Post-Genomic Bioinformatics: Assignments 1 and 2

Contents

Exercise 1: Exploratory Analysis of Transcriptomics Data

Exercise 2: Application of Machine Learning to Metabolomics Data for Disease Diagnosis

Exercise 1: Exploratory Analysis of Transcriptomics Data

Introduction

I analysed a microarray dataset for three patient groups who had contracted the Dengue virus, and a healthy control group, in order to compare the groups' gene expression profiles, and to find the most significantly differentially expressed genes between groups (data accessible at NCBI GEO database (Kwissa et al, 2014), accession GDS5093).

The sampling sub-groups are as follows: haemorrhagic fever (n=10, sampled 2-9 days after onset of symptoms), fever (n=18, sampled 2-9 days after onset of symptoms), convalescent (n=19, sampled at least four weeks after discharge) and healthy controls (n=9). Patients' disease status was confirmed by RT-PCR and serological analysis. RNA was extracted from whole blood that had been stored in RNALater preservative (Ambion) and globin-depleted. Gene expression levels were normalised using Robust Multiarray Averaging (RMA) and transformed (method not published).

Methods

All data analysis was performed using R (version 3.6.1) in RStudio (version 1.2.5019).

I downloaded the data and extracted the data matrix as "X" using the GEOquery package (version 2.54.1). During this step the data was log-transformed to reduce skew.

```
# Load required packages
library(GEOquery)
library(dendextend)
library(gplots)
# Download data
ged <- getGEO("GDS5093", GSEMatrix = TRUE)
# Extract ExpressionSet object
eset <- GDS2eSet(deg, do.log2=TRUE)
# Extract data matrix
X <- exprs(eset)
```

The read count distributions for each sample were checked for scaling and normality using a boxplot (see figure 1).

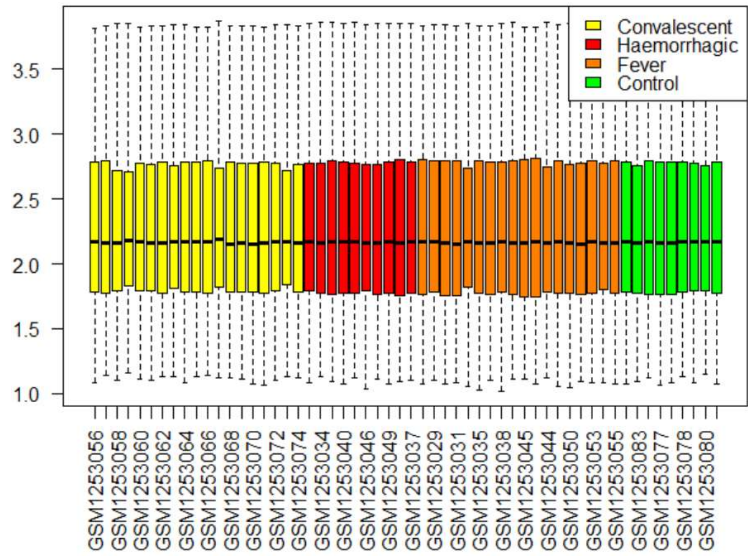


Figure 1. mRNA abundance distribution for each sample.

```
# Plot
par(mar=c(10,4.1,4.1,2.1))
boxplot(X,las=2,col=colour)
legend(x="topright",fill=c("yellow","red","darkorangel","gree
n"),legend=c("Convalescent","Haemorrhagic","Fever","Control")
)
```

For code used to assign colours, see appendix 1.

I then performed two types of unsupervised machine learning - hierarchical cluster analysis (HCA) and principal component analysis (PCA) to see whether the patient groups clustered together.

HCA

HCA was performed on data matrix X using the dendextend package (version 1.13.2). Every available combination of distance algorithm and linkage algorithm was attempted in order to find the best combinations for clustering the samples. The best linkage algorithms were "ward.D" and "ward.D2", and the best distance algorithms were Euclidean, Manhattan and Minkowski (see figure 2). An example of the code used to perform one HCA is included below. The code used to test all combinations is in appendix 1.

```
# Perform HCA
test007 <- hclust(dist(t(X),method="euclidean"),method="ward.D")
```

PCA

PCA was performed on data matrix X using the singular value decomposition (SVD) method to calculate the genes' loading scores for each principal component (PC). A 2D PC plot was generated to visualise the sample clusters (see figure 3a). The explained variance, and cumulative variance, for each PC was then calculated and plotted (see figures 3b and 3c).

```
# Perform PCA using SVD
Xpca <- prcomp(t(X),scale=TRUE)
```

In order to identify the most significantly differentially expressed genes between the four groups, the *p*-values were calculated using an ANOVA test.

```
# Extract gene names and assign to data matrix
Y <- Table(ged)
geneNames <- as.character(Y$IDENTIFIER)
rownames(X) <- geneNames
# Extract phenotypic data
pDat <- pData(eset)
# Add class row to data matrix
classvector <- pDat$disease.state
```

```
Xclass <- rbind(classvector,X)
# Perform ANOVA and extract p-values
pValues <- c()
for (i in (2:54716)){
  anova1 <- lm(Xclass[i,]~as.character(Xclass[1,]))
  anova2 <- anova(anova1)
  pValues <- c(pValues, anova2$`Pr(>F)`[1])
}
```

The original data matrix X was then subsetted for the 106 most significantly differentially expressed genes, to give matrix “Xsigonly”.

```
# Extract most significant genes
topP <- which((pValues<0.000000000000000001)==TRUE)
# Subset original matrix based on top 106 most significant genes
Xsigonly <- X[topP,]
# Check for correctness
geneNames[topP]==rownames(Xsigonly)
```

PCA was then performed again on Xsigonly. As before, plots to show clustering, explained variance and cumulative variance were produced (see figure 4).

```
# Repeat PCA
Rptpca <- prcomp(t(Xsigonly),scale=TRUE)
# Extract information including explained variance
Rptsumm <- summary(Rptpca)
```

A heatmap with dendrograms was also plotted for Xsigonly using gplots (version 3.0.1.1) to visualise not only how well the samples clustered based on expression levels of the top 106 genes, but also the up-regulation and down-regulation of the genes, and the similarity of the genes’ expression levels (see figure 5).

Results and discussion

Despite being classified into one of four groups according to disease status, both HCA (figure 2) and PCA (figures 3 and 4) showed that, in terms of gene expression levels, the patients fell into two groups: “fever or haemorrhagic fever” and “convalescent or healthy control”.

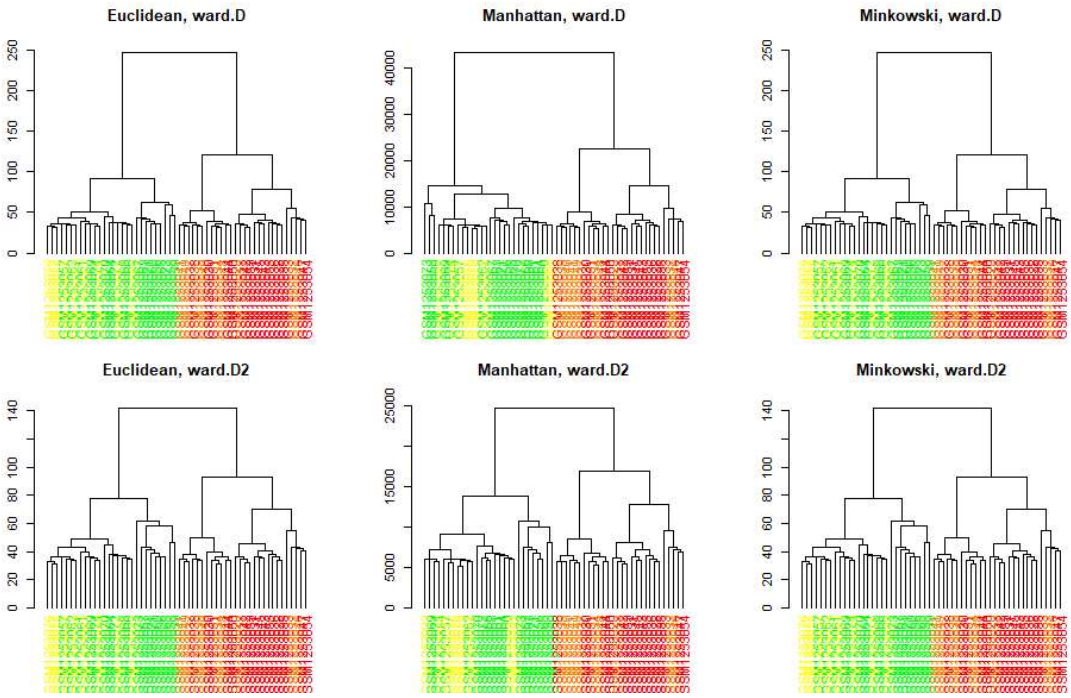
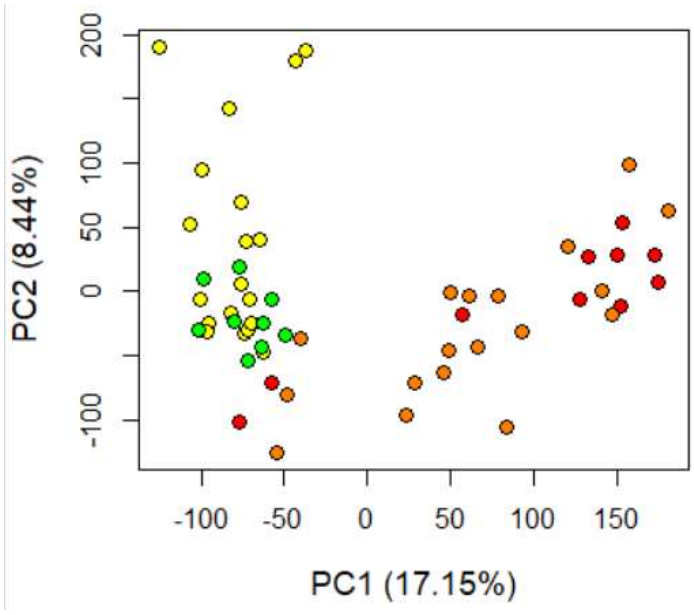


Figure 2. Hierarchical clustering dendrograms showing clustering of patients infected with the Dengue virus according to similarity of gene expression profile (red = “haemorrhagic fever”, orange = “fever”, yellow = “convalescent”, green = “healthy control”). The six best combinations of distance and linkage algorithms are shown. An example of the code used to plot one dendrogram is included below. The code used to test all combinations is in appendix 1.

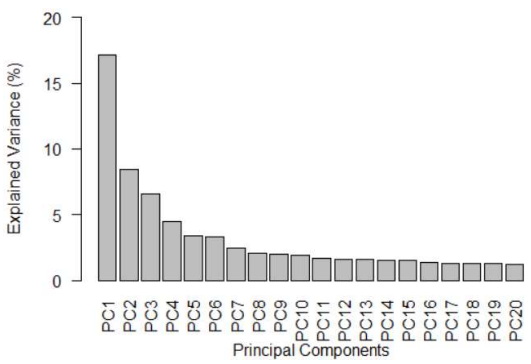
```
# Create dendrogram
dend007 <- as.dendrogram(test007)
# Create colour vector
colourCodes <- c(Control="green",Haemorrhagic="red",Fever="darkorange1",Convalescent="yellow")
# Plot dendrogram
par(mfrow=c(2,3))
labels_colors(dend007) <- colourCodes[pClass][order.dendrogram(dend007)]
plot(dend007, main="Euclidean, ward.D")
```

Although HCA and PCA yielded similar results, the cumulative variance for PC1 and PC2 was low at just 25.59% (see figure 3). Therefore, this PCA plot is of limited usefulness when describing trends in this dataset.

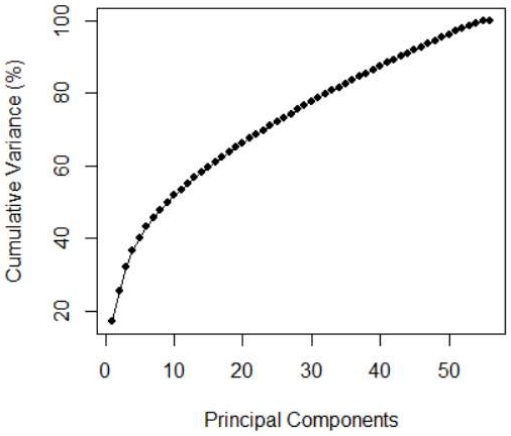
The second PCA, performed using only the top 106 most significantly differentially expressed genes, gives a much more useful 2D plot, as the cumulative variance for PC1 and PC2 is 90.85% (see figure 4). Here, we can see that the same two pairs of groups still cluster together, but that a boundary with a much larger margin could be drawn between the two clusters here, compared to the first PC plot.



a)



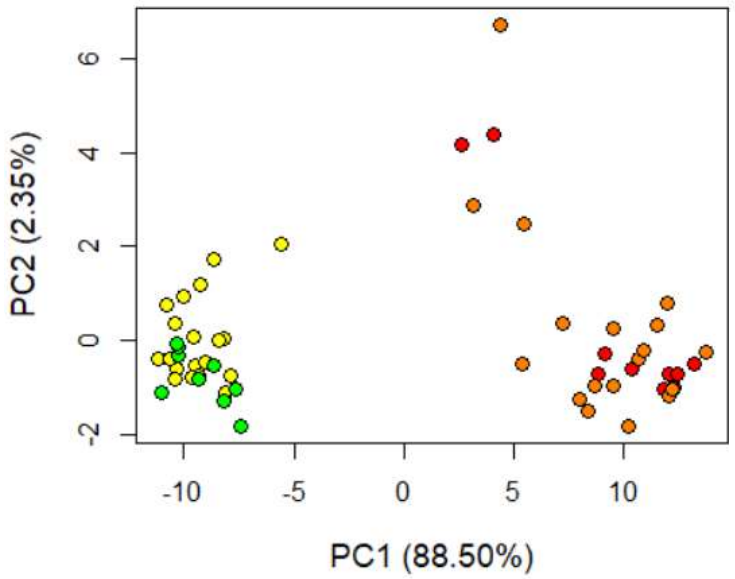
b)



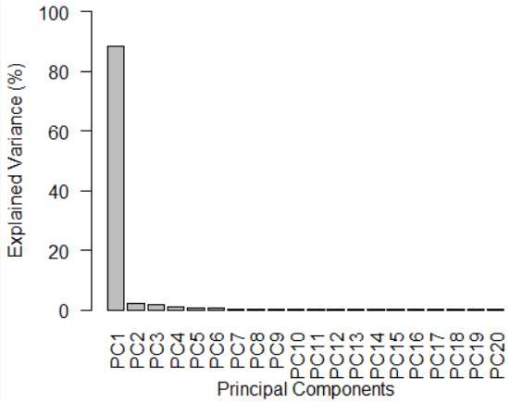
c)

Figure 3. Results of PCA using all genes in data matrix X.
a) 2D principal component plot showing clustering of patients infected with the Dengue virus according to similarity of gene expression profile (red = “haemorrhagic fever”, orange = “fever”, yellow = “convalescent”, green = “healthy control”).
b) Explained variance and c) cumulative variance for each principal component. These are quite low for the earliest PCs and thus the PCA is of limited usefulness. The code used to generate the plots is shown below.

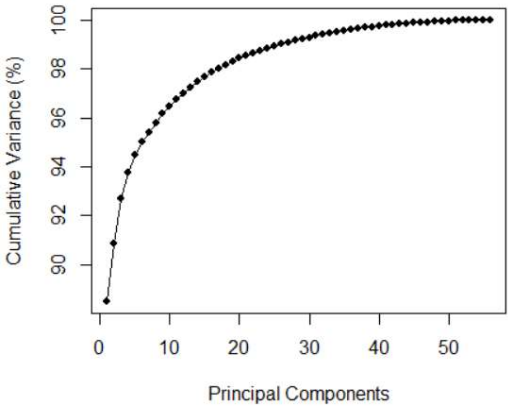
```
# Extract information including explained variance
summ <- summary(Xpca)
# PCA plot
Xscores <- Xpca$x
plot(Xscores, xlab="PC1 (17.15%)", ylab="PC2 (8.44%)", pch=21,
     bg=colour, cex=1.2, cex.lab=1.2, cex.axis=1)
# Make vector of explained variances for each PC
expVar <- summ$importance[2,]*100
# Plot
barplot(expVar[1:20], xlab="Principal Components",
        ylab="Explained Variance (%)",las=2,
        ylim=c(0,20))
# Make vector of cumulative variances for each PC
cumVar <- summ$importance[3,]*100
# Plot
plot(cumVar, type="o", col="black", pch=21, bg="black",
     cex=0.8, xlab="Principal Components",
     ylab="Cumulative Variance (%)")
```



a)



b)



c)

Figure 4. Results of PCA using top 106 most differentially expressed genes (data matrix Xsigonly).
a) 2D principal component plot showing clustering of patients infected with the Dengue virus according to similarity of gene expression profile (red = “haemorrhagic fever”, orange = “fever”, yellow = “convalescent”, green = “healthy control”).
b) Explained variance and c) cumulative variance for each principal component. These are much higher for the earliest PCs than are the equivalents in the first PCA. The code used to generate the plots is shown below.

```
# Look at PCA plot
Rptscores <- Rptpca$x
plot(Rptscores, xlab="PC1 (88.50%)", ylab="PC2 (2.35%)",
     pch=21,bg=colour, cex=1.2, cex.lab=1.2, cex.axis=1)
# Make vector of explained variances for each PC
RptexpVar <- Rptsumm$importance[2,]*100
# Plot
barplot(RptexpVar[1:20], xlab="Principal Components",
        ylab="Explained Variance (%)",las=2,
        ylim=c(0,100))
# Make vector of cumulative variance for each PC
RptcumVar <- Rptsumm$importance[3,]*100
# Plot
plot(RptcumVar, type="o", col="black", pch=21, bg="black",
     cex=0.8, xlab="Principal Components",
     ylab="Cumulative Variance (%)")
```

The heatmap in figure 5 was generated using Xsigonly, i.e. only the top 106 most significantly differentially expressed genes. It shows that there are clustered gene sets (left hand dendrogram) that are similarly up-regulated and down-regulated depending on whether the patient is currently unwell (fever/haemorrhagic fever groups) or not (convalescent/control groups). The top dendrogram supports the finding in HCA and PCA that the samples cluster into two groups: “fever or haemorrhagic fever” and “convalescent or healthy control”.

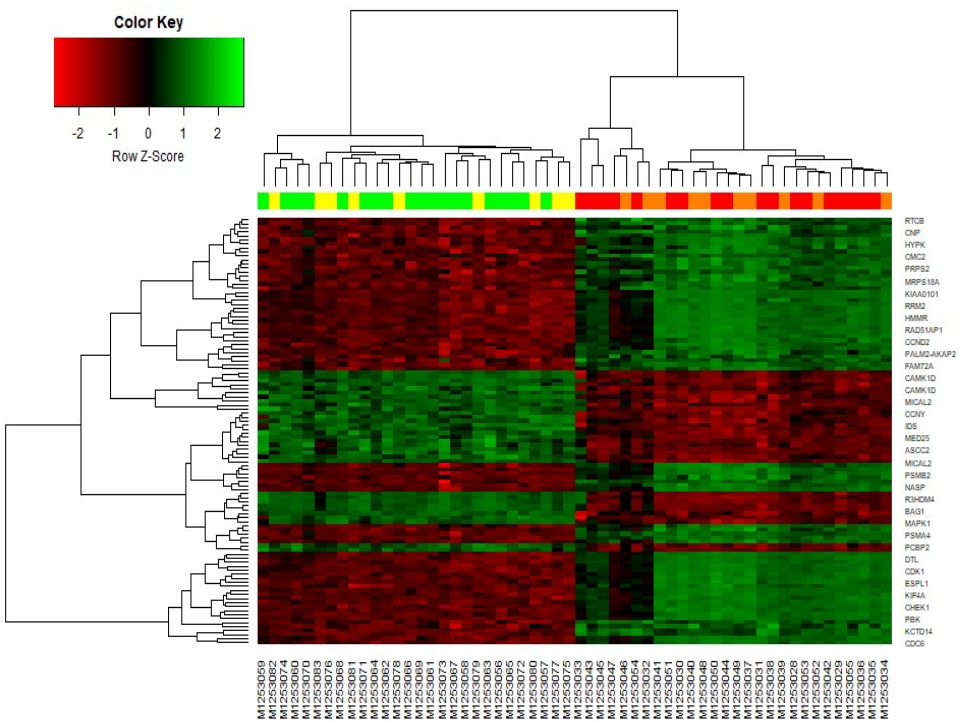


Figure 5. Heatmap to show up-regulation and down-regulation in the top 106 most significantly differentially expressed genes (genes form the vertical axis; samples form the horizontal axis). The colour bar displayed between the top dendrogram and the heatmap represents the sample groups (green = controls, yellow = convalescent, orange = fever, red = haemorrhagic fever).

```
# Plot a heatmap for the top 106 genes
heatmap.2(Xsigonly,col=redgreen(75),scale="row",key=T,
          keysize=1.5,density.info="none",trace="none",
          cexCol=0.9,
          ColSideColors=colourCodes[pClass])
```

Conclusion

The patients sampled in this study could be divided, according to their gene expression profiles, into two groups: “currently unwell with the Dengue virus” and “not currently unwell with the Dengue virus”. This is not very surprising, given that the tissue of origin was whole blood and thus the RNA profiles are likely to be reflective of immune activity in response to the virus. However, unsupervised machine learning alone can only provide a limited amount of information. Gene set enrichment analysis could provide more insight into the processes that are up-regulated and down-regulated in these two states.

Exercise 2: Application of Machine Learning to Metabolomics Data for Disease Diagnosis

Introduction

The aim of this study was to analyse GC-MS metabolomics data from four different sample types (blood; n=57, breath; n=60, faeces; n=45 and urine; n=36) and rank them in order of usefulness in diagnosing Crohn's disease (CD). The samples were from four participant groups: CD, ulcerative colitis (UC), irritable bowel syndrome (IBS) and healthy controls. The classyfire R package was to be used to perform supervised machine learning using support vector machines (SVM).

It is important to perform a differential diagnosis for patients with undiagnosed chronic gastrointestinal disorders in order to ensure the correct treatment is given (Tontini et al, 2015), hence the datasets chosen for analysis contained all four patient groups and a four-level class vector was used. However, this study will focus on measuring the accuracy of CD diagnosis specifically.

Methods

All data analysis was performed using R (version 3.6.1) in RStudio (version 1.2.5019). Additional packages used: classyfire (version 0.1-2), R.matlab (version 3.6.2) and caret (version 6.0-64). For the full R script, see appendix 2.

I imported the each of the raw .mat data files files using readMat() and created data matrices for each. The class vector was binary (CD vs non-CD) and the ratios were not suitable for machine learning (see table 1) so I created a multiclass vector in order to balance the ratios of each class (see table 2). Furthermore, a diagnostic tool that could distinguish between different chronic gastrointestinal disorders would be of more clinical utility than one that could not.

	Number of samples		
Raw data matrix name	CD	Non-CD	Total
Xblood	14	43	57
Xbreath	17	43	60
Xfaecal	11	34	45
Xurine	7	29	36

Table 1. Number of samples per class using original binary class vector.

	Number of samples				
Raw data matrix name	CD	Control	IBS	UC	Total
Xblood	14	18	14	11	57
Xbreath	17	19	14	10	60
Xfaecal	11	13	11	10	45
Xurine	7	14	8	7	36

Table 2. Number of samples per class using new multiclass vector.

The data was then scaled to make the distributions more normal, and to make the ranges similar across each variable, to avoid assigning undue importance to variables with a larger range.

Principal component analysis (PCA) using singular value decomposition (SVD) was performed for the purpose of dimensionality reduction, as the original data matrices contained 4,349 variables.

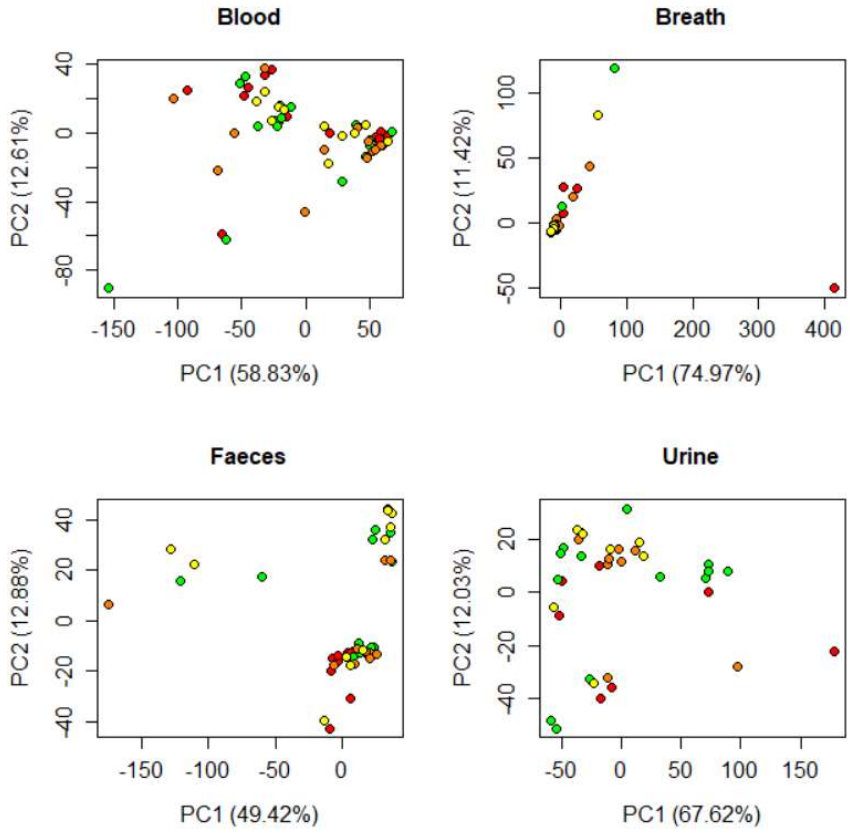


Figure 1. 2D principal component (PC) plots for each sample type. The different patient groups do not form distinct clusters in 2D space. Red = CD, orange = IBS, yellow = UC, green = controls.

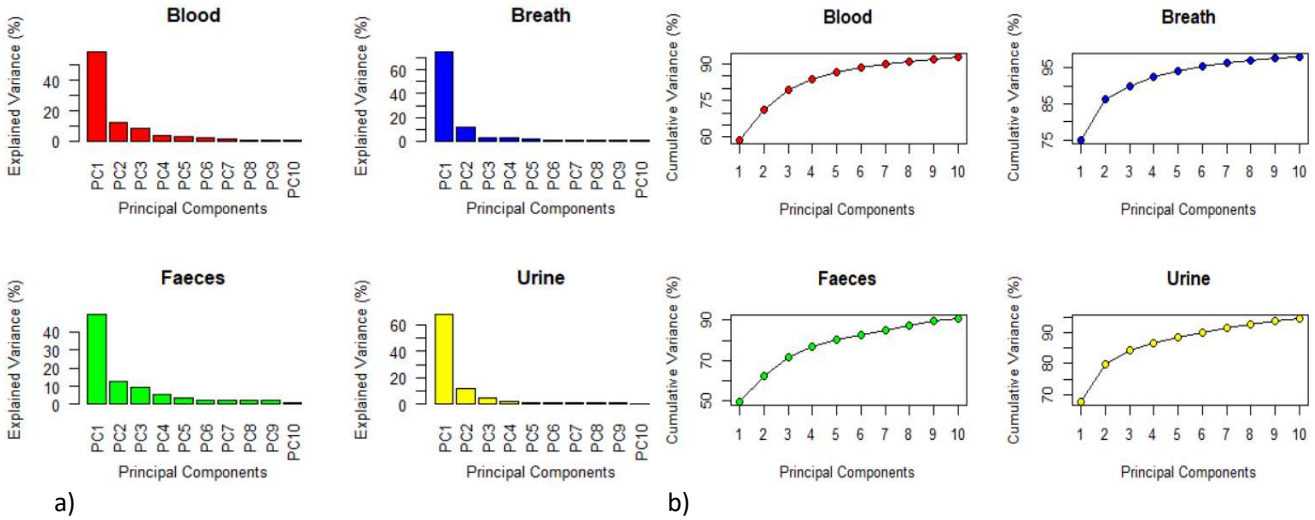


Figure 2. a) Explained variance and b) cumulative variance for the first ten PCs for each sample type

The data matrices produced by the PCA were subsetted to include enough PCs to explain at least 90% of the variance.

Low dimensional data matrix name	Number of PCs	Explained variance (%)
Xblood2	8	91.00
Xbreath2	4	92.39
Xfaecal2	10	90.66
Xurine2	6	90.09

Table 3. Low dimensional data matrices used for classification ensemble generation, and the number of PCs and explained variance, therein.

A classification ensemble was then generated for each sample type using the cfBuild() function from the classyfire package, which uses SVMs. 30 bootstraps and 30 ensembles were performed for each. The optimal number of ensembles was explored *post-hoc* using the ggEnsTrend() function. The performance levelled out after around 6 ensembles so no new SVM ensembles were attempted (see figure 3).

The sensitivity and specificity in diagnosing CD was calculated for each ensemble using the confusionMatrix() function from the caret package on the object returned by the getConfMatr() function from the classyfire package.

The ensembles' performance was assessed by cross-validation. Permutation testing was carried out on each of the low-dimensional data matrices using the cfPermute() function. This function randomly shuffles the class vector so that many samples will be assigned the incorrect disease, and then trains the SVMs based on this matrix. If ensembles produced using cfBuild() and cfPermute() performed equally well on the same data, this would show that any correct classification was due to chance. Five permutations were performed for each ensemble.

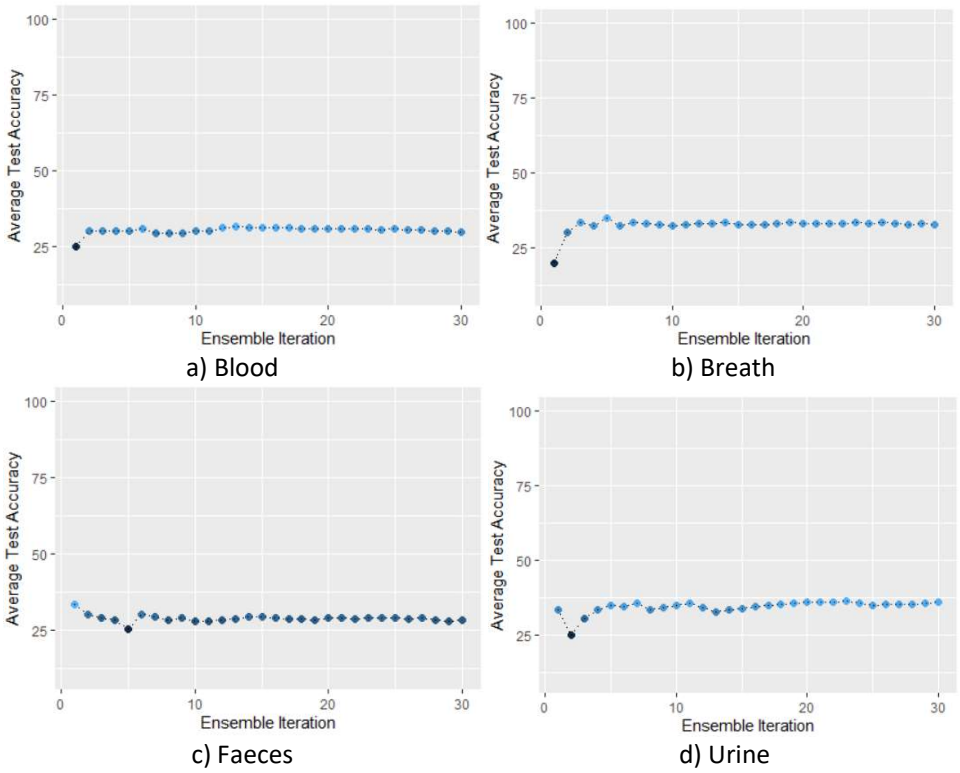


Figure 3. Average test accuracy vs number of ensembles for each sample type.

Results and Discussion

The best sample type for diagnosing CD was faeces, according to both specificity and sensitivity (see table 4).

Sample	Specificity (%)	Rank
Faecal	82.2	1
Breath	79.2	2
Blood	78.8	3
Urine	71.0	4

a)

Sample	Sensitivity (%)	Rank
Faecal	43.4	1
Breath	34.8	2=
Blood	34.8	2=
Urine	4.5	4

b)

Table 4. Specificity and sensitivity for each sample type in diagnosing CD using an SVM ensemble with 30 bootstraps and 30 ensembles.

The mean test accuracy of the blood, breath and faeces ensembles were slightly higher than the average mean test accuracy of the permutation set (see table 5 and figure 4). The accuracies ranged from 28.46 - 35.84%, which is poor. The urine data set gave the best ensemble in terms of overall accuracy, although the permutation test gave a higher mean accuracy than the original ensemble. The faeces data set, which gave the best specificity and sensitivity for diagnosing CD, ranked third by overall accuracy.

Sample type	Mean test accuracy of ensemble (%)	Average mean test accuracies in permutation testing (%)
Blood	29.83	27.70
Breath	32.83	29.93
Faeces	28.46	24.98
Urine	35.84	39.56

Table 5. Mean test accuracy of each original classifier ensemble and average mean test accuracy during permutation testing.
For each ensemble, bootNum = 30, ens = 30.
For each permutation ensemble, perm = 5.

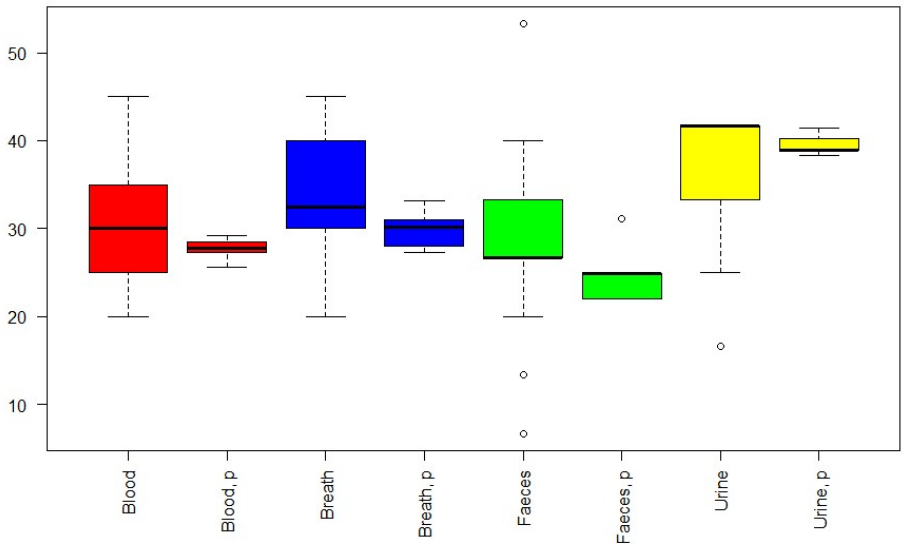


Figure 4. Test accuracy of each original ensemble (e.g. “Blood”) is shown alongside the average accuracy (%) of each permutation ensemble (e.g. “Blood, p”). For each ensemble, bootNum = 30, ens = 30. For each permutation ensemble, perm = 5.

Conclusion

This study did not produce an ensemble that was sensitive enough to be used to diagnose CD from GC-MS metabolomics data in a clinical setting, or accurate enough to be used to differentially diagnose patients with one of several gastrointestinal diseases. However, of all four of the sample types tested, faeces was the best in terms of specificity and sensitivity. This isn't surprising given that the sample has been produced by, and in contact with, the diseased organ. In order to improve upon the ensemble, a greater number of samples could be collected, enough PCs to account for a higher cumulative variance could be used, and a higher number of bootstraps could be attempted in future.

References

Barrett, T., Wilhite, S.E., Ledoux, P., Evangelista, C., Kim, I.F., Tomashevsky, M., Marshall, K.A., Phillippy, K.H., Sherman, P.M., Holko, M., Yefanov, A., Lee, H., Zhang, N., Robertson, C.L., Serova, N., Davis, S., Soboleva, A. (2013). NCBI GEO: archive for functional genomics data sets--update. *Nucleic Acids Res.* 41(Database issue):D991-5. doi: 10.1093/nar/gks1193.

Bengtsson, H. (2018). *R.matlab: Read and Write MAT Files and Call MATLAB from Within R. R package version 3.6.2.* CRAN, viewed 13 December 2019, <https://CRAN.R-project.org/package=R.matlab>

Chatzimichali, E. and Bessant, C. (2015). *classyfire: Robust multivariate classification using highly optimised SVM ensembles. R package version 0.1-2.* CRAN, viewed 13 December 2019, <https://CRAN.R-project.org/src/contrib/Archive/classyfire>

Davis, S. and Meltzer, P.S. (2007). GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. *Bioinformatics*. 15;23(14) 1846-1847. doi: 10.1093/bioinformatics/btm254.

Galili, T. (2015). dendextend: an R package for visualizing, adjusting and comparing trees of hierarchical clustering. *Bioinformatics* 15;31(22):37180-20. doi: 10.1093/bioinformatics/btv428.

Kuhn, M. (2019). *caret: Classification and Regression Training. R package version 6.0-84.* CRAN, viewed 13 December 2019, <https://CRAN.R-project.org/package=caret>

Kwissa, M., Nakaya, H.I., Onlamoon, N., Wrammert, J., Villinger, F., Perng, G.C., Yoksan, S., Pattanapanyasat, K., Chokephaibulkit, K., Ahmed, R., Pulendran, B. (2014). Dengue virus infection induces expansion of a CD14(+)CD16(+) monocyte population that stimulates plasmablast differentiation. *Cell Host Microbe* 9;16(1):115-27. doi: 10.1016/j.chom.2014.06.001.

R Core Team (2019). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria, viewed 12 December 2019, <https://www.R-project.org/>

RStudio Team (2019). *RStudio: Integrated Development for R*. RStudio, Inc., Boston, MA, viewed 12 December 2019, <http://www.rstudio.com/>

Tontini, G.E., Vecchi, M., Pastorelli, L., Neurath, M.F., Neumann, H. (2015) Differential diagnosis in inflammatory bowel disease colitis: State of the art and future perspectives. *World J Gastroenterol.* 21(1): 21-46. doi: 10.3748/wjg.v21.i1.21.

Warnes, G.R., Bolker, B., Bonebakker, L., Gentleman, R., Huber, W., Liaw, A., Lumley, T., Maechler, M., Magnusson, A., Moeller, S., Schwartz, M., Venables, B. (2019). *gplots: Various R Programming Tools for Plotting Data. R package version 3.0.1.1.* CRAN, viewed 12 December 2019, <https://CRAN.R-project.org/package=gplots>

Appendix 1: Additional Code for Exercise 1

Code for assigning sample colours for plotting

```
# Extract phenotypic data
pDat <- pData(eset)

# Extract disease state
pClass <- as.factor(pDat$disease.state)

# Re-name longer factor levels for brevity
levels(pClass)[levels(pClass)=="Dengue Fever"] <- "Fever"
levels(pClass)[levels(pClass)=="Dengue Hemorrhagic Fever"] <- "Haemorrhagic"
levels(pClass)[levels(pClass)=="healthy control"] <- "Control"

# Associate patient IDs with disease state
names(pClass) <- sampleNames(eset)

# Create index vectors for each class
haemIndx <- which((pClass=="Haemorrhagic")==TRUE)
feverIndx <- which((pClass=="Fever")==TRUE)
convalIndx <- which((pClass=="Convalescent")==TRUE)
controlIndx <- which((pClass=="Control")==TRUE)

# Make a colour vector
colour <- NULL
colour[haemIndx] <- "red"
colour[feverIndx] <- "darkorange1"
colour[convalIndx] <- "yellow"
colour[controlIndx] <- "green"
```

Code for testing all combinations of distance and linkage algorithms, and plotting the results

```
# Perform HCA
test001 <- hclust(dist(t(X),method="euclidean"),method="complete")
test002 <- hclust(dist(t(X),method="maximum"),method="complete")
test003 <- hclust(dist(t(X),method="manhattan"),method="complete")
test004 <- hclust(dist(t(X),method="canberra"),method="complete")
test005 <- hclust(dist(t(X),method="binary"),method="complete")
test006 <- hclust(dist(t(X),method="minkowski"),method="complete")

test007 <- hclust(dist(t(X),method="euclidean"),method="ward.D")
test008 <- hclust(dist(t(X),method="maximum"),method="ward.D")
test009 <- hclust(dist(t(X),method="manhattan"),method="ward.D")
test010 <- hclust(dist(t(X),method="canberra"),method="ward.D")
test011 <- hclust(dist(t(X),method="binary"),method="ward.D")
test012 <- hclust(dist(t(X),method="minkowski"),method="ward.D")

test013 <- hclust(dist(t(X),method="euclidean"),method="ward.D2")
test014 <- hclust(dist(t(X),method="maximum"),method="ward.D2")
test015 <- hclust(dist(t(X),method="manhattan"),method="ward.D2")
test016 <- hclust(dist(t(X),method="canberra"),method="ward.D2")
test017 <- hclust(dist(t(X),method="binary"),method="ward.D2")
test018 <- hclust(dist(t(X),method="minkowski"),method="ward.D2")

test019 <- hclust(dist(t(X),method="euclidean"),method="single")
test020 <- hclust(dist(t(X),method="maximum"),method="single")
test021 <- hclust(dist(t(X),method="manhattan"),method="single")
test022 <- hclust(dist(t(X),method="canberra"),method="single")
test023 <- hclust(dist(t(X),method="binary"),method="single")
test024 <- hclust(dist(t(X),method="minkowski"),method="single")

test025 <- hclust(dist(t(X),method="euclidean"),method="average")
test026 <- hclust(dist(t(X),method="maximum"),method="average")
test027 <- hclust(dist(t(X),method="manhattan"),method="average")
test028 <- hclust(dist(t(X),method="canberra"),method="average")
test029 <- hclust(dist(t(X),method="binary"),method="average")
test030 <- hclust(dist(t(X),method="minkowski"),method="average")

test031 <- hclust(dist(t(X),method="euclidean"),method="mcquitty")
test032 <- hclust(dist(t(X),method="maximum"),method="mcquitty")
test033 <- hclust(dist(t(X),method="manhattan"),method="mcquitty")
test034 <- hclust(dist(t(X),method="canberra"),method="mcquitty")
test035 <- hclust(dist(t(X),method="binary"),method="mcquitty")
test036 <- hclust(dist(t(X),method="minkowski"),method="mcquitty")

test037 <- hclust(dist(t(X),method="euclidean"),method="median")
test038 <- hclust(dist(t(X),method="maximum"),method="median")
```

```
test039 <- hclust(dist(t(X),method="manhattan"),method="median")
test040 <- hclust(dist(t(X),method="canberra"),method="median")
test041 <- hclust(dist(t(X),method="binary"),method="median")
test042 <- hclust(dist(t(X),method="minkowski"),method="median")

test043 <- hclust(dist(t(X),method="euclidean"),method="centroid")
test044 <- hclust(dist(t(X),method="maximum"),method="centroid")
test045 <- hclust(dist(t(X),method="manhattan"),method="centroid")
test046 <- hclust(dist(t(X),method="canberra"),method="centroid")
test047 <- hclust(dist(t(X),method="binary"),method="centroid")
test048 <- hclust(dist(t(X),method="minkowski"),method="centroid")

# Create dendrograms
dend001 <- as.dendrogram(test001)
dend002 <- as.dendrogram(test002)
dend003 <- as.dendrogram(test003)
dend004 <- as.dendrogram(test004)
dend005 <- as.dendrogram(test005)
dend006 <- as.dendrogram(test006)
dend007 <- as.dendrogram(test007)
dend008 <- as.dendrogram(test008)
dend009 <- as.dendrogram(test009)
dend010 <- as.dendrogram(test010)
dend011 <- as.dendrogram(test011)
dend012 <- as.dendrogram(test012)
dend013 <- as.dendrogram(test013)
dend014 <- as.dendrogram(test014)
dend015 <- as.dendrogram(test015)
dend016 <- as.dendrogram(test016)
dend017 <- as.dendrogram(test017)
dend018 <- as.dendrogram(test018)
dend019 <- as.dendrogram(test019)
dend020 <- as.dendrogram(test020)
dend021 <- as.dendrogram(test021)
dend022 <- as.dendrogram(test022)
dend023 <- as.dendrogram(test023)
dend024 <- as.dendrogram(test024)
dend025 <- as.dendrogram(test025)
dend026 <- as.dendrogram(test026)
dend027 <- as.dendrogram(test027)
dend028 <- as.dendrogram(test028)
dend029 <- as.dendrogram(test029)
dend030 <- as.dendrogram(test030)
dend031 <- as.dendrogram(test031)
dend032 <- as.dendrogram(test032)
dend033 <- as.dendrogram(test033)
dend034 <- as.dendrogram(test034)
dend035 <- as.dendrogram(test035)
dend036 <- as.dendrogram(test036)
dend037 <- as.dendrogram(test037)
dend038 <- as.dendrogram(test038)
dend039 <- as.dendrogram(test039)
dend040 <- as.dendrogram(test040)
dend041 <- as.dendrogram(test041)
dend042 <- as.dendrogram(test042)
dend043 <- as.dendrogram(test043)
dend044 <- as.dendrogram(test044)
dend045 <- as.dendrogram(test045)
dend046 <- as.dendrogram(test046)
dend047 <- as.dendrogram(test047)
dend048 <- as.dendrogram(test048)

# Create colour vector
colourCodes <- c(Control="green",Haemorrhagic="red",Fever="darkorange1",Convalescent="yellow")

# Plot dendrograms
par(mfrow=c(2,3))
labels_colors(dend001) <- colourCodes[pClass][order.dendrogram(dend001)]
plot(dend001)
labels_colors(dend002) <- colourCodes[pClass][order.dendrogram(dend002)]
plot(dend002)
labels_colors(dend003) <- colourCodes[pClass][order.dendrogram(dend003)]
plot(dend003)
labels_colors(dend004) <- colourCodes[pClass][order.dendrogram(dend004)]
plot(dend004)
labels_colors(dend005) <- colourCodes[pClass][order.dendrogram(dend005)]
plot(dend005)
labels_colors(dend006) <- colourCodes[pClass][order.dendrogram(dend006)]
plot(dend006)
labels_colors(dend007) <- colourCodes[pClass][order.dendrogram(dend007)]
plot(dend007)
```

```
labels_colors(dend008) <- colourCodes[pClass][order.dendrogram(dend008)]
plot(dend008)
labels_colors(dend009) <- colourCodes[pClass][order.dendrogram(dend009)]
plot(dend009)
labels_colors(dend010) <- colourCodes[pClass][order.dendrogram(dend010)]
plot(dend010)
labels_colors(dend011) <- colourCodes[pClass][order.dendrogram(dend011)]
plot(dend011)
labels_colors(dend012) <- colourCodes[pClass][order.dendrogram(dend012)]
plot(dend012)
labels_colors(dend013) <- colourCodes[pClass][order.dendrogram(dend013)]
plot(dend013)
labels_colors(dend014) <- colourCodes[pClass][order.dendrogram(dend014)]
plot(dend014)
labels_colors(dend015) <- colourCodes[pClass][order.dendrogram(dend015)]
plot(dend015)
labels_colors(dend016) <- colourCodes[pClass][order.dendrogram(dend016)]
plot(dend016)
labels_colors(dend017) <- colourCodes[pClass][order.dendrogram(dend017)]
plot(dend017)
labels_colors(dend018) <- colourCodes[pClass][order.dendrogram(dend018)]
plot(dend018)
labels_colors(dend019) <- colourCodes[pClass][order.dendrogram(dend019)]
plot(dend019)
labels_colors(dend020) <- colourCodes[pClass][order.dendrogram(dend020)]
plot(dend020)
labels_colors(dend021) <- colourCodes[pClass][order.dendrogram(dend021)]
plot(dend021)
labels_colors(dend022) <- colourCodes[pClass][order.dendrogram(dend022)]
plot(dend022)
labels_colors(dend023) <- colourCodes[pClass][order.dendrogram(dend023)]
plot(dend023)
labels_colors(dend024) <- colourCodes[pClass][order.dendrogram(dend024)]
plot(dend024)
labels_colors(dend025) <- colourCodes[pClass][order.dendrogram(dend025)]
plot(dend025)
labels_colors(dend026) <- colourCodes[pClass][order.dendrogram(dend026)]
plot(dend026)
labels_colors(dend027) <- colourCodes[pClass][order.dendrogram(dend027)]
plot(dend027)
labels_colors(dend028) <- colourCodes[pClass][order.dendrogram(dend028)]
plot(dend028)
labels_colors(dend029) <- colourCodes[pClass][order.dendrogram(dend029)]
plot(dend029)
labels_colors(dend030) <- colourCodes[pClass][order.dendrogram(dend030)]
plot(dend030)
labels_colors(dend031) <- colourCodes[pClass][order.dendrogram(dend031)]
plot(dend031)
labels_colors(dend032) <- colourCodes[pClass][order.dendrogram(dend032)]
plot(dend032)
labels_colors(dend033) <- colourCodes[pClass][order.dendrogram(dend033)]
plot(dend033)
labels_colors(dend034) <- colourCodes[pClass][order.dendrogram(dend034)]
plot(dend034)
labels_colors(dend035) <- colourCodes[pClass][order.dendrogram(dend035)]
plot(dend035)
labels_colors(dend036) <- colourCodes[pClass][order.dendrogram(dend036)]
plot(dend036)
labels_colors(dend037) <- colourCodes[pClass][order.dendrogram(dend037)]
plot(dend037)
labels_colors(dend038) <- colourCodes[pClass][order.dendrogram(dend038)]
plot(dend038)
labels_colors(dend039) <- colourCodes[pClass][order.dendrogram(dend039)]
plot(dend039)
labels_colors(dend040) <- colourCodes[pClass][order.dendrogram(dend040)]
plot(dend040)
labels_colors(dend041) <- colourCodes[pClass][order.dendrogram(dend041)]
plot(dend041)
labels_colors(dend042) <- colourCodes[pClass][order.dendrogram(dend042)]
plot(dend042)
labels_colors(dend043) <- colourCodes[pClass][order.dendrogram(dend043)]
plot(dend043)
labels_colors(dend044) <- colourCodes[pClass][order.dendrogram(dend044)]
plot(dend044)
labels_colors(dend045) <- colourCodes[pClass][order.dendrogram(dend045)]
plot(dend045)
labels_colors(dend046) <- colourCodes[pClass][order.dendrogram(dend046)]
plot(dend046)
labels_colors(dend047) <- colourCodes[pClass][order.dendrogram(dend047)]
plot(dend047)
```

```
labels_colors(dend048) <- colourCodes[pClass][order.dendrogram(dend048)]
plot(dend048)
```

Appendix 2: R Script for Exercise 2

```
# Load required packages
library(classyfire)
library(R.matlab)
library(caret)

# Read in files
blood <- readMat("blood/BWG_BL_CDvALL.mat")
breath <- readMat("breath/BWG_BR_CDvALL.mat")
faecal <- readMat("faecal/BWG_FA_CDvALL.mat")
urine <- readMat("urine/BWG_UR_CDvALL.mat")

# Create a data matrix for each
Xblood <- blood$XTIC
Xbreath <- breath$XTIC
Xfaecal <- faecal$XTIC
Xurine <- urine$XTIC

# Make a class for binary classification
yblood <- blood$CLASS # Class (1=CD, 2=everything else)
ybreath <- breath$CLASS
yfaecal <- faecal$CLASS
yurine <- urine$CLASS

# Make a vector of patient IDs (contains disease initials)
bloodpatients <- unlist(blood$SAM)
breathpatients <- unlist(breath$SAM)
faecalpatients <- unlist(faecal$SAM)
urinepatients <- unlist(urine$SAM)

# Make class vectors for multiclass classification

#####BLOOD
ybloodmulti <- vector()
for (i in bloodpatients){
  if (grepl("CD",i,fixed=TRUE)==TRUE){
    ybloodmulti=c(ybloodmulti,1)
  }
  if (grepl("CTRL",i,fixed=TRUE)==TRUE){
    ybloodmulti=c(ybloodmulti,2)
  }
  if (grepl("IBS",i,fixed=TRUE)==TRUE){
    ybloodmulti=c(ybloodmulti,3)
  }
  if (grepl("UC",i,fixed=TRUE)==TRUE){
    ybloodmulti=c(ybloodmulti,4)
  }
}

table(yblood)
# 1  2
# 14 43

table(ybloodmulti)
# 1  2  3  4
# 14 18 14 11

#####BREATH
ybreathmulti <- vector()
for (i in breathpatients){
  if (grepl("CD",i,fixed=TRUE)==TRUE){
    ybreathmulti=c(ybreathmulti,1)
  }
  if (grepl("CTRL",i,fixed=TRUE)==TRUE){
    ybreathmulti=c(ybreathmulti,2)
  }
  if (grepl("IBS",i,fixed=TRUE)==TRUE){
    ybreathmulti=c(ybreathmulti,3)
  }
  if (grepl("UC",i,fixed=TRUE)==TRUE){
    ybreathmulti=c(ybreathmulti,4)
  }
}

table(ybreath)
# 1  2
# 17 43
```

```
table(ybreathmulti)
# 1  2  3  4
# 17 19 14 10

#####FAECAL
yfaecmulti <- vector()
for (i in faecalpatients){
  if (grepl("CD",i,fixed=TRUE)==TRUE){
    yfaecmulti=c(yfaecmulti,1)
  }
  if (grepl("CTRL",i,fixed=TRUE)==TRUE){
    yfaecmulti=c(yfaecmulti,2)
  }
  if (grepl("IBS",i,fixed=TRUE)==TRUE){
    yfaecmulti=c(yfaecmulti,3)
  }
  if (grepl("UC",i,fixed=TRUE)==TRUE){
    yfaecmulti=c(yfaecmulti,4)
  }
}

table(yfaecal)
# 1  2
# 11 34

table(yfaecmulti)
# 1  2  3  4
# 11 13 11 10

#####URINE
yurinemulti <- vector()
for (i in urinepatients){
  if (grepl("CD",i,fixed=TRUE)==TRUE){
    yurinemulti=c(yurinemulti,1)
  }
  if (grepl("CTRL",i,fixed=TRUE)==TRUE){
    yurinemulti=c(yurinemulti,2)
  }
  if (grepl("IBS",i,fixed=TRUE)==TRUE){
    yurinemulti=c(yurinemulti,3)
  }
  if (grepl("UC",i,fixed=TRUE)==TRUE){
    yurinemulti=c(yurinemulti,4)
  }
}

table(yurine)
# 1  2
# 7 29

table(yurinemulti)
# 1  2  3  4
# 7 14  8  7

# Plot these frequencies:
par(mfrow=c(2,4))
plot(table(yblood),main="Blood, binary")
plot(table(ybloodmulti),main="Blood, multi")
plot(table(ybreath),main="Breath, binary")
plot(table(ybreathmulti),main="Breath, multi")
plot(table(yfaecal),main="Faecal, binary")
plot(table(yfaecmulti),main="Faecal, multi")
plot(table(yurine),main="Urine, binary")
plot(table(yurinemulti),main="Urine, multi")

# Use the sample IDs row names
rownames(Xblood) <- as.character(bloodpatients)
rownames(Xbreath) <- as.character(breathpatients)
rownames(Xfaecal) <- as.character(faecalpatients)
rownames(Xurine) <- as.character(urinepatients)

##### Data pre-processing #####

# Scale the data
Xbloodsc <- scale(Xblood, center=TRUE, scale=TRUE)
Xbreathsc <- scale(Xbreath, center=TRUE, scale=TRUE)
Xfaecalsc <- scale(Xfaecal, center=TRUE, scale=TRUE)
Xurinesc <- scale(Xurine, center=TRUE, scale=TRUE)

# Perform PCA
Xbloodpca <- prcomp(Xbloodsc,scale=FALSE) # Already scaled it
```

```
s_Xbloodpca <- summary(Xbloodpca)

Xbreathpca <- prcomp(Xbreathsc,scale=FALSE)
s_Xbreathpca <- summary(Xbreathpca)

Xfaecalpca <- prcomp(Xfaecalsc,scale=FALSE)
s_Xfaecalpca <- summary(Xfaecalpca)

Xurinepca <- prcomp(Xurinesc,scale=FALSE)
s_Xurinepca <- summary(Xurinepca)

# Get a vector of the explained variance for each PC
expVarblood <- s_Xbloodpca$importance[2,]*100
expVarbreath <- s_Xbreathpca$importance[2,]*100
expVarfaec <- s_Xfaecalpca$importance[2,]*100
expVarurine <- s_Xurinepca$importance[2,]*100

# Plot explained variances for first ten PCs
par(mfrow=c(2,2))
barplot(expVarblood[1:10], xlab="Principal Components", ylab="Explained Variance (%)",
        col="red", main="Blood", las=2)
barplot(expVarbreath[1:10], xlab="Principal Components", ylab="Explained Variance (%)",
        col="blue", main="Breath", las=2)
barplot(expVarfaec[1:10], xlab="Principal Components", ylab="Explained Variance (%)",
        col="green", main="Faeces", las=2)
barplot(expVarurine[1:10], xlab="Principal Components", ylab="Explained Variance (%)",
        col="yellow", main="Urine", las=2)

# Get a vector of the cumulative variance for each PC
cumVarblood <- s_Xbloodpca$importance[3,]*100
cumVarbreath <- s_Xbreathpca$importance[3,]*100
cumVarfaec <- s_Xfaecalpca$importance[3,]*100
cumVarurine <- s_Xurinepca$importance[3,]*100

# Plot explained variances for first ten PCs
par(mfrow=c(2,2))
plot(cumVarblood[1:10], type="o", col="black", pch=21, bg="red", cex=1.2,
     ylab="Cumulative Variance (%)", xlab="Principal Components",
     main="Blood", xaxt="n")
axis(side=1,at=c(1,2,3,4,5,6,7,8,9,10))
plot(cumVarbreath[1:10], type="o", col="black", pch=21, bg="blue", cex=1.2,
     ylab="Cumulative Variance (%)", xlab="Principal Components",
     main="Breath", xaxt="n")
axis(side=1,at=c(1,2,3,4,5,6,7,8,9,10))
plot(cumVarfaec[1:10], type="o", col="black", pch=21, bg="green", cex=1.2,
     ylab="Cumulative Variance (%)", xlab="Principal Components",
     main="Faeces", xaxt="n")
axis(side=1,at=c(1,2,3,4,5,6,7,8,9,10))
plot(cumVarurine[1:10], type="o", col="black", pch=21, bg="yellow", cex=1.2,
     ylab="Cumulative Variance (%)", xlab="Principal Components",
     main="Urine", xaxt="n")
axis(side=1,at=c(1,2,3,4,5,6,7,8,9,10))

# Index each sample by condition

# Convert class vectors to characters
bloodnames <- as.character(ybloodmulti)
breathnames <- as.character(ybreathmulti)
faecalnames <- as.character(yfaecmulti)
urinenames <- as.character(yurinemulti)

# Make selection vectors
bl_CD_idx <- which((bloodnames=="1")==TRUE)
bl_CTRL_idx <- which((bloodnames=="2")==TRUE)
bl_IBS_idx <- which((bloodnames=="3")==TRUE)
bl_UC_idx <- which((bloodnames=="4")==TRUE)

br_CD_idx <- which((breathnames=="1")==TRUE)
br_CTRL_idx <- which((breathnames=="2")==TRUE)
br_IBS_idx <- which((breathnames=="3")==TRUE)
br_UC_idx <- which((breathnames=="4")==TRUE)

fa_CD_idx <- which((faecalnames=="1")==TRUE)
fa_CTRL_idx <- which((faecalnames=="2")==TRUE)
fa_IBS_idx <- which((faecalnames=="3")==TRUE)
fa_UC_idx <- which((faecalnames=="4")==TRUE)

ur_CD_idx <- which((urinenames=="1")==TRUE)
ur_CTRL_idx <- which((urinenames=="2")==TRUE)
ur_IBS_idx <- which((urinenames=="3")==TRUE)
ur_UC_idx <- which((urinenames=="4")==TRUE)
```

```
# Make vector of colours for each sample type
bl_colour <- NULL
bl_colour[bl_CD_idx] <- "red"
bl_colour[bl_CTRL_idx] <- "green"
bl_colour[bl_IBS_idx] <- "darkorange1"
bl_colour[bl_UC_idx] <- "yellow"

br_colour <- NULL
br_colour[br_CD_idx] <- "red"
br_colour[br_CTRL_idx] <- "green"
br_colour[br_IBS_idx] <- "darkorange1"
br_colour[br_UC_idx] <- "yellow"

fa_colour <- NULL
fa_colour[fa_CD_idx] <- "red"
fa_colour[fa_CTRL_idx] <- "green"
fa_colour[fa_IBS_idx] <- "darkorange1"
fa_colour[fa_UC_idx] <- "yellow"

ur_colour <- NULL
ur_colour[ur_CD_idx] <- "red"
ur_colour[ur_CTRL_idx] <- "green"
ur_colour[ur_IBS_idx] <- "darkorange1"
ur_colour[ur_UC_idx] <- "yellow"

# Plot Principal Component plots for PC1 and PC2

# Get expVars for axis labels
expVarblood[1] # 58.83
expVarblood[2] # 12.61
expVarbreath[1] # 74.97
expVarbreath[2] # 11.42
expVarfaec[1] # 49.42
expVarfaec[2] # 12.88
expVarurine[1] # 67.62
expVarurine[2] # 12.03

par(mfrow=c(2,2))
plot(Xbloodpca$x, xlab="PC1 (58.83%)", ylab="PC2 (12.61%)",
     pch=21, cex=1.2, bg=bl_colour, cex.lab=1.2, cex.axis=1.2,
     main="Blood")
plot(Xbreathpca$x, xlab="PC1 (74.97%)", ylab="PC2 (11.42%)",
     pch=21, cex=1.2, bg=br_colour, cex.lab=1.2, cex.axis=1.2,
     main="Breath")
plot(Xfaecalpca$x, xlab="PC1 (49.42%)", ylab="PC2 (12.88%)",
     pch=21, cex=1.2, bg=fa_colour, cex.lab=1.2, cex.axis=1.2,
     main="Faeces")
plot(Xurinepca$x, xlab="PC1 (67.62%)", ylab="PC2 (12.03%)",
     pch=21, cex=1.2, bg=ur_colour, cex.lab=1.2, cex.axis=1.2,
     main="Urine")
# Use enough PCs to capture 90% of the variation

cumVarblood[8] # 91.00%
cumVarbreath[4] # 92.39%
cumVarfaec[10] # 90.66%
cumVarurine[6] # 90.09%

# Take a new matrix using the correct number of PCs
Xblood2 <- Xbloodpca$x[,1:8]
Xbreath2 <- Xbreathpca$x[,1:4]
Xfaecal2 <- Xfaecalpca$x[,1:10]
Xurine2 <- Xurinepca$x[,1:6]

# Make matrices from class vectors
ybloodmulti <- as.matrix(ybloodmulti)
ybreathmulti <- as.matrix(ybreathmulti)
yfaecmulti <- as.matrix(yfaecmulti)
yurinemulti <-as.matrix(yurinemulti)

# Build a classification ensemble with classyfire
blood_ens <- cfBuild(Xblood2,ybloodmulti,
                     bootNum=30,ensNum=30)

breath_ens <- cfBuild(Xbreath2,ybreathmulti,
                     bootNum=30,ensNum=30)

faecal_ens <- cfBuild(Xfaecal2,yfaecmulti,
                     bootNum=30,ensNum=30)

urine_ens <- cfBuild(Xurine2,yurinemulti,
                     bootNum=30,ensNum=30)
```

```
# % of correctly classified objects
getAvgAcc(blood_ens)$Test # 29.83
getAvgAcc(breath_ens)$Test # 32.83
getAvgAcc(faecal_ens)$Test # 28.46
getAvgAcc(urine_ens)$Test # 35.84

# Look at confusion matrices
blood_CF <- getConfMatr(blood_ens)
blood_CF_stats <- confusionMatrix(blood_CF)
# Accuracy 0.2725
#                       Class: 1 Class: 2 Class: 3 Class: 4
# Sensitivity            0.3482   0.2214   0.3229   0.1639
# Specificity            0.7882   0.7323   0.7763   0.7345

breath_CF <- getConfMatr(breath_ens)
breath_CF_stats <- confusionMatrix(breath_CF)
# Accuracy 0.2563
#                        Class: 1 Class: 2 Class: 3 Class: 4
# Sensitivity            0.3478  0.26190  0.20000  0.14894
# Specificity            0.7915  0.75368  0.73264  0.73504

faecal_CF <- getConfMatr(faecal_ens)
faecal_CF_stats <- confusionMatrix(faecal_CF)
# Accuracy : 0.3025
#                       Class: 1 Class: 2 Class: 3 Class: 4
# Sensitivity            0.4336   0.2087   0.2755   0.2838
# Specificity            0.8223   0.7333   0.7583   0.7577

urine_CF <- getConfMatr(urine_ens)
urine_CF_stats <- confusionMatrix(urine_CF)
# Accuracy : 0.2095
#                      Class: 1 Class: 2 Class: 3 Class: 4
# Sensitivity          0.044776   0.3056  0.26364  0.10000
# Specificity          0.709581   0.7782  0.75601  0.71340

# Graphical representation of class predictions against
# actual class:
ggClassPred(blood_ens, displayAll=TRUE, fillBrewer=TRUE,
            showText=TRUE)
ggClassPred(breath_ens, displayAll=TRUE, fillBrewer=TRUE,
            showText=TRUE)
ggClassPred(faecal_ens, displayAll=TRUE, fillBrewer=TRUE,
            showText=TRUE)
ggClassPred(urine_ens, displayAll=TRUE, fillBrewer=TRUE,
            showText=TRUE)

# Plot histograms of individual test accuracy
par(mfrow=c(2,2))
hist(blood_ens$testAcc)
hist(breath_ens$testAcc)
hist(faecal_ens$testAcc)
hist(urine_ens$testAcc)

# Boxplot of test accuracy for each sample type
boxplot(blood_ens$testAcc,breath_ens$testAcc,
        faecal_ens$testAcc,urine_ens$testAcc,
        col=c("red","blue","green","yellow"),
        names=c("Blood","Breath","Faeces","Urine"),
        main="Test Accuracy")

# Plot accuracy, specificity and sensitivity for Crohn's
# according to confusionMatrix()
Accuracy <- c(blood_CF_stats$overall[1]*100,
              breath_CF_stats$overall[1]*100,
              faecal_CF_stats$overall[1]*100,
              urine_CF_stats$overall[1]*100)

names(Accuracy) <- c("Blood","Breath","Faecal","Urine")
barplot(Accuracy,col=c("red","blue","green","yellow"),
        main="Accuracy",
        ylab="%",ylim=c(0,50))

CrohnsSensitivity <- c(blood_CF_stats$byClass[1,1]*100,
                       breath_CF_stats$byClass[1,1]*100,
                       faecal_CF_stats$byClass[1,1]*100,
                       urine_CF_stats$byClass[1,1]*100)

names(CrohnsSensitivity) <- c("Blood","Breath","Faecal","Urine")
barplot(CrohnsSensitivity,col=c("red","blue","green","yellow"),
        main="Crohn's sensitivity",
        ylab="%",ylim=c(0,50))
```

```
CrohnsSpecificity <- c(blood_CF_stats$byClass[1,2]*100,
                       breath_CF_stats$byClass[1,2]*100,
                       faecal_CF_stats$byClass[1,2]*100,
                       urine_CF_stats$byClass[1,2]*100)

names(CrohnsSpecificity) <- c("Blood","Breath","Faecal","Urine")
barplot(CrohnsSpecificity,col=c("red","blue","green","yellow"),
        main="Crohn's specificity",
        ylab="%",ylim=c(0,90))


# Was choice of number of ensembles good?
ggEnsTrend(blood_ens, ylim=c(10,100))
ggEnsTrend(breath_ens, ylim=c(10,100))
ggEnsTrend(faecal_ens, ylim=c(10,100))
ggEnsTrend(urine_ens, ylim=c(10,100))

# Permutation testing
blood_permute <- cfPermute(Xblood2,
                           ybloodmulti,
                           bootNum=30,
                           ensNum=30,
                           permNum=5)

breath_permute <- cfPermute(Xbreath2,
                            ybreathmulti,
                            bootNum=30,
                            ensNum=30,
                            permNum=5)

faecal_permute <- cfPermute(Xfaecal2,
                            yfaecmulti,
                            bootNum=30,
                            ensNum=30,
                            permNum=5)

urine_permute <- cfPermute(Xurine2,
                           yurinemulti,
                           bootNum=30,
                           ensNum=30,
                           permNum=5)

# Average accuracies after permutation testing

# Plot histograms of individual test accuracy
par(mfrow=c(2,2))

hist(blood_ens$testAcc)
hist(blood_permute$avgAcc)

hist(breath_ens$testAcc)
hist(breath_permute$avgAcc)

hist(faecal_ens$testAcc)
hist(faecal_permute$avgAcc)

hist(urine_ens$testAcc)
hist(urine_permute$avgAcc)

boxplot(blood_ens$testAcc,
        blood_permute$avgAcc,
        breath_ens$testAcc,
        breath_permute$avgAcc,
        faecal_ens$testAcc,
        faecal_permute$avgAcc,
        urine_ens$testAcc,
        urine_permute$avgAcc,
        col=c("red","red","blue","blue","green","green","yellow","yellow"),
        las=2,
        names=c("Blood","Blood, p","Breath","Breath, p",
        "Faeces","Faeces, p","Urine", "Urine, p"))


blood_permute$avgAcc
# 25.67 27.33 27.83 29.17 28.50
mean(blood_permute$avgAcc) # 27.7

breath_permute$avgAcc
# 33.17 28.00 30.17 27.33 31.00
mean(breath_permute$avgAcc) # 29.934
```

```
faecal_permute$avgAcc
# 22.00 24.89 24.89 22.00 31.11
mean(faecal_permute$avgAcc) # 24.978

urine_permute$avgAcc
# 38.89 41.39 40.28 38.33 38.89
mean(urine_permute$avgAcc) # 39.556

# Original average accuracies

getAvgAcc(blood_ens)$Test # 29.83
getAvgAcc(breath_ens)$Test # 32.83
getAvgAcc(faecal_ens)$Test # 28.46
getAvgAcc(urine_ens)$Test # 35.84
```