Q5 Software Project Application Programming Interface
DRAFT – 12/19/2005

Table of Contents:

q5 3
mzXML 4
normalize 5
pcaProject 6
pca 7

q5

Purpose *–Provides the main function for the script. Trains a disciminant on given training data and then assesses its predictive capacity with testing data. This is the function of the Q5 algorithm. Each of the functions it performs are accomplished by calling functions that are passed as parameters, this allows users to substitute their own functions for any of the steps in Q5 (with the exception of the statistical parameters generated).*
Type *- This is the main function that implements the Q5 algorithm.*

Functional Description *–Performs the Q5 algorithm in a flexible fashion that allows the user to alter its sub-functionality.*

Subordinate *–Four functions passed by the user. Provides default values that depend on pca, lda, pcaProject, and MASS:::predict.lda.*

Dependencies - *Requires the MASS package.*

Preconditions –*Requires two matrices of data**,** one for training, one for testing, requires two vectors that describe the classes of the data. Optionally takes four functions for dimensional reduction, discriminant construction, data projection, and class prediction. Optionally takes one integer that determines the number of eigenvectors to remove. Optionally takes two filenames, one to output the statistics and one to write a plot of the eigenvalues as a png graphic.*

Interfaces - *q5(trainData, trainClasses, testData, testClasses, reductionFunction=pca, eigenVectorsToRemove=4, discriminantFunction=lda, projectionFunction=pcaProject, predictionFunction=MASS:::predict.lda, statfile=NULL, plotfile=NULL)*

Takes two matrices of real numbers, one of training data, one of testing data. Takes two vectors of integers the label the class of each row in the two matrices, 1 indicates diseased state. Returns the generated discriminant, the estimated positive predictive value of the discriminant, the percentage of cases in the testing data that were correctly classified, the estimated sensitivity and specificity.

Graphics - *Generates a plot of the eigenvalues. If plotfile is NULL, the plot is shown on the screen. If plotfile is the name of a file the plot is placed in that file as a png graphic.*

Resources - *None*

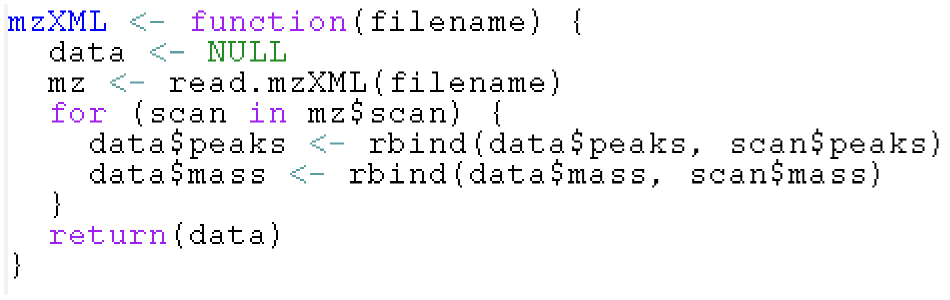
Error Conditions –*The function will fail with an error message if any of the passed objects are not of the correct type.*

Example Implementation -

```r
q5 <- function(trainData, trainClasses, testData, testClasses,
               reductionFunction=pca, eigenVectorsToRemove=4,
               discriminantFunction=lda,
               projectionFunction=pcaProject,
               predictionFunction=MASS:::predict.lda,
               statfile=NULL,
               plotfile=NULL) {
  # for trainClasses and testClasses, 1 indicates disease.
  pcaRes <- reductionFunction(trainData, eigenVectorsToRemove)
  trainData <- projectionFunction(trainData, pcaRes$means, pcaRes$vecs)
  discriminant <- discriminantFunction(trainData, trainClasses)

  testData <- projectionFunction(testData, pcaRes$means, pcaRes$vecs)
  results <- predictionFunction(discriminant, testData)

  truePos <- length(intersect(which(results$class == 1), which(testClasses == 1)))
  falsePos <- length(intersect(which(results$class == 1), which(testClasses != 1)))

  trueNeg <- length(intersect(which(results$class != 1), which(testClasses != 1)))
  falseNeg <- length(intersect(which(results$class != 1), which(testClasses == 1)))

  positivePredictiveValue <- truePos / (truePos + falsePos)
  percentageCorrectlyClassified <- (truePos + trueNeg) / (truePos + falsePos + trueNeg + falseNeg)
  sensitivity <- truePos / (truePos + falseNeg)
  specificity <- trueNeg / (falsePos + trueNeg)

  # write the statistics file
  sink(statfile)
  cat('positivePredictiveValue:\t')
  cat(positivePredictiveValue)
  cat('\n')
  cat('percentageCorrectlyClassified:\t')
  cat(percentageCorrectlyClassified)
  cat('\n')
  cat('percentageCorrectlyClassified:\t')
  cat(percentageCorrectlyClassified)
  cat('\n')
  cat('sensitivity:\t')
  cat(sensitivity)
  cat('\n')
  cat('specificity:\t')
  cat(specificity)
  cat('\n\n')
  sink()

  # plot the graph of the eigenvalues
  if (!is.null(plotfile)) {
    png(file=plotfile)
  }

  plot(pcaRes$vals)
  dev.off()

  return(list(discriminant=discriminant,
              positivePredictiveValue=positivePredictiveValue,
              percentageCorrectlyClassified=percentageCorrectlyClassified,
              sensitivity=sensitivity, specificity=specificity))
}
```

mzXML

- ***Purpose*** *- The mzXML function loads a datafile from a user supplied string containing the path to an mxXML file. It returns a list of two values; "mass", the mass to charge ratio of each peak in th escan and "peaks" which are the intensities for each peak in the scan.*
- ***Type*** *- This is a conveneince function within the Q5 script for the user's use.*
- ***Functional Description*** *- Reads each scan in the file and appends its peaks and mass to charge ratios to separate items in a list.*
- ***Subordinate*** *- This function makes external calls to function read.mzXML in the caMassClass.*
- ***Dependencies*** *- Requires the caMassClass package.*
- ***Preconditions*** *- Requires a string describing the location of a local file in mzXML format. This function is called prior to executing q5.*
- ***Interfaces*** *- mzXML(filename)*
- ***Graphics*** *- This is a non-graphical routine.*
- ***Resources*** *- None*
- ***Error Conditions*** *- The function will fail with an error message on malformed mzXML. The function will also provide an error message if the user supplied string does not contain the path to a local mzXML file.*
- ***Example Implementation –***



```
mzXML <- function(filename) {
  data <- NULL
  mz <- read.mzXML(filename)
  for (scan in mz$scan) {
    data$peaks <- rbind(data$peaks, scan$peaks)
    data$mass <- rbind(data$mass, scan$mass)
  }
  return(data)
}
```

normalize

- ***Purpose** – Normalizes the peak intensity matrix. This is not part of the main script but it provides a simple normalization for the convenience of users.*
- ***Type** - This is a function that can be called before q5 to normalize the data.*
- ***Functional Description** - The normalize function normalizes the peak intensities of a scan to the range [0, 1] using a standard linear transformation.*
- ***Subordinate** - This function makes no function calls to dependency packages.*
- ***Dependencies** – None.*
- ***Preconditions** - Requires a two dimensional matrix of real values*
- ***Interfaces** – normalize(mat)*
- ***Graphics** - None.*
- ***Resources** – None.*
- ***Error Conditions** – This function will fail with an error message if it doesn't receive a matrix of rank 2.*
- ***Example Implementation** -*

```
normalize <- function(mat) {
  for (ixRow in seq(nrow(mat))) {
    rowMin <- min(mat[ixRow,])
    rowRange <- max(mat[ixRow,]) - rowMin
    mat[ixRow,] <- (mat[ixRow,] - rowMin) / rowRange
  }
  return(mat)
}
```

-

pcaProject

- ***Purpose*** *– Projects data into the pca space on which the discriminant was defined.*
- ***Type*** *- This is a function that can be passed to q5 as the projectionFunction parameter.*
- ***Functional Description*** *– Subtracts the provided column means from the given matrix, multiplies the user supplied matrix by the user supplied eigenvectors. Thus it projects a matrix into the subspace of the principle components.*
- ***Subordinate*** *– This function makes no function calls to dependency packages.*
- ***Dependencies*** *– None.*
- ***Preconditions*** *- Requires two two dimensional matrices of real values and a vector.*
- ***Interfaces*** *– pcaProject(mat, means, vecs)*
- ***Graphics*** *- None.*
- ***Resources*** *– None.*
- ***Error Conditions*** *– This function will fail with an error message if it receives the wrong type of input. The size of the second dimension of the matrix (mat) must be equal to the size of the first dimension of the second matrix (eigvecs).*
- ***Example Implementation –***

```
pcaProject <- function(mat, means, vecs){
  # project into the pca space
  for (ix in seq(ncol(mat))) {
    mat[,ix] <- mat[,ix] - means[ix]
  }
  mat <- mat %*% vecs
  return(mat)
}
```

pca

- The pca function performs principle components analysis on the given matrix.
- ***Type*** *- This is a function within the Q5 script with an exposed development interface.*
- ***Purpose*** *– Dimensional reduction of the data.*
- ***Function*** *– Performs principle components analysis.*
- ***Subordinate*** *– None.*
- ***Dependencies*** *– Takes a two dimensional matrix of real values and an integer.*
- ***Preconditions*** *- Requires a two dimensional matrix of real values and an integer.*
- ***Interfaces*** *– Calculates the means of each column of the matrix. Performs principle components analysis. Removes some number of the least significant principle components (defaults to four). Returns the column means, the eigenvectors and the eigenvalues.*
- ***Graphics*** *- None.*
- ***Resources*** *– None.*
- ***Error Conditions*** *– This function will fail if it receives the wrong type of input with an error message.*
- ***Example Implementation –***

```
pca <- function(mat, eigenVectorsToRemove=4) {
  means <- rep(-1, ncol(mat))
  for (ix in seq(ncol(mat))) {
    means[ix] <- mean(mat[,ix])
    mat[,ix] <- mat[,ix] - means[ix]
  }
  res <- prcomp(mat)
  vals <- res$sdev**2
  vecs <- res$rotation
  x <- sort(vals, index.return=TRUE, decreasing=TRUE)
  vals <- vals[x$ix[1:(length(x$ix) - eigenVectorsToRemove)]]
  vecs <- vecs[,x$ix[1:(length(x$ix) - eigenVectorsToRemove)]]
  return(list(means=means, vecs=vecs, vals=vals))
}
```