(hc3, 3)
> table(Y, cthc3)

            cthc3
Y            1  2  3
  ALL B-cell 18  1  0
  ALL T-cell  8  0  0
  AML         5  0  6
```



Divisive hierarchical clustering is available using `diana` from the *cluster* package.

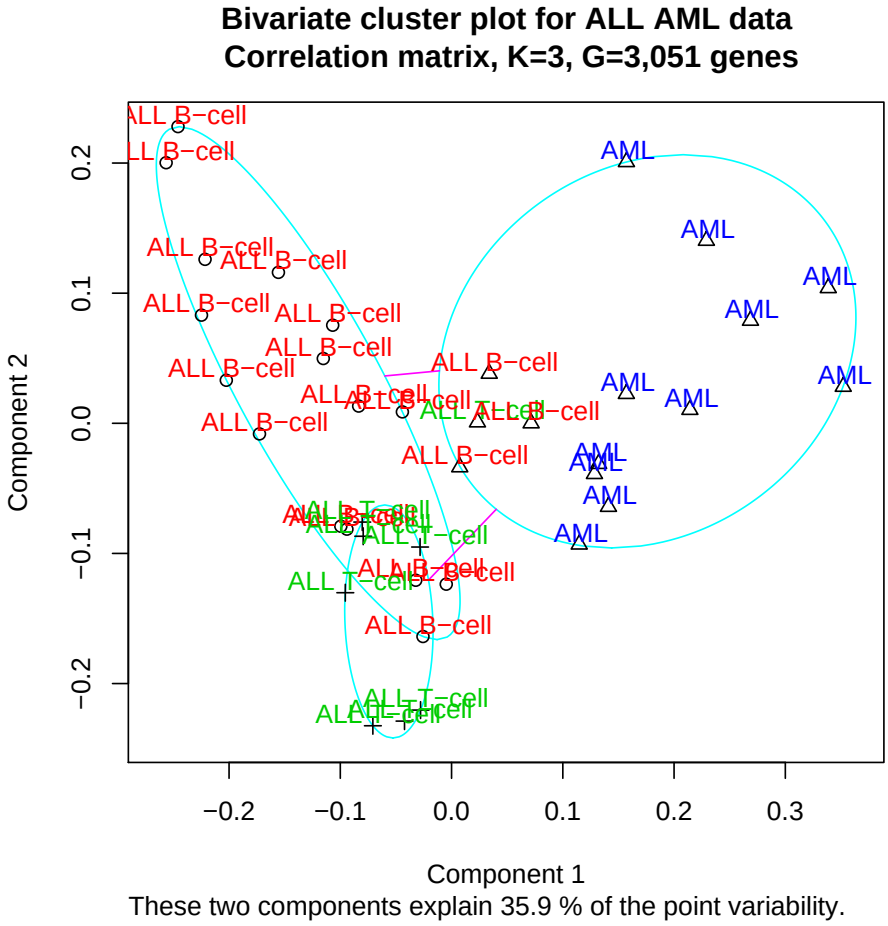
```
> di1 <- diana(as.dist(d))
> cophdi <- cor(cophenetic(di1), as.dist(d))
> plot(di1, which.plots = 2, main = paste("Dendrogram for ALL AML data: Coph = ",
+     round(cophdi, 2)), sub = "Divisive algorithm, correlation matrix, G= 3,051 gene
> ct.di <- cutree(di1, 3)
> table(Y, ct.di)

             ct.di
Y             1  2  3
  ALL B-cell 16  2  1
  ALL T-cell  1  7  0
  AML         0 11  0
```



3 Partitioning Methods

Here we apply the *partitioning around medoids* (PAM) method to cluster tumor mRNA samples.

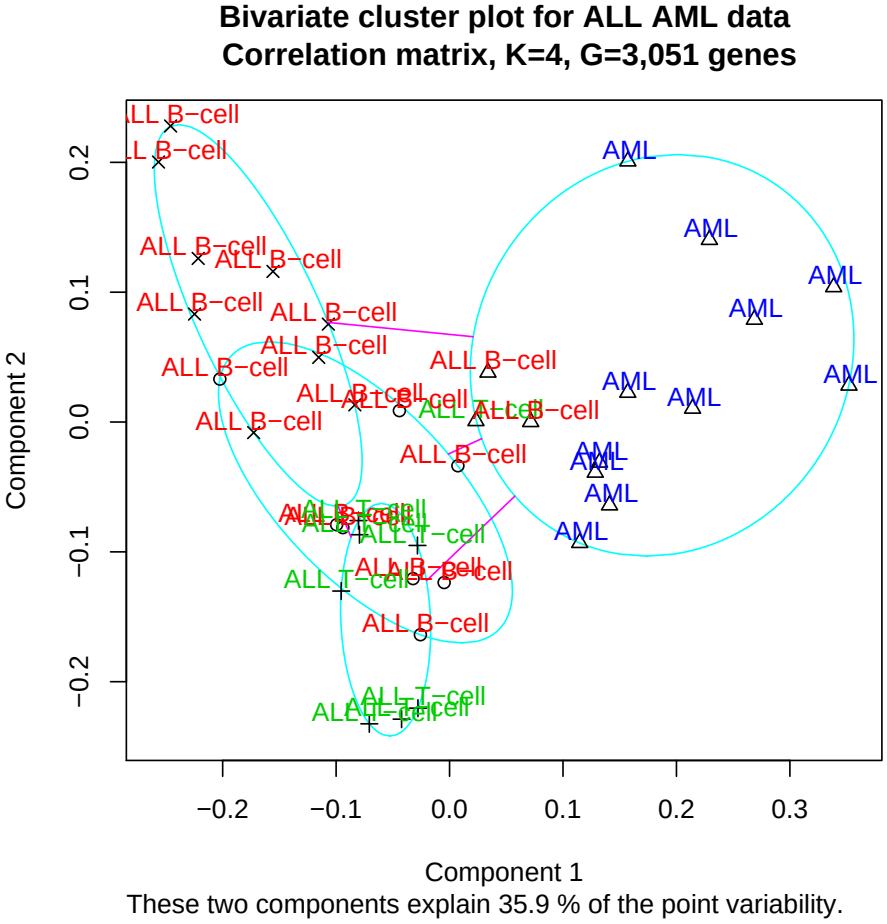
```
> set.seed(12345)
> pm3 <- pam(as.dist(d), k = 3, diss = TRUE)
> table(Y, pm3$clustering)

Y             1  2  3
  ALL B-cell 16  3  0
  ALL T-cell  0  1  7
  AML         0 11  0

> clusplot(d, pm3$clustering, diss = TRUE, labels = 3, col.p = 1,
+     col.txt = as.integer(factor(Y)) + 1, main = "Bivariate cluster plot for ALL AML
```




```
> plot(pm3, which.plots = 2, main = "Silhouette plot for ALL-AML Data")
```

One of the more difficult questions that arises when clustering data is the assessment of how many clusters there are in the data. We now consider using PAM with four clusters.

```
> set.seed(12345)
> pm4 <- pam(as.dist(d), k = 4, diss = TRUE)
> clusplot(d, pm4$clustering, diss = TRUE, labels = 3, col.p = 1,
+     col.txt = as.integer(factor(Y)) + 1, main = "Bivariate cluster plot for ALL AML
```


Bivariate cluster plot for ALL AML data
Correlation matrix, K=4, G=3,051 genes
Component 2
Component 1
These two components explain 35.9 % of the point variability.
ALL B−cell
AML
ALL T−cell
−0.2
−0.1
0.0
0.1
0.2
0.3