```
#!/usr/bin/env perl -w
use Math::Complex;


#Perform Leave-five out analysis to determine two cutpoints on the following genes

for $gene (qw/ CEBPG E2F1 CAT ERCC4 ERCC5 GPX1 GPX3 GSTM3 GSTP1 GSTT1 GSTZ1 MGST1
SOD1 XRCC1 /) {


#Set the number of total gene expression measurements for each gene that will be
iterated through

if ($gene eq 'CEBPG' or $gene eq 'CAT' or $gene eq 'ERCC5' or $gene eq 'GPX1' or
$gene eq 'GPX3' or $gene eq 'GSTM3' or $gene eq 'GSTP1' or $gene eq 'GSTZ1' or $gene
eq 'MGST1' or $gene eq 'SOD1' or $gene eq 'XRCC1') {
   $individualcount = 49;  #number of gene expression measurements for these genes
} elsif ($gene eq 'E2F1') {
   $individualcount = 48;  #number of gene expression measurements for this gene
} elsif ($gene eq 'ERCC4') {
   $individualcount = 47;  #number of gene expression measurements for this gene
} elsif ($gene eq 'GSTT1') {
   $individualcount = 39;  #number of gene expression measurements for this gene
}

#Open the data file containing gene expression measurements and binary class 0 =
non-cancer and 1 = cancer
open GENEDATASET, "<$gene.csv";
my @genedataset = ();
while (my $line2 = <GENEDATASET>) {  #push the genedataset into an array
   chomp ($line2);
   my @tmp = split(",",$line2);
   push @genedataset, [@tmp];
}
close GENEDATASET;

#Open a file that contains a combinatorial list of all possible combinations of
leaving five out for the data set (created using combinatorial.pl algorithm)

open LEAVEFIVEOUT, "<combinatorial$individualcount-5";
open OUTPUT, ">$gene.txt";
while (my $line = <LEAVEFIVEOUT>) {  #iterate through this combinatorial file to
find which five to leave out during each iteration
   chomp ($line);  #Set or re-set all the variables that can possibly be used during
this iteration
   my @vallist = ();
   my @trainingset = ();
   my @validationset = ();
   my $cancersum = 0;
   my $noncancersum = 0;
   my $FP = 0.5*($individualcount-5);
   my $TP = 0.5*($individualcount-5);
   my $FN = 0;
   my $TN = 0;
   my @maxima = ();
   $maxima[0] = 0;
   $maxima[1] = -100;
   my @minima = ();
   $minima[0] = 0;
   $minima[1] = 100;
   $BCIerrormax = 0;
   $BCIerrormin = 0;
   $BCIerrormaxmin = 0;
   $NBCIerrormax = 0;
   $NBCIerrormin = 0;
   $NBCIerrormaxmin = 0;

   @vallist = split(",",$line);  # set the array for the five individuals (0-4) to
be stored for later validation, or to just be left out of cut-point analysis.

   for ($x = 1; $x <$individualcount + 1; $x++) { # split the genedataset into
training and validation/leave-out sets.
      if ($x == $vallist[0] or $x == $vallist[1] or $x == $vallist[2] or $x ==
$vallist[3] or $x == $vallist[4]) {
         $xlow = $x - 1;
         push @validationset, $genedataset[$xlow];
      } else {
         $xlow = $x - 1;
         push @trainingset, $genedataset[$xlow];
      }
   }

   for ($x = 0; $x <$#trainingset + 1; $x++) { # count the number of individuals
with cancer in the training set
      $cancersum = $cancersum + $trainingset[$x][1];
   }

   $noncancersum = $individualcount - $cancersum - 5; # infer the number of
individuals without cancer in the training set


   for ($x = 0; $x <$#trainingset + 1; $x++) { # create the simple moving average (n
= 5) and input it into the trainingset array
      my $individualsum = 0; #reset a variable
      my $numberofindividuals = 0; # reset a variable

      #Count the number of individuals to be averaged over.  Normally n = 5, but in
the extremes, cannot average n=5.
      if ($x < 2) {
         my $xlow = 0;
         my $xhigh = $x + 2;
         $numberofindividuals = $xhigh - $xlow;
         for ($y = $xlow; $y <$xhigh + 1; $y++) {
            $individualsum = $individualsum + $trainingset[$y][1];
         }
      } elsif ($x > $#trainingset - 2) {
         my $xlow = $x - 2;
         my $xhigh = $#trainingset;
         $numberofindividuals = $xhigh - $xlow;
         for ($y = $xlow; $y <$xhigh + 1; $y++) {
            $individualsum = $individualsum + $trainingset[$y][1];
         }
      } else {
         my $xlow = $x - 2;
         my $xhigh = $x + 2;
         $numberofindividuals = $xhigh - $xlow;
         for ($y = $xlow; $y <$xhigh + 1; $y++) {
            $individualsum = $individualsum + $trainingset[$y][1];
         }
      }

      $numberofindividuals = $numberofindividuals +1;

      # for each gene expression value in the ordered array, determine the
normalized sum of cancer and non-cancer individuals.  Normalized on a fractional
scale of 0 to 1 for both.
```

```
      $BCIsum = 
(($individualsum)/($numberofindividuals-$individualsum+($individualsum*($noncancersu
m/$cancersum))))*($noncancersum/$cancersum);
      $NBCIsum = 1 - $BCIsum;

      # input the simple moving average values for each individual into the array
      $trainingset[$x][2] = $BCIsum;
      $trainingset[$x][3] = $NBCIsum;
      # determine ROC data points
      $trainingset[$x][4] = $TP;
      $trainingset[$x][6] = $FP;
      $TN = 0.5*($individualcount-5) - $FP;
      $FN = 0.5*($individualcount-5) - $TP;
      $trainingset[$x][5] = $TN;
      $trainingset[$x][7] = $FN;
      $trainingset[$x][8] = $TP/($TP+$FN); # Sensitivity or true positive rate
      $trainingset[$x][9] = $FP/($FP+$TN); # 1 - Specificity or false positive rate
      $trainingset[$x][10] = $trainingset[$x][8] - $trainingset[$x][9]; # TPR - FPR

#      print "$TP $FP\n";
#      print "$trainingset[$x][0] and $trainingset[$x][10]\n";

#Looking for maxima and minima in TPR - FPR inflection points
      # determine which gene expression value represents the upper most cut-point 
based on the maximal distance between TPR - FPR
      if ($trainingset[$x][10] > $maxima[1]) {
         $maxima[0] = $trainingset[$x][0];
         $maxima[1] = $trainingset[$x][10];
      }
      # determine which gene expression value represents the lower most cut-point 
based on the maximal distance between TPR - FPR
      if ($trainingset[$x][10] < $minima[1]) {
         $minima[0] = $trainingset[$x][0];
         $minima[1] = $trainingset[$x][10];
      }
      $TP = $TP - $BCIsum;
      $FP = $FP - $NBCIsum;
   }


#print "$gene -- $individualcount\n";

#if the upper cut-point gene expression value is greater than the lower cut-point 
gene expression value than accept the pair of values and put into a list of possible 
cut-points for that gene
   if ($maxima[0] > $minima[0]) {
      print "$line --  $maxima[0] and $minima[0] \n";
      print OUTPUT "$maxima[0],$minima[0]\n";
   }

#   print "$total\n";

#   print "BCI and NBCI error - - - - $line \n";
#   print "Minima: $minima[0], $BCIerrormin and $NBCIerrormin\n";
#   print "Maxima: $maxima[0], $BCIerrormax and $NBCIerrormax\n";
#   print "Max/Min: $maxima[0] & $minima[0], $BCIerrormaxmin and 
$NBCIerrormaxmin\n\n";

}
close LEAVEFIVEOUT

}
```

```
# The data generated from this file can be somewhat large... feed it into the
cutpointcondenser.pl algorithm to generate histogram data for each gene.
```