1. The goal of this exercise is to assess the stability of the clusters defined by the hierarchical clustering algorithm under small perturbations to the data. Recall that clusters can be defined by making a cut through the hierarchical clustering dendrogram after a specified number of merges.

To identify the clusters at a particular cut of the dendrogram in R, you will need to do some processing with the $x$merge array, where $x$ is the hierarchical clustering object generated by R (i.e. the result of a call of the form $x$ <- hclust(...)). I will also be posting some Matlab hierarchical clustering code. In R, the information in $x$merge has the following meaning (from help(hclust)).

merge: an n-1 by 2 matrix. Row i of 'merge' describes the merging of clusters at step i of the clustering. If an element j in the row is negative, then observation -j was merged at this stage. If j is positive then the merge was with the cluster formed at the (earlier) stage j of the algorithm. Thus negative entries in 'merge' indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.

Generate 20 bootstrapped data sets from the NCI60 lung data by resampling the genes. For each bootstrapped data set, calculate the hierarchical clustering solution. The discrepancy within any pair of solutions can be measured using the proportion of all sample pairs that are assigned to the same class in exactly one of the solutions, relative to the number of sample pairs that are assigned to the same class in at least one of the solutions.

Determine how the mean and variance of the discrepancy measure over all $\binom{20}{2}$ bootstrap sets varies with the level of the dendrogram at which the cut is made, and with the linkage type of the hierarchical clustering algorithm that is used.

Solution:

```r
## The path to all the data.
PA <- '/data/BigFiles/Stat545/NCI60/'

## Read in the expression file.
M <- scan(sprintf('%sNov1', PA), 0)
M <- matrix(M, ncol=59, byrow=TRUE)
```
## Log-transform.
M <- M * (M >= 0)
M <- log(M+1) / log(2)

## Cell line names.
CL <- scan(sprintf('%sCell_lines', PA), '', sep='\n')

## Tissue type labels.
TT <- scan(sprintf('%sTissues', PA), '', sep='\n')

## Unique tissue type labels.
UT <- unique(TT)

## Return a list of cluster indicators after K joins. This performs
## the same function as 'cutree'.
CL <- function(m, K)
{
## A list of cluster labels for each sample.
  ci <- 59+seq(59)

## The current label for the cluster formed at stage j.
  b <- seq(59)

  for (k in (1:K))
  {
## Merge two singletons to create a new cluster.
    if (m[k,2] < 0)
    {
      ci[abs(m[k,])] <- b[k]
    }

## Merge a singleton with a non-singleton
    else if (m[k,1] < 0)
    {
      ci[abs(m[k,1])] <- b[m[k,2]]
      b[k] <- b[m[k,2]]
    }

## Merge two non-singletons.
    else
    {
      ci[ci == b[m[k,2]]] <- b[m[k,1]]
      b[m[k,2]] <- b[m[k,1]]
      b[k] <- b[m[k,1]]
    }
  }
}

2
## Count the proportion of distinct object pairs that are assigned to
## the same class in either C1 or C2 but not both relative to all
## pairs that are assigned to the same class in either C1 or C2. C1
## and C2 are the class label vectors for the objects under two
## clustering solutions.
disc <- function(C1, C2)
{
  n <- length(C1)

  f <- 0
  m <- 0

  for (i in (1:(n-1)))
  {
    for (j in ((i+1):n))
    {
      j1 <- (C1[i] == C1[j])
      j2 <- (C2[i] == C2[j])

      if (j1 | j2) { m <- m+1 }
      if ( (j1 | j2) & (!(j1 & j2)) ) { f <- f+1 }
    }
  }

  f/m
}

## Generate clustering solutions for bootstrapped data sets.
L <- array(0, c(58*20,2))
for (r in (1:20))
{
  ## Generate a bootstrapped data set.
  n <- dim(M)[1]
  ii <- ceiling(n*runif(n))
  B <- M[ii,]

  ## Try one or the other of these as a dissimilarity measure.
  D <- dist(t(B))
  ##D <- as.dist(1 - cor(M))

  x <- hclust(D, method = "single", members=NULL)
  L[(58*(r-1)+1):(58*r),] <- x$merge
}
## Calculate the discrepancy measure for every pair of bootstrapped data sets, and at every possible cutting level in the tree.

```r
F <- array(0, c(190,58))
for (K in 1:58)
{
  ## Loop through all pairs among the 20 bootstrapped sets.
  ii <- 1
  for (r1 in 1:19)
  {
    ## Get the class labels for the first solution.
    M1 <- L[(58*(r1-1)+1):(58*r1),]
    C1 <- CL(M1, K)
    for (r2 in (r1+1):20)
    {
      ## Get the class labels for the second solution.
      M2 <- L[(58*(r2-1)+1):(58*r2),]
      C2 <- CL(M2, K)
      ## Calculate the discrepancy measure.
      F[ii,K] <- disc(C1, C2)
      ii <- ii+1
    }
  }
}
FM <- colMeans(F)
pdf('disc-single.pdf')
plot(FM, xlab='Number of joins', ylab='Discrepancy', main='Single Linkage')
dev.off()
```

The results are plotted below. If we focus on the final ten joins, which is the usual area of interest, there are clear differences among the methods. For complete linkage, only the penultimate join is reasonably well defined (with a discrepancy around 0.1), and the other joins among the top ten are quite ambiguous. For single linkage, all of the final ten joins are quite distinct. As expected, average linkage is somewhat intermediate, with the final three or four joins being reasonably clear.
2. The goal of this exercise is to investigate whether an optimal number of genes can be identified for prediction using the ridged logistic regression classifier applied to the Michigan lung data. You will also compare the performance of supervised gene selection and unsupervised gene selection. It will be important to consider the uncertainty levels of the cross-validated error rate estimates in order to determine whether any meaningful conclusions can be drawn. You can choose any one of the response variables to analyze (we’ll arrange to have each of the four response variables studied by at least two people).

To make things manageable, begin by selecting the genes in the top quarter based on either variance or mean. Next get the cross-validated error rate estimates for prediction using the top 25%, 20%, ..., 5% of these genes based on selection with the Z-statistic (note that these percentages are relative to the reduced set of genes). Also get the cross-validated error rate estimates for prediction using the same number of genes selected
based on overall variability in expression. Do this for penalty parameters \( \lambda = 10, 100, \) and 1000.

It would take too long to do a systematic boostrapping analysis, so we will divide and conquer. There are two main questions of interest: (i) does selection on Z-statistics outperform unsupervised selection based on variances?; (ii) is there a definable “optimal number of genes” that significantly outperforms other choices for the number of genes? Each student will focus on either (i) or (ii), and we’ll coordinate so that both (i) and (ii) are covered for each of the four response variables.

For (i), select the best setting (percentage of genes and the value of \( \lambda \)) for Z-statistic selection and, also the best setting for unsupervised selection. Do 10 (more as time permits) bootstrap runs for both of these settings under both selection procedures (so a total of \( 4 \times 10 = 40 \) bootstrap runs are done, unless the optimal settings for supervised and unsupervised selection coincide). Then state your findings as to what, if any, conclusions can be reliably reached.

For (ii), select the best performing setting out of the 15 supervised selection settings considered above (5 percentages and 3 values for \( \lambda \)). Compare this to the worst performing percentage at the same value of \( \lambda \). Comparison should be based on 10 (more as time permits) bootstrap runs.

Apart from the fact that 10 bootstrap runs are not sufficient for drawing definitive conclusions, the results of both of these analyses (i/ii) are not completely rigorous for another reason. Can you identify this reason, and propose an analysis strategy that would resolve the problem?

**Solution:**

Some of the code that I used is below. For example, using sex as the outcome variable to address (ii), the CV performance anywhere within the top 10% to top 2% of genes is between 63 and 66 correct. Ten bootstrap replications using the top 5% of genes yielded values between 48 and 68, so one could not say that there is a well-defined optimum in the top 10% to 2% range.

```r
## The path to all the data.
PA <- '/data/BigFiles/Stat545/michigan/processed/'

## Read in the expression file.
M <- scan(sprintf('%sGE', PA), 0)
M <- matrix(M, ncol=96, byrow=TRUE)

## Remove the normals.
M <- M[,1:86]
```

## Read in the expression file.
```r
M <- scan(sprintf('%sGE', PA), 0)
M <- matrix(M, ncol=96, byrow=TRUE)
```
## Log-transform.
M <- M * (M >= 0)
M <- log(M+1) / log(2)

## Use stage as the response.
if (0)
{
    Y <- scan(sprintf('%sST', PA), 0)
    ## Relabel to 1/2.
    Y[Y==3] <- 2
}

## Use K-ras mutation status as the response.
if (0)
{
    Y <- scan(sprintf('%sKR', PA), 0)
    ii <- (Y > 0)
    M <- M[,ii]
    Y <- Y[ii]
}

## Use sex as the response.
if (1)
{
    Y <- scan(sprintf('%sSX', PA), 0)
}

## Use age (dichotomized at 60yr) as the response.
if (0)
{
    Y <- scan(sprintf('%sAge', PA), 0)
    Y <- (Y < 60) + 1
}

## Fit a ridged logistic regression model to the data in Y and X,
## using ridging penalty L. The Y values should be coded 0/1, and the
## first column of X is assumed to be identically 1 (for the
## intercept). Note that X is laid out in the usual regression format
## where each row is an observation and each column is a variable.
plogist <- function(Y, X, L)
{

## Number of observations (n) and number of features (p).

\[
n \leftarrow \text{length}(Y)
\]

\[
p \leftarrow \text{dim}(X)[2]
\]

## The regression coefficients.

\[
B \leftarrow \text{array}(0, \text{c}(p,1))
\]

\[
B[1] \leftarrow 1
\]

while (1)
{
    \[
    BX \leftarrow \exp(X \times B)
    \]
    \[
    C0 \leftarrow 1 + BX
    \]
    \[
    C1 \leftarrow BX / C0
    \]
    \[
    C2 \leftarrow C1 / C0
    \]
    
    ## The gradient
    \[
    B0 \leftarrow B
    B0[1] \leftarrow 0
    \]
    \[
    G \leftarrow t(X) \times (Y - C1) - 2L \times B0
    \]
    
    ## Check for convergence.
    \[
    \text{if (sqrt(sum(G^2)) < 1e-2) break}
    \]
    
    ## The Hessian
    \[
    A \leftarrow X \times \text{array}(C2, \text{dim}(X))
    \]
    \[
    H \leftarrow -t(X) \times A
    \]
    \[
    M \leftarrow \text{diag}(p)
    M[1,1] \leftarrow 0
    \]
    \[
    H \leftarrow H - 2L \times M
    \]
    
    ## Update the parameters.
    \[
    B \leftarrow B - \text{solve}(H, G)
    \]
}

B

## Select genes ranking above the qth quantile in terms of either
## variance or mean. This is an unsupervised selection so I1 and I2
## aren’t used.

\[
\text{GS1} \leftarrow \text{function}(M, I1, I2, q)
\]
{
    \[
    E \leftarrow \text{rowMeans}(M)
    \]
V <- apply(M, 1, var)

EQ <- quantile(E, q)
VQ <- quantile(V, q)

## This is the return value.
(E > EQ) | (V > VQ)
}

## Select genes ranking above the qth quantile in terms of Z-statistic
## between the two groups.
GS2 <- function(M, I1, I2, q)
{
    ## The group-wise means.
    E1 <- rowMeans(M[,I1])
    E2 <- rowMeans(M[,I2])

    ## The group-wise variances.
    V1 <- apply(M[,I1], 1, var)
    V2 <- apply(M[,I2], 1, var)

    ## The sample sizes in the two groups.
    n1 <- sum(I1)
    n2 <- sum(I2)

    ## The denominator of the Z-statistics.
    SD <- sqrt(V1/n1 + V2/n2)

    ## Calculate the Z-statistics, being careful about divisions by 0.
    Z <- ifelse(SD>1e-6, abs(E1-E2)/SD, 0)

    ## The cutpoint for inclusion in the selected set.
    QZ <- quantile(Z, q)

    ## This is the return value.
    Z > QZ
}

## Select genes ranking above the q1th quantile in terms of mean and
## variance, and also ranking above the q2th quantile in terms of
## Z-statistic between the two groups.
GS3 <- function(M, I1, I2, q1, q2)
{

J1 <- GS1(M, I1, I2, q1)
J2 <- GS2(M, I1, I2, q2)

## This is the return value.
J1 & J2
}

## Get the cross-validated error rate for penalized logistic
## regression using a given smoothing parameter L. Note that X should
## not include an intercept, and is stored with the gene expression
## levels in the rows and the samples across the columns.
plogistCV <- function(Y, X, L, q1, q2)
{
  ## The sample size.
  n <- dim(X)[2]

  ## The fitted values.
  F <- array(0,n)

  ## Cross-validation loop.
  for (i in (1:n))
  {
    ii <- setdiff((1:n), i)
    XX <- X[,ii]
    YY <- Y[ii]

    ## Select genes using the hold-out set.
    J <- GS3(XX, (YY==1), (YY==2), q1, q2)

    ## Include an intercept.
    X1 <- cbind(array(1,n-1), t(XX[J,]))

    B <- plogist(YY-1, X1, L)
    x <- c(1, X[J,i])
    F[i] <- (sum(B * x) > 0)*1
  }

  ## The number of correct predictions.
  sum(F == Y-1)
}

## Gene selection.
J <- GS1(M, Y==1, Y==2, 0.75)
## Get the number of correct predictions for supervised selection of ## the top 20%, 10%, ... of genes. This is addressing part of (ii).
for (ee in c(0.8, 0.9, 0.95, 0.96, 0.97, 0.98, 0.99))
{
    E <- plogistCV(Y, M[J,], L, 0, 0.95 + ee)
    print(E)
}

## Check the uncertainty for a penalty coefficient of 1000 by ## bootstrapping.
L <- 1000
for (r in (1:10))
{
    n <- length(J)
    ii <- ceiling(n*runif(n))
    X <- M[J[ii],]
    E <- plogistCV(Y, X, L, 0, 0.95)
    print(E)
}