Queen Mary University of London

School of Biological and Chemical Sciences

# Coursework Coversheet

| Student Details<br>Place barcode in the dashed box below. | MARK |
|---|---|
| 1 7 0 4 8 4 3 6 7 | |
| GRIGORIADIS, Dionysios | |

| | | |
|---|---|---|
| **BIO725P** | | BESSANT, CONRAD |
| **POST GENOMIC BIOINFORMATICS** | | |
| **COURSE ASSIGNMENT** | | |
| **Friday, 15 December 2017, 6:00 PM** | | |

# Feedback

If generic feedback is provided on the module webpage then please indicate by ticking here: ☐

**General comments**

**Suggestions for improvement**

# Declaration [to be completed by student]

I certify that this coursework that I am submitting is my own work, that it has not been copied in part or in whole from any other person, and that any ideas or quotations from the work of other people, published or otherwise, are properly referenced. I have read and understood the School guidelines on plagiarism and I am aware that penalties will be applied for any plagiarism or other poor academic practice.

| | |
|---|---|
| | 15/12/2017 |

# Table of Contents

# Question 1

## Introduction

Dengue is a mosquito-borne viral infection (dengue virus) more commonly found in tropical areas. The main symptomatology of the disease includes flu-like symptoms, however dengue is possible to develop into the life-threatening severe Dengue (Hemorrhagic Fever) (Bhatt *et al.*, 2013). In this analysis, whole blood transcriptome analysis data of samples from patients with Dengue Fever (n=18), Haemorrhagic Fever (n=10), patients during convalescence (n=19) and healthy individuals (9) were used to investigate the relationships between the gene expression profiles of these populations and to identify the genes that show the most significantly differential expression among these populations. This expression dataset was created for the purposes of a study available on Gene Expression Omnibus (GEO) from NCBI (GEO accession number: GDS5093) analysing the whole blood of patients with acute Dengue virus (DENV) infection and during convalescence (Kwissa *et al.*, 2014, data accessible at NCBI GEO database (Edgar, Domrachev and Lash, 2002), accession GSE51808). The whole-blood transcriptome of the samples was assessed with gene microarrays.

## Methods

All the below described analyses were conducted in R, using RStudio open source software (RStudio Team, 2015).

**Data import from GEO into R**

The data from GEO (GEO SOFT format file) was loaded straight into a list of objects into R (GSE Matrix) using the "GEOquery" package of the Bioconductor software (Davis and Meltzer, 2007). The data includes gene expression values for 54715 genes of 56 samples. This matrix was then converted into an ExpressionSet object which is a single convenient structure which can be used from the Bioconductor software. Disease state of the samples was also assigned to a variable in order to classify the samples into samples from convalescent patients (CP)(n=19), from healthy individuals (H), from patients with Dengue Fever (DF) and patients with Dengue Haemorrhagic Fever (DHF).

**Exploratory Data Analysis - Boxplot**

The gene expression measurements of the assay data were stored into a matrix which was used as an input for the downstream analyses. This matrix has gene names as its row names and sample IDs as its column names. This matrix summary provides the 5-number summary of the gene expression values (minimum, first quartile, median, third quartile and maximum) for each feature (sample). Based on this summary, a box plot of the sample intensities is drawn to assess if the scale and distribution of the data on different arrays is comparable. Code used for this visualisation:

```
#TableEX is the matrix with the gene expression values described above
summary(TableEX) #Provides the 5-number summary of the gene expression values
op <- par(mar = c(10,6,4,2) + 0.1)
boxplot(TableEX, col=colour, las=2, ylab = "Intensity\nGene Expression values",ann = FALSE)
mtext(side = 1, text = "Sample ID", line = 7)
par(op)
```

**Cluster Analysis**

Hierarchical clustering (HCA) was used to partition the expression dataset into several subsets based on their gene expression. The distance algorithm that used for this analysis measured the Euclidean distance between the objects, Ward's method of linkage was used, while the visualization of the clustering was graphically represented in tree structure. The HCA was performed with the built-in R function for Hierarchical Clustering; `hclust`. "dendextend" package was used to visualise the dendrogram (Galili, 2015). Transposed TableEX matrix from the exploratory analysis was used as input for clustering function. Code used for the clustering visualisation:

```
hc = hclust(dist(t(TableEX)), method = "ward.D")
dend = as.dendrogram(hc)
colorCodes = c(Control = "green", D_Fever = "red", H_Fever = "blue", Covalescent = "orange")
labels_colors(dend) = colorCodes[pClass][order.dendrogram(dend)]
par(mar=c(3,1,1,5))
plot(dend, cex = 0.6)
```

**Principal Component Analysis (PCA)**

In this analysis, the R built-in `prcomp` function was used to perform Singular value decomposition (SVP) in order to examine the correlations between the four under-investigation populations based on their gene expression values. The percentage variance for each successive PC was extracted and presented in a bar plot (see on appendix), while the two-dimensional PCA scores (PC2 vs. PC1) were plotted to reveal patterns in the data if the first two components accounted for most of the variance of the data. Code to visualise the PCA:

```
#Percentage variance for each PC is extracted and plotted as %
expVar = s$importance[2,] * 100 #s = summary of the PCA
barplot(expVar, xlab = "Principal Components", ylab ="Explained Variance (%)",
        col = "steelblue")
#PCA scores are extracted - PC2 vs PC1 plot
Xscores = Xpca$x #Xpca = the PCA
plot(Xscores, xlab = "PC1", ylab = "PC2", cex = 0.7,
cex.lab = 0.7, cex.axis = 0.7, pch = 21, bg = colour)
```

**Identification of the the genes that show the most significant difference in expression between the four examined populations**

"limma" package was used for this identification (Ritchie *et al.*, 2015). To perform this analysis, an appropriate design matrix was initially created, which was used to fit a model to the data in order to make all pair-wise comparisons (6 possible comparisons) between the four groups (H, CP, DF, DHF). The outcome of each hypothesis test was extracted in order to investigate how many and which genes had significant differentially expression in each comparison and a venn diagram was created using this information. Finally, the statistic F value and the corresponding F.p.value, which combine the six pair-wise comparisons into one F-test were used from the `toptableF` function of the package to find the top 250 genes that show the most significant difference in expression between the four examined populations. The code used for this analysis is included in detail as an appendix at the end of this report. The biological role of these proteins was identified using STICH (Szklarczyk *et al.*, 2016).

## Results & Discussion

The boxplot that was created to assess the gene expression values intensities among the samples and the under-investigation populations revealed that none of the samples stand out

from the rest (Figure 1). More specifically, all the samples have a very comparable median expression level, while the scale of the boxes is almost identical (Figure 1). Conclusively, the scale and distribution of the gene expression values is comparable between the samples and therefore the data on the previously created expression matrix is appropriate for further analysis.

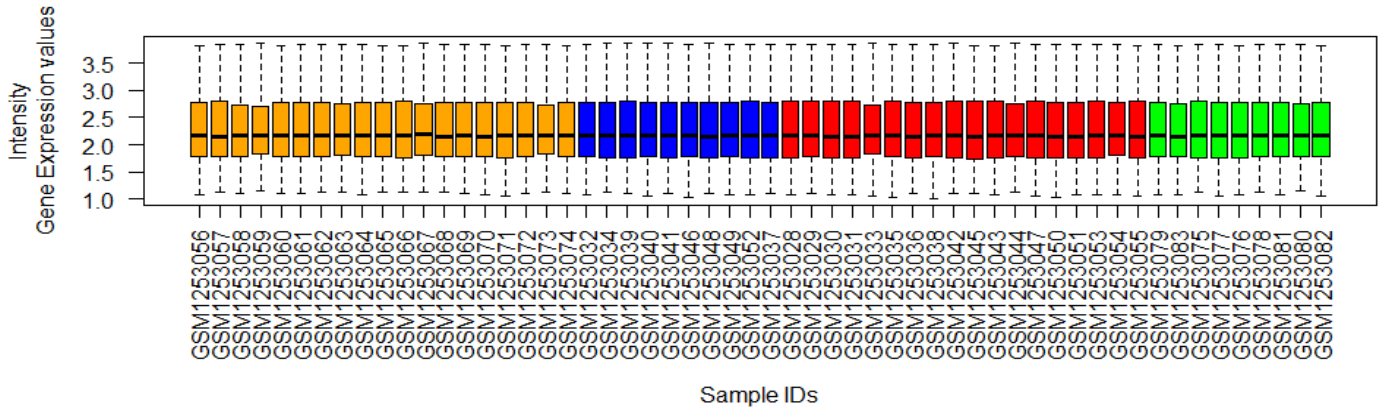


Figure 1. Box Plot for sample intensities. Bars represent interquartile range. Bar colors: -Yellow: Samples from convalescent patients, -Blue: Samples from patients with Haemorraghic Dengue Fever, -Red: Samples from patients with Dengue Fever, -Green: Samples from healthy individuals.

The dendrogram that represent the unsupervised hierarchical clustering analysis revealed that the four under-examination populations are clustered into two well separated groups (main groups) based on their gene expression (Figure 2). The first main group includes the samples from CP and H, while the second main group includes the samples from DF or DHF patients. This separation seems strong and indicates that patients with DF or DHF have significantly distinct gene expression profile in comparison with healthy patients or patients during convalescence indicating that the disease affects the gene expression profile of the patients. Moreover, there is also a clustering within these two main groups that seems to slightly separate CP and H in the first group and DF and DHF patients in the second group, although is weaker compared to the first one. This clustering of subgroups reveals that despite the shared gene expression profile between the patients of the same main groups, there might be a differential expression between CP and HP of the first main group and between DF and DHF patients of the second main group.

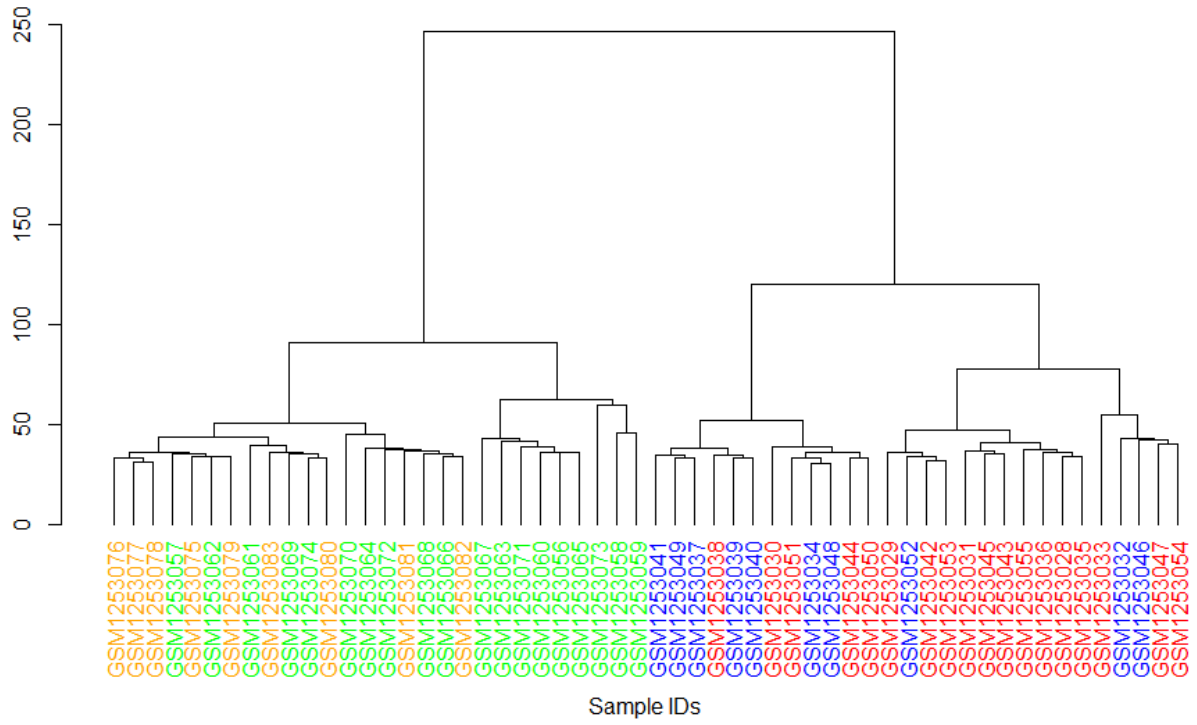


Figure 2. Cluster dendrogram results. The dendogram shows relationships in gene expression among Convalescent patients (orange), Healthy individuals (green), patients with Dengue Fever (Red) and patients with Dengue Hemorrhagic Fever (Blue).

Principal component analysis was used to reveal any underlying patterns in the data based on the gene expression values of the samples. However, the first couple of principal components accounted only for less than 30% of the variance of the data (Figure 5) and therefore the scores table cannot be that reliable. The plot of the PCA scores of the PC2 to the PC1 revealed patterns of similarities between CP and H individuals and similarities between DF and DHF patients. More specifically, the plot indicates a gene expression pattern for the one main group of H and CP and a distinct gene expression pattern for the other main group of DHF and DF patients similar to what clustering analysis showed.

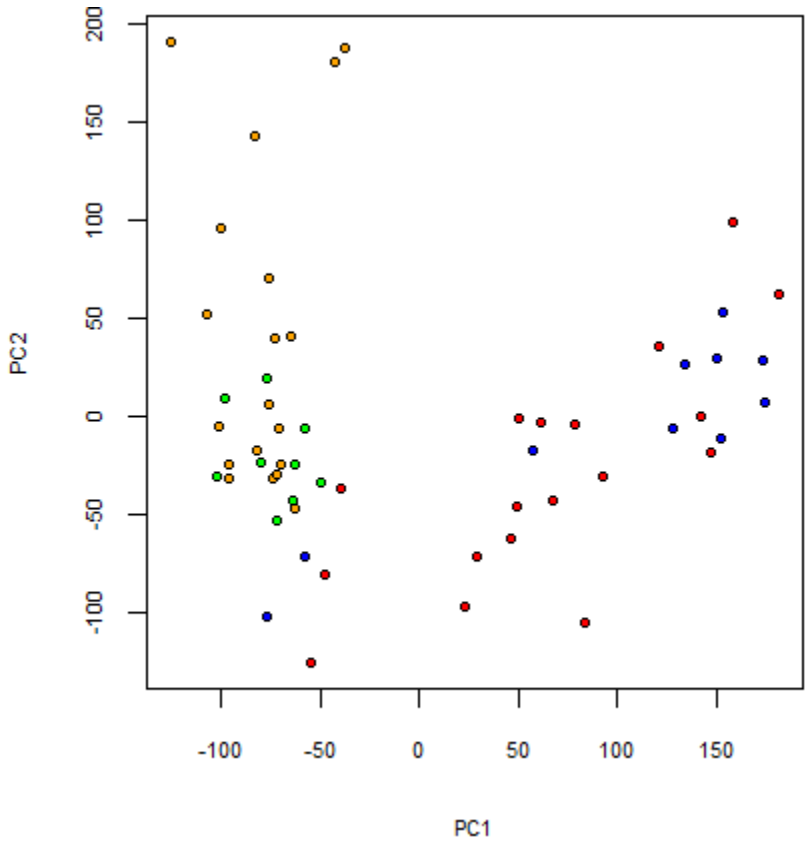


Figure 3. Plot of the PCA scores of the 2nd Principal component to the scores of the 1st one. Green: Healthy individuals, Orange: Convalescent patients, Red: Dengue Fever patients, Blue: Dengue Hemorrhagic patients.

Based on the statistic F value and the corresponding F.p.value, which combine the six pair-wise comparisons into one F-test the top 250 genes that show the most significant difference in expression between the four examined populations were identified. From these genes the top 10 are presented here (Table 1). The identification of these genes indicated that the disease really alters the gene expression profile of the patients, which is similar to what clustering dendrogram showed previously.

Table 1. Top 10 differentially expressed genes between the four examined populations.

| ID | AveExpr | F | adj.P.Val |
|---|---|---|---|
| KCTD14 | 2.212447 | 176.3946 | 1.50E-23 |
| KCTD14 | 2.265089 | 166.1715 | 3.32E-23 |
| CAMK1D | 3.143115 | 154.4923 | 1.35E-22 |
| BAG1 | 3.408736 | 132.2294 | 4.53E-21 |
| FAM72A | 2.786199 | 127.4414 | 8.82E-21 |
| BAG1 | 3.443067 | 125.7956 | 1.01E-20 |
| BAK1 | 2.710259 | 120.0307 | 2.66E-20 |
| PALM2-AKAP2 | 2.917247 | 116.5669 | 4.18E-20 |
| CEP55 | 2.519249 | 116.5359 | 4.18E-20 |
| CD38 | 3.112704 | 115.0205 | 4.82E-20 |

Then, the number of differentially expressed genes between each under-examination population with each other was assessed and used to create a descriptive Venn diagram (Figure 4). A large amount of genes were found to be differentially expressed between CP or H and DF or DHF patients, something which is consistent with the hierarchical clustering that separated the two main subgroups. However, no differentially expressed genes were found between DF and DHF patients' samples (not included in the diagram), while only 2 genes were found to have different expression levels between CP and health individuals, something which shows that these subgroups have similar gene expression profiles in contrast to the slight separation that clustering demonstrated. Eventually, only 7 genes (*CAPRIN1, CTDSP2, GPR107, TOMM40L, ATF7, ZNF33A, MED27)* were found to be differentially expressed in all five different comparisons (DF vs DHF excluded) confirming the previous conclusion that the disease really alters the gene expression profile of the patients.

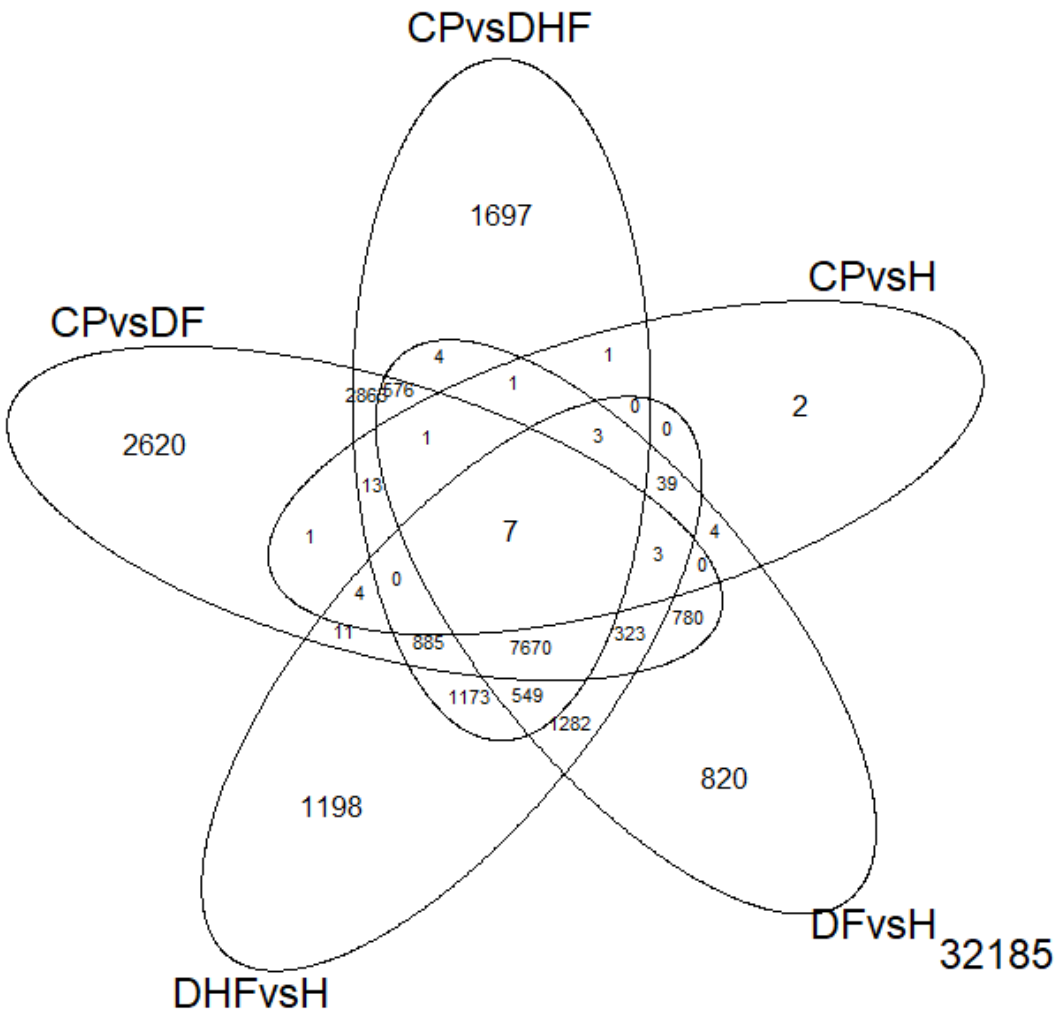


Figure 4. Venn diagram showing the number of differential expressed genes in the comparisons: CPvsDHF = 1697, CPvsDF = 2620, DHFvsH = 1198, DFvsH = 820, CPvsH = 2, CP vs DHF = 1697. The number of genes that are differentially expressed in all comparisons (center) is 7. The number of genes found differentially expressed no matter the comparison (down-right corner) is 32185

## Conclusion

The performed data analysis revealed that the Dengue disease alters the gene expression profile of the infected patients having either Dengue Fever or Dengue Hemorrhagic Fever compared to convalescent patients or healthy individuals. The heterogeneity of the gene expression profile between Infected patients and convalescent or healthy individuals is also supported by the fact that only 7 genes which have altered expression in all comparisons were found, while the number of genes with difference in their expression was nearly zero when DFvsDHF or CPvsH samples were compared. Moreover, 250 genes that show the most significant difference in expression

between those populations were identified. The majority of the proteins that these 250 genes encode seems to participate in pathways related to mitotic cell cycle process and cell cycle process in general, something which could possibly indicate that the Dengue infection acts by modifying the cell cycle.

# Appendix

## *Figures*

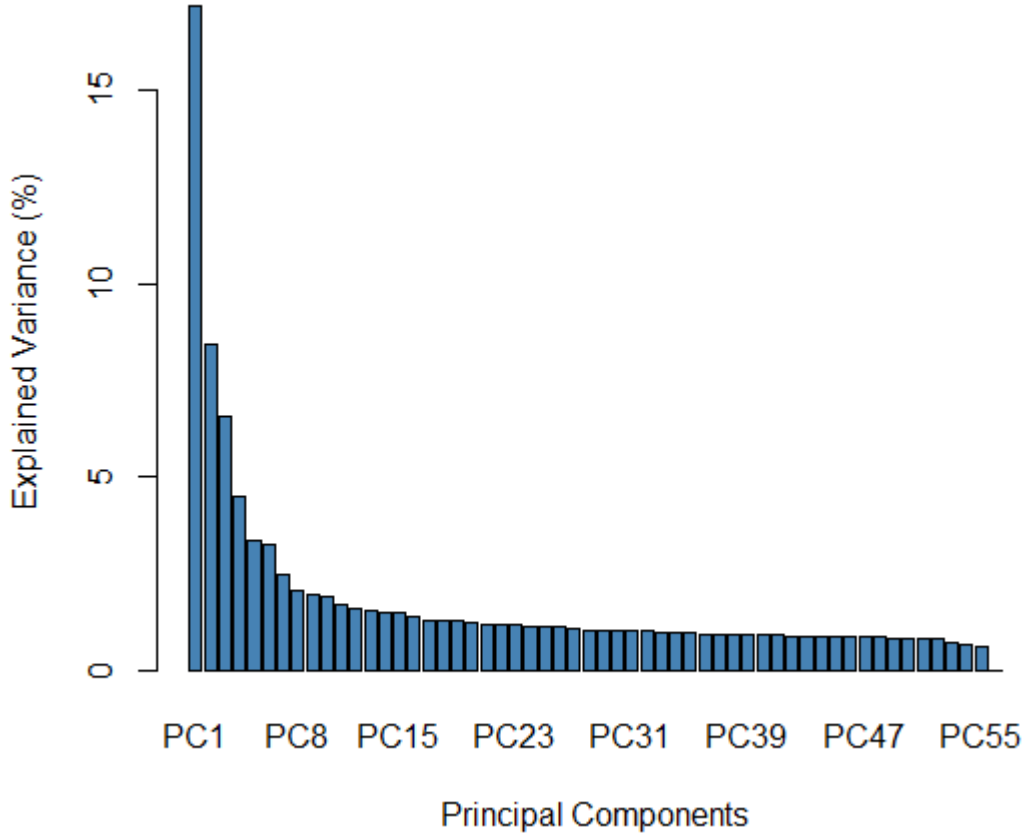


Figure 5. Percentage of the variance explained for each principal component of PCA applied to the gene expression dataset. PC1 and PC2 account for less than 30% of the total data variance.

*Code*

**Identification of the the genes that show the most significant difference in expression between the four examined populations**

```
#Identify top genes
#Design matrix:
design = model.matrix(~0 + pClass)
design

#Model fit - Pairwise comparisons - All possible combinations
fit = lmFit(TableEX, design)
contrasts = makeContrasts(CPvsDF = pClassConvalescent - pClassD_Fever, CPvsDHF =
pClassConvalescent - pClassH_Fever,
                          CPvsH = pClassConvalescent - pClassControl, DFvsDHF =
pClassD_Fever - pClassH_Fever,
                          DFvsH = pClassD_Fever - pClassControl,DHFvsH = pClassH_Fever
- pClassControl, levels = design)

fit = contrasts.fit(fit, contrasts)
fit = eBayes(fit)

#top genes:
toptable = topTableF(fit, number = 250) #250 top genes from all possible combinations

#Which & How many genes are differentially expressed between all the groups
results = decideTests(fit)
summary(results) #DFvsDHF - no significantly differentialy expressed genes - Excluded
from the analysis
de.common = which(results[,1]!=0 & results[,2]!=0 & results[,3]!=0 & results[,5]!=0 &
results[,6]!=0)
length(de.common) #how many
de.common #which

#Venn diagram:
vennDiagram(results[,-4], col=colour) #without DFvsDHF
toptable10 = topTableF(fit, number = 10) #10 top genes from all possible combinations
```

# Question 2

## Introduction

Crohn's disease (CD) is a gastrointestinal chronic disease that causes inflammation of the lining of the digestive system (M'Koma, 2013). Similarly, ulcerative colitis (UC) and irritable bowel syndrome (IBS) are also gastrointestinal diseases with similar symptomatology. In this analysis, chemometrics were applied to GC-MS data from a metabolomics experiment on different sample types (breath, blood, urine and faeces) of patients with the above types of gastrointestinal diseases. There are two types of data for each sample type. The first contains the GC-MS data from Crohn's disease samples and from healthy individuals (CDvCTRL), while the second contains the GC-MS data from Crohn's disease samples and from samples from patients which have one of the other types of gastrointestinal diseases (UC, IBS) or are healthy (CDvALL). The aim of this analysis wasto apply chemometrics to this data in order to rank the ability of each of the examined sample types to diagnose Crohn's disease.

In order to create an algorithm that successfully predicts Crohn's disease, both the CDvCTRL and CDvALL datasets can be used. More specifically, if the dataset that is used for building the classification algorithm is a CDvCTRL dataset then the algorithm could predict if a patient sample has CD but if the dataset that is used to build the classification algorithm is a CDvALL dataset then the final algorithm could hopefully predict if a patient sample has CD among patients with the above similar diseases (UC, IBS).

## Methods

All the below described analyses were conducted in R, using RStudio open source software (RStudio Team, 2015).

**Data import**

The GC-MS data are stored as .MAT files, since they have been exported from MATLAB. For each sample type (breath, blood, urine and faeces) there are two .MAT files as described previously (CDvCTRL and CDvALL). The import of each .MAT file into R was conducted using the "R.matlab" package for R (Bengtsson, 2016). For each of these files, a matrix was created (X

Matrix), which contain the sample IDs and the GC-MS data (features) that the classification algorithm will use to predict the outcome. The two classes are stored in a y variable. For CDvCTRL data (class 1 = control, class 2 = CD), for CDvALL (class 1 = CD, class 2 = ALL). Code that used for the data import:

```
bgw <- readMat("some_path/BWG_FA_CDvCTRL.mat") #Data import
x <- bgw$XTIC #X Matrix: rows -> samples columns -> features(GC-MS data)
y <- bgw$CLASS #Classes of the data: two classes 1 and 2
rownames(X) <-as.character(unlist(bgw$SAM)) #X matrix: rows = samples
```

**Data pre-processing**

The class frequencies of each dataset were investigated before applying the classification algorithm. Obviously, the distribution of the labels of the CDvsALL datasets is imbalanced, something which will probably lead to poor classification results for the Crohn's Disease class which is the minority class, even if the classification results for the majority class are good. This is not the case for the distribution frequencies of CDvsCTRL datasets which are generally in balance apart from the Breath CDvsCTRL dataset. Due to this imbalance of the class frequencies of the CDvsALL datasets, only CDvsCTRL datasets were used for classification and thus to determine the ability of each of the examined tissues to diagnose Crohn's disease.

Table 2. Class frequencies between CDvsCTRL and CDvsALL datasets.

| **CD vs CTRL datasets class frequencies** | | | | **CD vs ALL datasets class frequencies** | | | |
|---|---|---|---|---|---|---|---|
| | Class 1 | Class 2 | Difference | | Class 1 | Class 2 | Difference |
| Faecal | 12 | 11 | 1 | Faecal | 11 | 34 | 23 |
| Urine | 14 | 8 | 6 | Urine | 7 | 29 | 22 |
| Blood | 18 | 14 | 4 | Blood | 14 | 43 | 29 |
| Breath | 19 | 10 | 9 | Breath | 10 | 50 | 40 |
| Average Difference | | | 5 | Average Difference | | | 28.5 |

Moreover, scalling was applied on the input data (X Matrix). Scaling is done by dividing the values by their standard deviations using the `scale` built-in R function.

Principal component analysis was also applied to the X Matrix. More specifically, the R built-in `prcomp` function was used to perform Singular value decomposition (SVP) on the previously scaled data. Cumulative variance was then calculated for all the resulted principal components. Only principal components that explain until 90%-91% of the variance are kept, and the rest are thrown out. Thus, only the Scores of these PCs were used to build the classifier instead of the raw data. This will lead to a quicker classification with a better generalisation performance.

Table 3. Cumulative variance for all the CDvsCTRL datasets. Only the 13 first PCs are shown here.

| | **FA_CDvsCTRL** | **UR_CDvsCTRL** | **BL_CDvsCTRL** | **BR_CDvsCTRL** |
|---|---|---|---|---|
| **PC1** | 31.532 | 74.187 | 55.733 | 59.234 |
| **PC2** | 54.501 | 83.906 | 72.123 | 71.391 |
| **PC3** | 68.643 | 88.381 | 84.696 | 82.315 |
| **PC4** | 76.588 | 90.801 | 87.677 | 86.807 |
| **PC5** | 80.313 | 92.727 | 89.964 | 90.07 |
| **PC6** | 83.657 | 94.353 | 91.769 | 92.58 |
| **PC7** | 86.732 | 95.651 | 93.361 | 94.698 |
| **PC8** | 88.749 | 96.554 | 94.583 | 95.88 |
| **PC9** | 90.679 | 97.199 | 95.659 | 96.662 |
| **PC10** | 92.08 | 97.761 | 96.428 | 97.392 |
| **PC11** | 93.259 | 98.228 | 97.064 | 98.01 |
| **PC12** | 94.353 | 98.557 | 97.645 | 98.399 |
| **PC13** | 95.383 | 98.842 | 98.111 | 98.722 |

Summarizing, for each sample type, a single classification ensemble was built with the appropriate CDvsCTRL dataset.

Code used to perform the principal component analysis (representative case):

```
Xsc = scale(x, center = FALSE, scale = TRUE) #Scaling
```

```
Xpca = prcomp(Xsc) #PCA
s = summary(Xpca)
cumVar = s$importance[3,] * 100 #Cumulative Variance
cumVar
Xscores = Xpca$x #PC Scores
```

**Classification ensemble with classyfire**

The "classyfire" package was used to build a classification ensemble for the given dataset using the `cfbuild` function. (Djoumbou Feunang *et al.*, 2016). This function takes as its argument a given classified dataset and constructs support vector machine (SVM) classification ensembles. The `cfbuild` function was used which constructs the SVM classification ensembles. Using this function, the optimization of the SVM is achieved by splitting the data into training and testing subsets using bootstrapping method. According to the software developers, bootstrapping has to be repeated several times for better optimization, for this reason the default number of bootstrap iterations (100) was chosen for every classification procedure. The number of classifiers that form the classification ensemble was set to 100 (developers suggestion) but at the end of each run it was checked that this number (ensNum) was higher than the number of iteration that a stable average classification rate emerged to confirm the accuracy and reliability of the model. For each particular dataset, the following parameters were chosen:

Table 4. Parameters selected for building the classification ensembles.

| Classes | Provided data | Bootstrap Iterations | The number of classifiers that form the classification ensemble |
|---|---|---|---|
| Urine CDvsCTRL | PCA Scores PC1 – PC9 | 100 | 100 |
| Urine CDvsAll | PCA Scores PC1 – PC4 | 100 | 100 |
| Blood CDvsCTRL | PCA Scores PC1 – PC6 | 100 | 100 |
| Breath CDvsALL | PCA Scores PC1 – PC5 | 100 | 100 |

For each classification, the average accuracy, the accuracies histogram and the confusion matrix were used to assess the classification performance of each sample type. Code used to build the classifier and assess the results (representative case):

```
ens = cfBuild(Xscores[,1:9],y,bootNum = 100, ensNum = 100, cpus = 8) #Building the SVM
#ensembles with first 9 PCs
getAvgAcc(ens) #Average accuracy of the classification
getConfMatr(ens) #ConfusionMatrix
ggClassPred(ens, displayAll = TRUE, fillBrewer = TRUE, showText = TRUE) #graph
#Confusion Matrix
ggEnsTrend(ens, ylim=c(60,80)) #
ggEnsHist(ens, mean = TRUE, density = TRUE) #Test Accuracies histogram
```

**Permutation testing**

Permutation testing was applied on the classification ensemble of the best performing sample type, to compare the classification performance achieved for this dataset and the performance that would be achieved by random chance. The input data used for the permutation testing were the same as the classifier building. Then the distribution of averaged accuracies, one for each permutation, is extracted and compared to the accuracies distribution of the original classification to evaluate the significance of the performance of the classifier. Code used:

```
#Permutation test
A_CDvCTRL_permutation = cfPermute(Xscores[,1:9], y, permNum = 100, ensNum = 100,
       bootNum = 100, cpus = 8)
FA_CDvCTRL_permutation$avgAcc #extracting the average accuracy values
mean(FA_CDvCTRL_permutation$avgAcc) #mean

#Compare between classifier and permutation:
t.test(FA_CDvCTRL$testAcc,FA_CDvCTRL_permutation$avgAcc)

#Creating the histograms
par(mfrow=c(2,1))
perm_accs = FA_CDvCTRL_permutation$avgAcc
testaccs = getAcc(FA_CDvCTRL)
hist(testacc$Test, xlim =c(30,90), col = "steelblue",
     xlab = "Accuracies", ylab = "Count", main = "Ensemble")
abline(v=mean(testacc$Test), col = "red")
hist(perm_accs, xlim =c(30,90),col = "lightcoral", xlab = "Accuracies",
     ylab = "Count", main = "Permutation")
abline(v=mean(perm_accs$avgAcc), col = "red")
```

## Results & Discussion

As can be deducted from the diagnostic plots of all classifications done (**Error! Reference source not found.**), using 100 SVMs in all classification ensembles was a justifiable choice as in

all four cases the number of ensemble iteration that a stable average classification rate is emerged (Faecal = 25, Urine = 30, Breath = 50, Blood = 40) is lower than 100. Comparing the average accuracies between the different datasets it seems that the higher accuracy was achieved from faecal samples (average accuracy = 68.38%, while the other datasets achieved lower scores (Urine = 59.25%, Breath = 65.56%, Blood = 54.28%), indicating that with faecal samples a higher ratio of correct predictions to the total predictions made was achieved. Similarly, 72% of the feacal CD samples were actually predicted as CD samples according to the plot that shows the per class accuracies (%) for all the correctly classified and misclassified samples. The corresponding percentages for urine, breath and blood samples were 40%, 0% and 49%, respectively. The huge contrast between the average accuracy (65.56&) and correctly classified CD samples (0%!) that breath samples showed is probably attributed to the imbalance between the class frequencies of this dataset (Table 2). Thus, despite the good results for the majority class (Control - 66% True Positives), the minority class (CD) results are poor, a situation that increases the average accuracy of the classification without the results being accurate. These results indicate that based on the given dataset faecal samples have the greatest ability to diagnose CD. Blood and Urine samples are ranked $2^{nd}$ and $3^{rd}$ respectively with similar results, while breath samples showed a significant inability to predict CD, something which has to be further tested due to the inappropriate breath samples dataset used here.

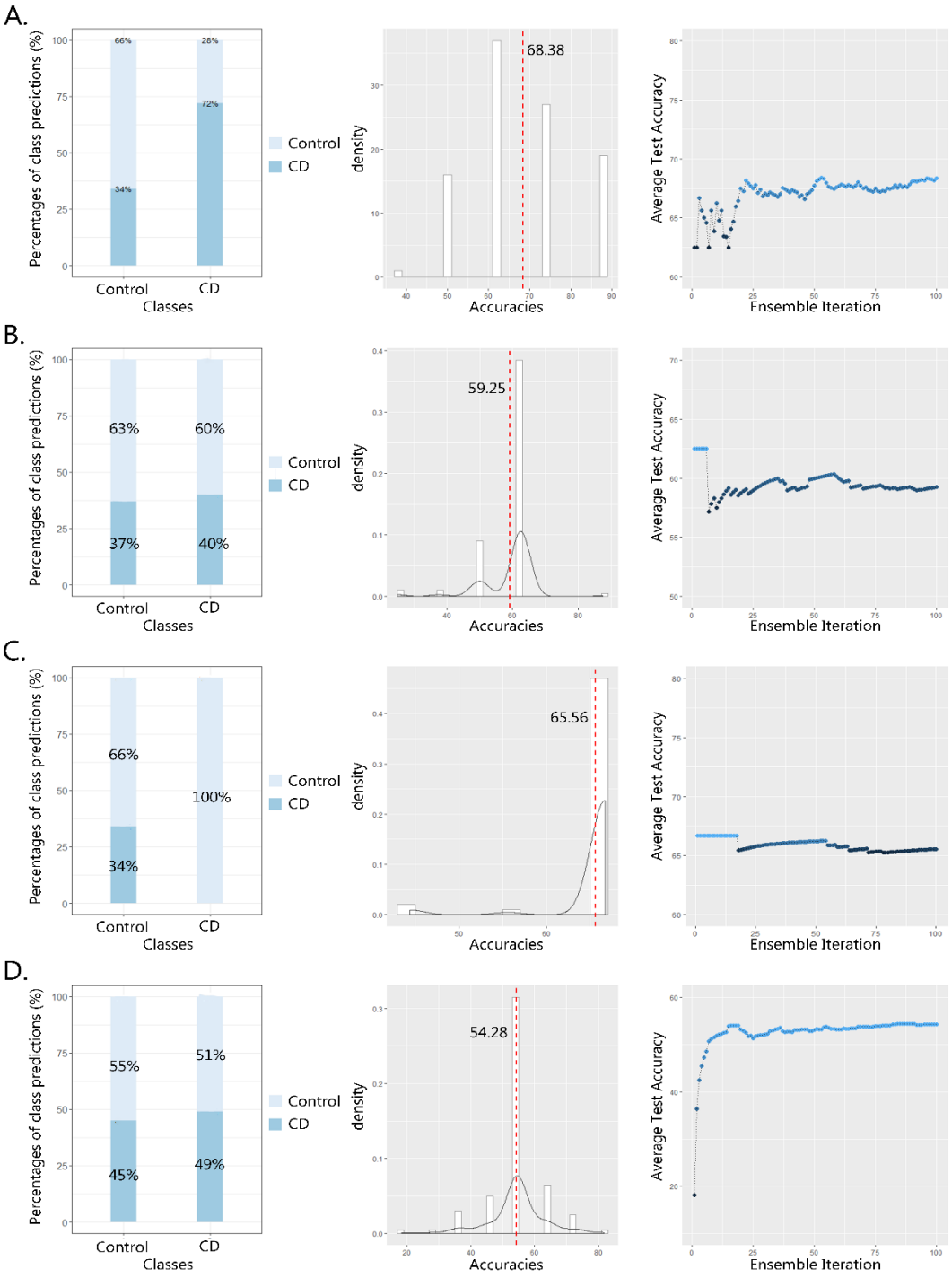


Figure 6. Performance evaluation of the classification. Diagnostic plots for comparing between the classification performance using different data as input (A. Faecal samples, B. Urine samples, C. Breath samples, D. Blood samples). Three plots are depicted for each dataset. Left plot: Per class accuracies (%) for all the correctly classified and misclassified samples. Center plot: Histogram of the test accuracies for all the classifiers within the ensemble. Mean (red line) represents the average accuracy. Right plot: average test accuracies for every new classifier added to the model.

To confirm that faecal samples indeed have a significant ability to predict CD, a permutation test was ran. Indeed, the average accuracies of the permutation tests were significantly lower than the test accuracies of the ensembles (Welch Two Sample t-test, classification avg. accuracy mean = 68.38, permutation mean = 50.84 $p < 2.2e\text{-}16$) (Figure 7).

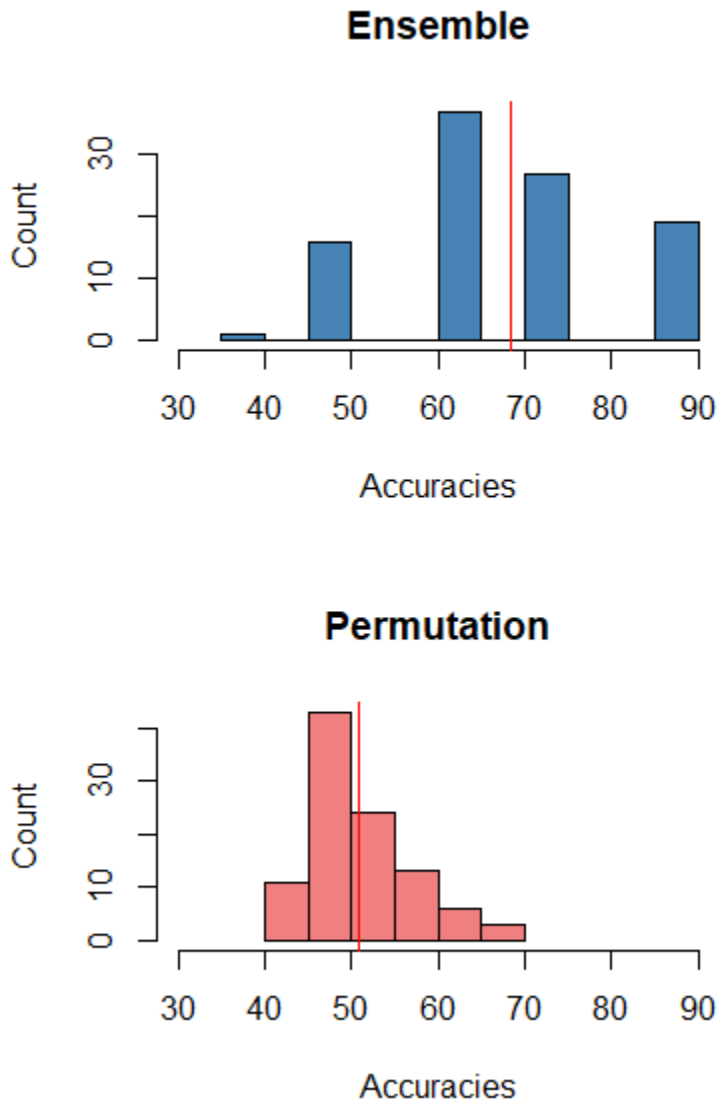


Figure 7. Histograms illustrating the distribution of ensemble test accuracies and Permutation tests average accuracies. Red line: Mean value of the distribution.

# Conclusion

In the above analysis four sample types where ranked for their ability to predict Crohn's disease patients from healthy individuals. Faecal samples showed the most reliable results. Urine and Blood samples were not that reliable, while Breath samples did not manage to make a single correct prediction. Crohn's disease is a gastrointestinal chronic disease and therefore it is reasonable for the quantity and type of proteins of feacal samples to have a diagnostic power on the disease, something which could possibly explain the above results. A higher ability of CD prediction was expected for blood and urine because both of these two sample types have repeatedly show a significant diagnostic power on many diseases. However, based on these results these expectations were not met either because these samples are indeed biologically unrelated to the disease or because more data need to be collected. The results regarding the predictive ability of Breath samples to CD cannot be evaluated due to the imbalanced class frequencies of this dataset which led to such poor results. In all cases, it is more than obvious that more data have to be collected in order for more reliable and accurate classifications to be applied.

# Question 3

## Introduction

One of the main problems in proteomics is the possibility of finding false negative peptides during peptide spectrum matching procedures. For this reason, it is a common practice for researchers to perform a decoy database search along with the standard peptide database search to identify peptides based on the MS/MS spectra file provided. The most common decoy databases contain all the peptide sequences reversed. Subsequently, the resulted peptide spectrum matches (PSM) from the "standard" database search are of unknown quality, while the PSMs from the decoy database search are definitely non-correct. The resulted PSMs from these two searches are sorted by their score, thus a score threshold can be set in order to obtain PSMs with high quality. False discovery rate (FDR) is the ratio between the false PSMs and the total number of PSMs above this set threshold and therefore, it can provide a measure of peptide matching performance. In the following analysis, LC-MS/MS data from a published study was used (Elias and Gygi, 2010) in order for the effect of the FDR threshold on the results of protein identification to be investigated.

## Methods

The following data analysis was performed on Galaxy Integrated Omics (GIO) software system (Fan *et al.*, 2015) which is a proteomics-adjusted version of Galaxy web-based platform (Afgan *et al.*, 2016).

To investigate the effect of the FDR threshold on the results of protein identification 4 different protein identification pipelines were followed:

Table 5. Protein identification procedures that were followed.

| Pipeline no. | Decoy database search | % FDR threshold |
|---|---|---|
| 1 | - | No filtering applied |
| 2 | ✓ | 5 |
| 3 | ✓ | 1 |
| 4 | ✓ | 0 |

**Data**

The LC-MS/MS data for this analysis were provided as an MGF mascot generic format file, which contains multiple MS/MS spectra in a single file, encoded via precursor mass, charge and m/z (Deutsch, 2012).

The database that was used for the peptide searching includes the Chinese Hamster reference proteome and a dataset of common contaminants both in fasta file format. These two datasets were concatenated together using the "Concatenate dataset" tool of the Galaxy.

**MS-GF+ Search**

MS-GF+ is a sensitive and universal MS/MS database search tool which is included in GIO and it was used to find peptide spectrum matches by scoring the MS/MS spectra from the MGF file against peptides derived from the given protein sequence database. The other parameters that were used for the database search were similar for all 4 identification procedures and they were determined according to the Mass spectrometry experimental procedure that was followed by the researchers. According to the study:

- Protein Modifications: Provide information about the post translational modifications of the proteins for the mass of the modified residues to be appropriately adjusted.
  - Fixed Modification: Carbamidomethylation of cysteine (+57.0214)
  - Variable Modifications: oxidation of methionine (+15.9949), SILAC labels ($^{13}C_6$-Lys, $^{13}C_6$-Arg, $^{13}C_6$ $^{15}N_2$-Lys and $^{13}C_6$ $^{15}N_4$-Arg)
- Search type
  - Decoy Search: Yes or No. It depends on which of the protein identification pipelines were followed.

- Digestion of proteins
  - Digestion Enzyme: Trypsin
  - Protocol: No protocol
- Mass spectrometer parameters
  - Instrument Type: High-res LTQ. LTQ-Orbitrap Velos mass spectrometer (Thermo Scientific) was used.
  - Precursor ion tolerance: 10.0 ppm

**False Discovery Rate**

False Discovery Rate tool of the mzidlib software was used to calculate FDR. More specifically this tool takes as input the MZID file created from the MSGF+ database search that includes a decoy database search and calculates Local FDR, Q-value and FDRScore for each PSM. Parameters used:

- FDR level: Peptide Spectrum Match. The above described scores will be calculated for each PSM.
- Regular expression indicating decoy peptides: MS-GF+ ("XXX_"). This is for the software to recognize the peptide IDs in the MZID file.
- Decoy Ratio Value: 1. Decoy database to target database ratio.
- Specification of the score that will be used for the PSMs ordering:
  - CV term of score to use for FDR calculation: Original search engine score.
  - CV term: E-Value: This value represents the number of times that we would expect to obtain an equal or higher score purely by chance. This scoring system was used to order the PSMs.
- Are better scores lower?: Yes. The answer is obvious by the definition of the Evalue.

**Filtering the PSMs**

This step sets a threshold above which all the PSMs are indicated as high-quality and can be further used for protein identification. Threshold tool of the mzidlib software is included in GIO platform that perform this task was used by taking as its input the MZID file created at the previous step. Parameters that used in this tool.

- Thresholding Level: PSM. FDR score has been calculated for the different peptide spectrum matches after the previous step (Post processing score).
- Specification of the score that will be used for thresholding. The parameters used here should match the parameters that used for the FDR score calculation.
    - Score Type: Post processing score
    - Post processing score to use for thresholding: Distinct peptide-level FDR (MS:1002360)
- Treshold value: e.g. 0.05. The threshold is set. This value represents the ratio between the false PSMs and the total number of PSMs above the set threshold. For example a threshold value equals to 0.05 (5%) means that the 5% of the total PSMs that are indicated as high-quality and can be further used for protein identification are false PSMs.
- Are better scores lower?: Yes. E-value.

**Protein Identification**

This is the final step that will assign the PSMs into proteins. For this task, another mzidlib tool is used (ProteoGrouper), which performs a sequence-based protein inference based on a set of PSMs (PAG) by taking as its input the MZID file created at the previous step. Parameters used:

- CV term for the PSM score to create a protein score: Distinct peptide-level FDR (MS:1002360). Like the previous step.
- Log tranform scores: Yes. Because E-value has been used.

The final MZID file that is produced by this procedure can be visualized with the ProViewer software that is available in GIO.

**GIO Workflow**

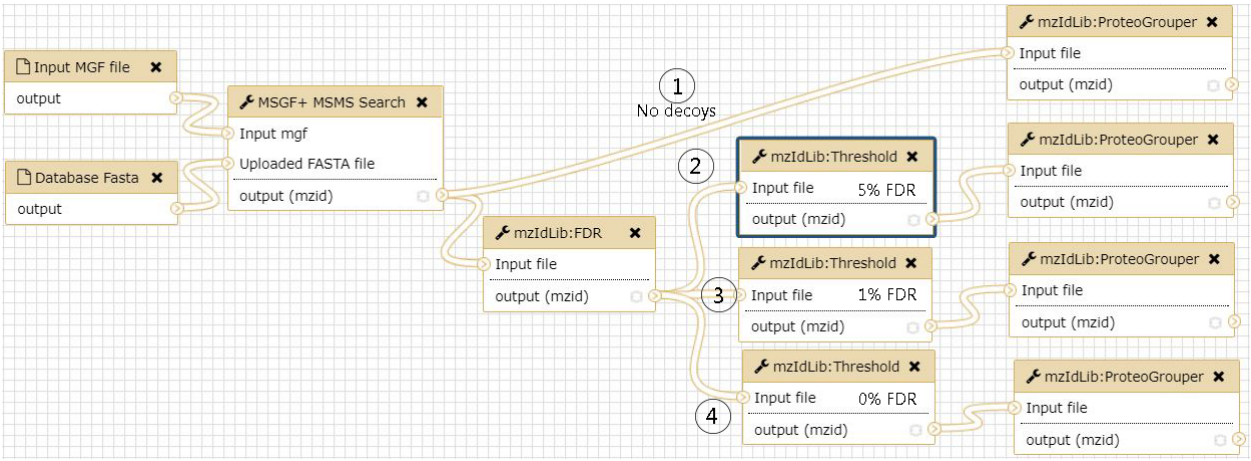


Figure 8. Schematic illustration of the Workflow used to perform the described analysis in GIO. Circled numbers refer to the pipeline that was used for protein identification according to Table 4.

## Results – Discussion

Table 6. Numbers of distinct peptides and proteins identified through each of the pipelines that were followed. The numbers inside the brackets represent the percentage of decrease from the first pipeline's results (identified peptides or proteins).

| Pipeline | Distinct peptide IDs | Identified Proteins |
|---|---|---|
| Simple MSGF+ search (no decoys) | 59757 | 12227 |
| MSGF+ search at 5% FDR | 11053 (-81.53%) | 1981 (-83.79%) |
| MSGF+ search at 1% FDR | 9094 (-84.78%) | 1248 (-89.79%) |
| MSGF+ search at 0 % FDR | 2665 (-95.54%) | 591 (-95.17%) |

As expected, the implication of different FDR thresholds on the number of peptides and proteins identified from these pipelines. According to the table below (Table 6), the search that was conducted without a decoy database search returned 59757 peptides, a number that represents the total number of matches, false peptides included. Applying a 5% FDR threshold, the number of PSMs was dramatically reduced to 11053 (decreased by 81.53% compared to no decoys search), since only the 5% of this number now represents false peptides. Reasonably, the number of peptides was further reduced to 9094 by applying a 1% FDR threshold (decreased by 84.78% compared to no decoys search), as only the 1% of this number now represents false

PSMs. Applying the strictest possible FDR threshold 0% led to a sharp decrease of the PSMs to only 2665 (95.54% decrease compared to no decoys search), since now this number does not include any peptides identified, therefore it contains only the high-quality PSMs.

These results considerably affected the number of the identified proteins. Significantly, the rate of reduction of identified proteins is proportional to the rate of reduction of identified peptides (from pipeline 1 to 4).

## Conclusion

The credibility of the identification of peptides from MS/MS data is of great importance in proteomics. The above analysis showed the huge impact that FDR threshold has on the number of identified peptides and proteins from MS/MS spectra data using a typical proteomics pipeline, something which was expected based on the understanding of how this filtering method works. Reasonably, this huge impact of the FDR threshold on the identification of peptides and proteins could lead to wrong interpretation of an experiments conclusions, since only minor changes of the FDR threshold could alter the result. Furthermore, this also reveals the danger of losing information, as a strict FDR threshold could lead to exclusion of true peptide identification with low scoring. A solution to this problem could be given by increasing the PSM scores initially, something that can be done by minimizing the database size as much as possible. Moreover, researchers should test peptide/protein identification pipelines several times under implicating different FDR-scores each time and investigating the identified proteins to find which FDR threshold leads to more proteins that could be reasonably included in the initial samples if this information is available. Researchers could also further use these different sets of identified proteins for further study and compare between their final results to decide if the different FDR thresholds significantly affected their final biological result.

# Question 4

## Introduction

Biological networks represent visualizations of connections that can be found in different fields of biology such as proteomics (Bensimon, Heck and Aebersold, 2012). Depictions of networks consisting of proteins that modifying each other are very popular and useful for extracting biologically significant information. Kinases are proteins which have the ability to modify other protein targets by phosphorylizing them in certain sites called phosphosites (Manning *et al.*, 2002). The aim of the following analysis is to build and visualise a kinase signalling network in order to investigate how single point mutations in critical residues of these kinases can alter this network. The data used to create this network was a list of proteins identified from a sample using LC-MS/MS.

## Methods

The following data analysis was performed using Python 2.7. All the code that used for this analysis is included as an appendix at the end of this question. All the information about the under-investigation proteins was found on UniProt database (The Uniprot consortium, 2017). Programmatic access to the UniProt database was achieved through its application programming interfaces (APIs). The interaction network was visualized using Cytoscape software (Shannon *et al.*, 2003).

**Loading and using the protein list containing CSV file**

The protein-containing CSV file was loaded in python and the names of its proteins with two or more peptides were converted and inserted into a python list. These protein names were then converted into UniProt accession IDs using UniProt's special Python code for this conversion (UniProt API).

**Using the protein list to extract kinase-phosphosite relationships from UniProt**

Uniprot database was searched for each of these accession IDs using Uniprot's API. The output for each accession number was a JSON formatted text only with the residue modifications of the corresponding protein. Request URL:

"https://www.ebi.ac.uk/proteins/api/features?offset=0&size=100&accession="+PROTEIN ACCESSION+"&types=MOD_RES"

Then, the name of the modified residue, the position of this residue and the kinases that phosphorylates it were captured using regular expressions on this JSON text and stored into variables, while the name of the protein was initially stored in a variable. The kinase names were converted into (XXXXX_HUMAN) format. In autocatalysis cases, kinase name was by the target name. This information was used to create a new CSV file with the following format:

| | | | | |
|---|---|---|---|---|
| 1 | H2A2C_HUMAN | T | 121 | DCAF1_HUMAN |
| 2 | CD44_HUMAN | S | 672 | PKC_HUMAN |
| 3 | TENA_HUMAN | S | 72 | FAM20C_HUMAN |
| 4 | T132A_HUMAN | S | 529 | FAM20C_HUMAN |
| 5 | TGON2_HUMAN | S | 71 | FAM20C_HUMAN |

Because this file had some kinases that were not appropriately separated (e.g. PKB/AKT1 instead of PKB AKT1), while some kinases had autocatalysis as their name, the new CSV file was loaded again and these errors were corrected.

**Filtering the targets**

The aim of this analysis was the construction of a kinase network, in which the interactions between kinases are investigated. Therefore, if the target protein was not in the kinases list the information about this target was excluded from this analysis. Furthermore, if an interaction was appeared more than one times in the file the duplicates were also excluded.

**Creating the SIF file – Visualising the network**

This CSV file was converted into a format which is commonly used for importing interactions when building a network. The format of this file is of this type [mofidier] [effect on] [target]:

```
LYN_HUMAN phosphorylates KSYK_HUMAN
KSYK_HUMAN phosphorylates KSYK_HUMAN
ERBB2_HUMAN phosphorylates ERBB2_HUMAN
AMPK_HUMAN phosphorylates ULK1_HUMAN
MTOR_HUMAN phosphorylates ULK1_HUMAN
```

This file was imported into Cytoscape in which the appropriate settings were chosen in order for the software to interpret which names represent the modifiers and which the targets. Graphical options were also altered to improve the network visualization.

## Results & Discussion

The visualization of these interactions revealed a complex network with many nodes interacting with each other (Figure 9). As it was expected, the majority of the network kinases form a complex network based on their phosphorylation-interactions. However, there is a small group of kinases which do not interact with other molecules apart from their self (autocatalysis).

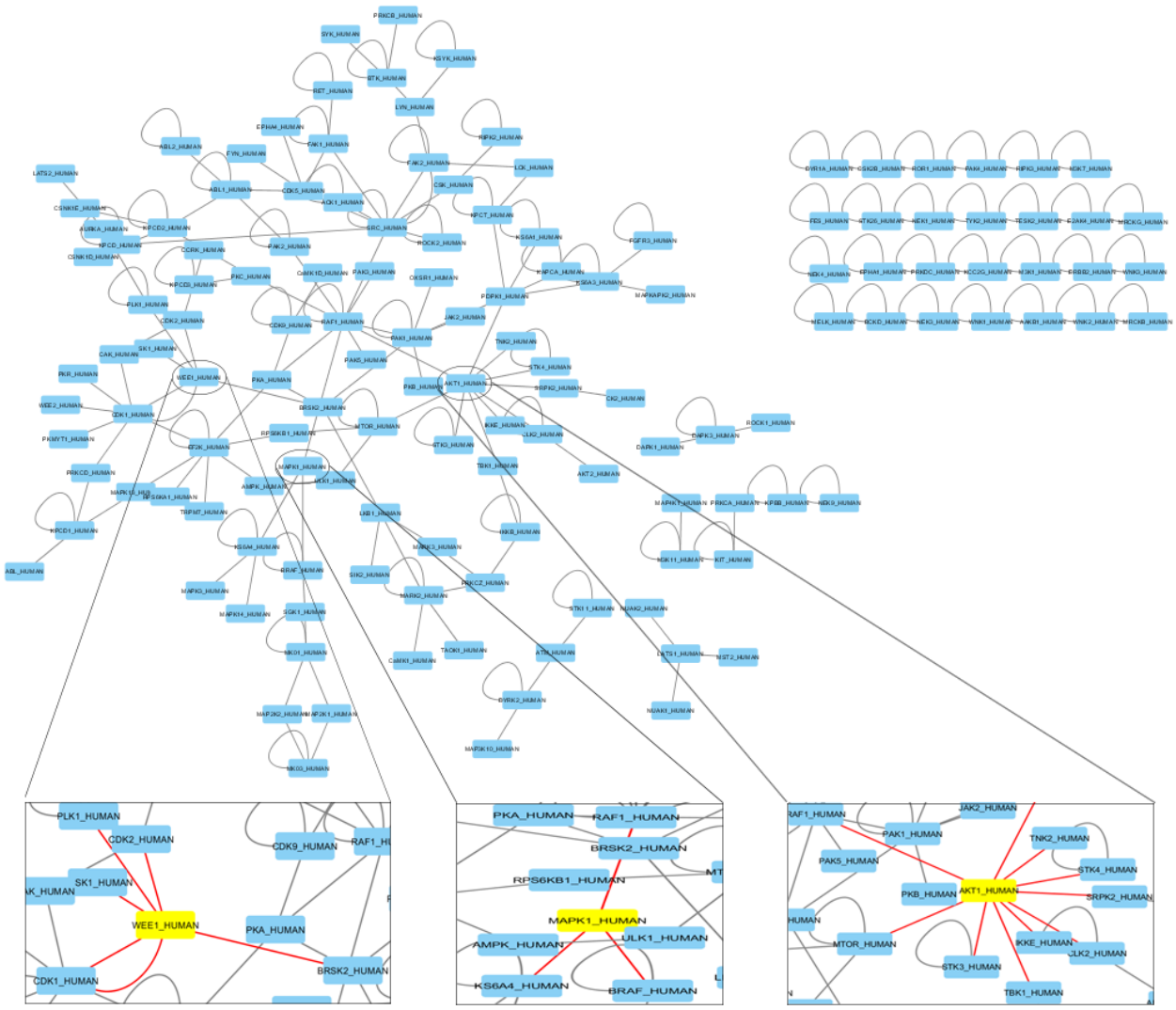


Figure 9. Visualization of the produced human kinase signalling network. Grey lines represent phosphorylation modifications with the direction from the modifier to the target. Blue nodes represent the kinases of the network. The three bottom figures showing zoomed in sections of the network. Bottom left figure: Interactions of the WEE1 kinase, Bottom middle figure: Interactions of the MAPK1, Bottom right figure: Interactions of the AKT1 kinase.

Information from the above constructed network and UniProt database were combined to investigate possible network alterations due to residual changes of the kinases.

According to UniProt database, WEE1 kinase is phosphorylated on its 642$^{nd}$ residue (serine) by BRSK1 and BRSK2. A mutation that causes this residue to change from serine to glutamine will make this interaction impossible as glutamine residues cannot be phosphorylated, however the catalytic activity of this molecule will remain unaltered. In the above network, this

modification will cause the exclusion of the BRSK1 as it does not interact with any other molecule in the network. Contrariwise, WEE1 kinase and BRSK2 will remain to the network as they both interact with other kinases (WEE1 is phosphorylated by PLK1, CDK1 in other residues and phosphorylates CDK1 and CDK2, BRSK2 is phosphorylated by LKB1 and PKA and phosphorylates PAK).

MAPK1, on the other hand, appears to have only modifier role in this network, since it is not being phosphorylated. However, this molecule phosphorylates BRAF, KS6A4 and RAF1. Therefore, a mutation that leads to the inhibition of its activity will cause the exclusion of this kinases from the network. In this case, BRAF kinase will remain in the network as it performs autocatalysis while it is also being phosphorylated by SGK1. Similarly KS6A4 and RAF1 will remain in the network as they both being phosphorylated by other molecules (KS6A4: autocatalysis, MAPK3, MAPK14 and RAF1: PKA, PKC, AKT1, autocatalysis, PAK1, PAK2, PAK3, PAK5, SRC).

A change of the 474$^{th}$ residue of the AKT1 from tyrosine to phenylalanine due to a mutation will probably not cause any changes in the network. Although, this residue is found to be phosphorylated in AKT1 there are no further information about the kinase that mediates this modification and therefore is not included in this network. Moreover, a change in this residue will not cause any major changes in the catalytic function of this molecule.

Using the distribution tool of the Cytoscape software after selecting a tree view of the network the kinases were distributed between the two most distant nodes of the network based on the number of interactions needed to get from one to another. These two kinases are CSNK1D and AKT1.

## Conclusion

The above analysis results revealed how complex the kinases network could be in a biological sample, while it became clear that a kinase can interact with an another kinase not only directly but also indirectly through very complex pathways. Furthermore, it was also revealed that even a point mutation in the gene of a kinase which participates in a kinase signalling network could have a big impact not only on its own role in the network but can also affect a significant number of other molecules. Taking all of the above into consideration, it is

obvious that understanding the function of these molecules or even trying to block/enhance their activity (drug discovery) could be very difficult, especially when the knowledge about the examined sample is limited.

# Appendix

## *Figures*

Figure 10. The kinase signaling network is depicted in a distributed tree layout to find the two most distant nodes.

## *Code*

```
# coding: utf-8

#load the identified proteins csv file
proteins = open("protein_list.csv", "r")
proteins.readline()
protein_names=[]
names_peptides={}
for line in proteins:
    splitted = line.split(",")
    names_peptides[splitted[1]]=int(splitted[2])
for ele in names_peptides:
    if names_peptides[ele] >= 2:
        protein_names.append(ele) #protein names

#Convert the names to UniProt accession ID.
import urllib,urllib2
import requests, sys, re
text1=" "
text1=" "+"Target"+" "+"Position"+" "+"Residue"+" "+"Kinase\n"
accessions=[]
counter = 0
for name in protein_names:
    url = 'http://www.uniprot.org/uploadlists/'

    params = {
    'from':'ID',
    'to':'ACC',
    'format':'tab',
    'query':str(name)
    }

    data = urllib.urlencode(params)
```

```
request = urllib2.Request(url, data)
    contact = "" # Please set your email address here to help us debug in case of problems.
    request.add_header('User-Agent', 'Python %s' % contact)
    response = urllib2.urlopen(request)
    page = response.read(200000)

    try:
        protein_accession = page.split("\t")[10] #Protein accession
        #print protein_accession
    except IndexError: #avoid UniProt code errors
        continue

    #get MOD_RES for each search in json format
    requestURL                                                                                                                                    =
"https://www.ebi.ac.uk/proteins/api/features?offset=0&size=100&accession="+str(protein_accession)+"&types=MOD_RES"

    r = requests.get(requestURL, headers={ "Accept" : "application/json"})

    if not r.ok:
      r.raise_for_status()
      sys.exit()

    responseBody = r.text #responce in json format
    #regex to capture phosphorylation targets, kinases, position, residue
    regex = r"Phospho\w+\;\W+by\W\w+/?\w+((,\W+\w+)+)?(\Wand\W+\w+/?\w+)?\",\W+begin\W+\d+\","
    matches = re.finditer(regex, responseBody)
    #Extracting the information from the regex
    for matchNum, match in enumerate(matches):
        matchNum = matchNum + 1
        pattern = ("{start}-{end}: {match}".format(matchNum = matchNum, start = match.start(), end = match.end(), match =
match.group()))
        residue = pattern.split(":")[1].split(";")[0][8:]
        position = pattern.split(":")[2].split('"')[1]
        kinases = pattern.split('"')[0].split("by")[1].split("and")[0].split(",")
        residue = str(residue)
        if residue == "serine":
            residue = "S"
        elif residue == "threonine":
            residue = "T"
        elif residue == "tyrosine":
            residue = "Y"
        position = str(position)
        target = str(name)
        try:
            and_kinases = pattern.split('"')[0].split("by")[1].split("and")[1]
            kinases.append(and_kinases)
            for kin in kinases:
                if str(kin) == " autocatalysis":
                    text1+=str(counter)+" "+str(target)+" "+residue+" "+position+" "+str(target)+"_HUMAN"+"\n"
                    counter+=1
                else:
                    text1+=str(counter)+" "+str(target)+" "+residue+" "+position+" "+str(kin[1:].strip())+"_HUMAN"+"\n"
                    counter+=1

        except IndexError: #If a target has only one kinase
            for kin in kinases:
                if str(kin) == " autocatalysis":
                    text1+=str(counter)+" "+str(target)+" "+residue+" "+position+" "+str(target)+"_HUMAN"+"\n"
                    counter+=1
                else:
                    text1+=str(counter)+" "+str(target)+" "+residue+" "+position+" "+str(kin[1:].strip())+"_HUMAN"+"\n"
                    counter+=1
            continue
            #text 1 is a string containing the information needed in this layout:
            #1 H2A2C_HUMAN T 121 DCAF1_HUMAN\n
            #...

#write text1 to a new csv file
import StringIO
s = StringIO.StringIO(text1)
with open('protein_kinase_mods.csv', 'w') as f:
    for line in s:
        f.write(line)

#The previously saved document is loaded again
#to fix some issues and create the final csv
```

```
#file
FILE = open("protein_kinase_mods", "r")
header = FILE.readline()
targets = []
kinases = []
targets_2 = []
kinases_2 = []

#Targets list - Kinases list
for line in FILE:
    splitted = line.split("\t")
    targets.append(splitted[1].strip()) #Targets
    kinases.append(splitted[4].strip()) #Kinases
count=0
indexlist=[]

#error fix: Some kinases have name autocatalysis - change to target name
for i in range(0,len(kinases)):
    if kinases[i]=="autocatalysis_HUMAN":
        kinases[i]=targets[i]

#error fix: separate kinases with /
for i in range(0,len(kinases)):
    if "/" in kinases[i]:
        targets_2.append(targets[i])
        kinases_2.append(kinases[i].split("/")[0]+"_HUMAN")
        kinases[i] = kinases[i].split("/")[1]

    else:
        continue
#splitted kinases are added to the list
kinases.append(kinases_2)
targets.append(targets_2)

#Targets filtering
#If they are not in kinases list
#kinase and target for this index are removed
for i in range(0,len(targets)):
    if targets[i] not in kinases:
        targets[i]=" "
        kinases[i]=" "
while " " in targets:
    targets.remove(" ")
while " " in kinases:
    kinases.remove(" ")

#csv_row: the string that will be written in the csv file
    csv_row=""
for j in range(0,len(kinases)):
    csv_row += str(kinases[j])+" "+"phosphorylates"+" "+str(targets[j])+"\n"
#write to the file
import StringIO
s = StringIO.StringIO(csv_row)
with open('final.csv', 'w') as f:
    for line in s:
        f.write(line)
inFile = open('final.csv','r')
outFile = open('final_no_dups.csv','w')
listLines = []
for line in inFile:
    if line in listLines:
        continue
    else:
        outFile.write(line)
        listLines.append(line)
outFile.close()
inFile.close()

#load the saved file again and remove duplicates
inFile = open('final.csv','r')
outFile = open('final_no_dups.csv','w')
listLines = []
for line in inFile:
    if line in listLines:
        continue
    else:
```

```
        outFile.write(line)
        listLines.append(line)
outFile.close()
inFile.close()
```

# References


Afgan, E. *et al.* (2016) 'The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update', *Nucleic Acids Research*. Oxford University Press, 44(W1), pp. W3–W10. doi: 10.1093/nar/gkw343.

Bengtsson, H. (2016) *R.matlab: Read and Write MAT Files and Call MATLAB from Within R*. Available at: https://github.com/HenrikBengtsson/R.matlab (Accessed: 12 December 2017).

Bensimon, A., Heck, A. J. R. and Aebersold, R. (2012) 'Mass Spectrometry–Based Proteomics and Network Biology', *Annual Review of Biochemistry*, 81(1), pp. 379–405. doi: 10.1146/annurev-biochem-072909-100424.

Bhatt, S. *et al.* (2013) 'The global distribution and burden of dengue', *Nature*, 496(7446), pp. 504–507. doi: 10.1038/nature12060.

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

Deutsch, E. W. (2012) 'File formats commonly used in mass spectrometry proteomics.', *Molecular & cellular proteomics : MCP*. American Society for Biochemistry and Molecular Biology, 11(12), pp. 1612–21. doi: 10.1074/mcp.R112.019695.

Djoumbou Feunang, Y. *et al.* (2016) 'ClassyFire: automated chemical classification with a comprehensive, computable taxonomy.', *Journal of cheminformatics*. Springer, 8, p. 61. doi: 10.1186/s13321-016-0174-y.

Edgar, R., Domrachev, M. and Lash, A. E. (2002) 'Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.', *Nucleic acids research*, 30(1), pp. 207–10. Available at: http://www.ncbi.nlm.nih.gov/pubmed/11752295 (Accessed: 12 December 2017).

Elias, J. E. and Gygi, S. P. (2010) 'Target-decoy search strategy for mass spectrometry-based proteomics.', *Methods in molecular biology (Clifton, N.J.)*. NIH Public Access, 604, pp. 55–71. doi: 10.1007/978-1-60761-444-9_5.

Fan, J. *et al.* (2015) 'Galaxy Integrated Omics: Web-based Standards-Compliant Workflows for Proteomics Informed by Transcriptomics', *Molecular & Cellular Proteomics*, 14(11), pp. 3087–3093. doi: 10.1074/mcp.O115.048777.

Galili, T. (2015) 'dendextend: an R package for visualizing, adjusting and comparing trees of hierarchical clustering', *Bioinformatics*. Oxford University Press, 31(22), pp. 3718–3720. doi: 10.1093/bioinformatics/btv428.

Kwissa, M. *et al.* (2014) 'Dengue virus infection induces expansion of a CD14(+)CD16(+) monocyte population that stimulates plasmablast differentiation.', *Cell host & microbe*, 16(1), pp. 115–27. doi: 10.1016/j.chom.2014.06.001.

M'Koma, A. E. (2013) 'Inflammatory bowel disease: an expanding global health problem.', *Clinical medicine insights. Gastroenterology*. SAGE Publications, 6, pp. 33–47. doi: 10.4137/CGast.S12731.

Manning, G. *et al.* (2002) 'The protein kinase complement of the human genome.', *Science (New York, N.Y.)*. American Association for the Advancement of Science, 298(5600), pp. 1912–34. doi: 10.1126/science.1075762.

Ritchie, M. E. *et al.* (2015) 'limma powers differential expression analyses for RNA-sequencing and microarray studies', *Nucleic Acids Research*, 43(7), pp. e47–e47. doi: 10.1093/nar/gkv007.

RStudio Team (2015) 'RStudio: Integrated Development Environment for R'. RStudio, Inc., Boston, MA. Available at: http://www.rstudio.com/. (Accessed: 12 December 2017).

Shannon, P. *et al.* (2003) 'Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks', *Genome Research*, 13(11), pp. 2498–2504. doi: 10.1101/gr.1239303.

Szklarczyk, D. *et al.* (2016) 'STITCH 5: augmenting protein–chemical interaction networks with tissue and affinity data', *Nucleic Acids Research*, 44(D1), pp. D380–D384. doi: 10.1093/nar/gkv1277.

‘UniProt: the universal protein knowledgebase’ (2017) *Nucleic Acids Research*. Oxford University Press, 45(D1), pp. D158–D169. doi: 10.1093/nar/gkw1099.