Connecting Nutrition, Health and Environment

Wiktor Jurkowski, Perla Troncoso Rey, Earlham Institute

25 - 26 January 2017

Contents

List of Figures

List of Tables

1 Introduction

In this practice, we will look at how to explore gene expression data (which could be extracted from microarray or RNA sequencing data). In the first part of this practical session we will see general techniques to explore the patterns or structure of the data using Principal Component Analysis, PCA, and Hierarchical Clustering. We will then look into compiling an interaction network using an online resource and how to visualise the network using Cytoscape [2] [3].

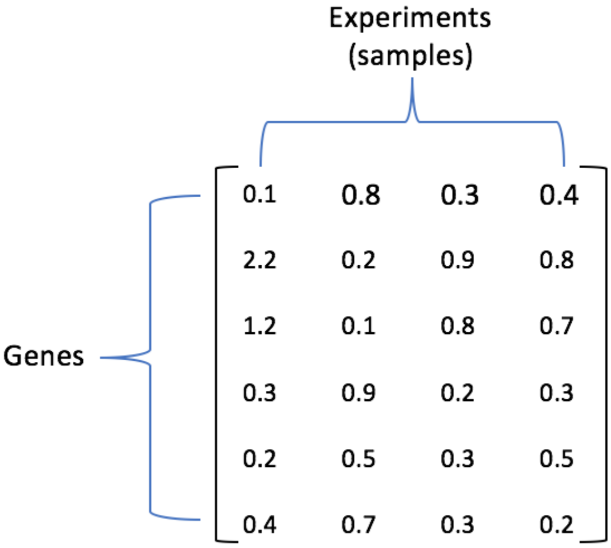
In the second part, we will look at ways to rank genes (and/or metabolites) using approaches based on logistic regression. We will then perform pathway analysis using those rankings.

We will use a publicly available data from a study on fatty liver disease of obese and lean human subjects [7].

Part I
Exploring the expression data

One could obtain gene expression from microarrays or RNA sequencing data. It is not within the scope of this session to look at the details of obtaining quantifying the expression of genes from microarrays or RNA sequencing data but instead we will start our analysis assuming that expression data has been quantified. The gene expression data is normally stored in tabular file, representing a matrix where the columns are the samples or experiments, and the rows represent the genes.

Example of expression data is shown in Figure 1. The table shows the expression of six genes in four different experiments or samples. This is, gene A has expression of 0.1 for sample 1, 0.8 for sample 2, 0.3 for sample 3, and so forth.



$$\begin{bmatrix} 0.1 & 0.8 & 0.3 & 0.4 \\ 2.2 & 0.2 & 0.9 & 0.8 \\ 1.2 & 0.1 & 0.8 & 0.7 \\ 0.3 & 0.9 & 0.2 & 0.3 \\ 0.2 & 0.5 & 0.3 & 0.5 \\ 0.4 & 0.7 & 0.3 & 0.2 \end{bmatrix}$$

Figure 1: Example of expression data with samples across the columns and individual genes down the rows.

Expression data can be obtained using different algorithms. One of the most well know are TopHat and Cufflinks protocol for the analysis of RNA sequencing data, which includes quantification of gene expression. Table 1 shows an example of the output provided by cufflinks with the estimated gene-level expression values. Cufflinks uses the notation "XLOC_*numeric_sequence*" to identify a gene.

2 Principal Component Analysis

Principal Component Analysis, commonly known as PCA, is a mathematical technique that is used to explore data, specially high-dimensional data, to extract the most important trends in the data.

When thinking of gene expression data, high dimensionality comes from the large number of dimensions of the data. This is, the result of each experiment can be thought as a kind of space, where each each feature is a coordinate in the space. There are typically thousands of genes (dimensions) and the structure of pattern in the data extends

Table 1: Example of the output provided by cufflinks for the quantification of gene expression from RNA sequencing data.

tracking_id	sample1	sample2	sample3	sample4	sample5	sample6	sample7	sample8
XLOC_000001	35.1077	50.9662	78.7724	35.4736	69.6067	63.9241	57.7967	61.4227
XLOC_000002	49.7359	64.6178	46.8884	74.617	66.0371	42.9654	645.65	64.8351
XLOC_000003	0	0	0.937767	0	0	0	0	0
XLOC_000004	89.7196	85.5504	185.678	74.617	142.783	168.718	172.63	167.206
XLOC_000005	12.6778	39.1347	158.483	22.0181	28.5566	45.0613	15.9701	50.0481
XLOC_000006	10.7273	9.1011	10.3154	13.4555	7.13915	6.28762	7.60483	12.512
XLOC_000007	0	0	0.937767	0	0	0	0	0
XLOC_000008	55.5871	37.3145	86.2746	66.0544	66.9295	53.4448	54.7548	75.0722
XLOC_000009	37.0581	16.382	24.3819	24.4646	38.3729	15.7191	24.3355	50.0481
XLOC_000010	812.352	483.269	696.761	748.616	1094.97	521.873	675.309	741.622
XLOC_000011	0	0	0	0	0	1.04657	0.760483	1.13746

to all the dimensions.

How PCA works

The mean represents the average of the values in the data:

$$\bar{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

The variance provides the the spread of the data:

$$\mathrm{Var}(\mathbf{X}) = \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{\mathbf{X}})^2 \tag{2}$$

For example, Figure 2 shows two distributions with the same mean but different variance. This means that the data points are at the same location but with a different strength. Thus, the third statistic we'll need is the covariance.

The covariance represents the degree of co-dependence of two variables, i.e., it measures the co-dependency of two variables, given by:

$$\mathrm{Cov}(\mathbf{X}, \mathbf{Y}) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{\mathbf{X}})(y_i - \bar{\mathbf{Y}}) \tag{3}$$

Increases with increasing co-dependency and variance. Just as the variance measures the degree to which a set of data varies, the co-variance is a measure of the way two sets of data vary together.

$$\mathrm{Cov}(\mathbf{X}, \mathbf{X}) = \mathrm{Var}(\mathbf{X}) \tag{4}$$

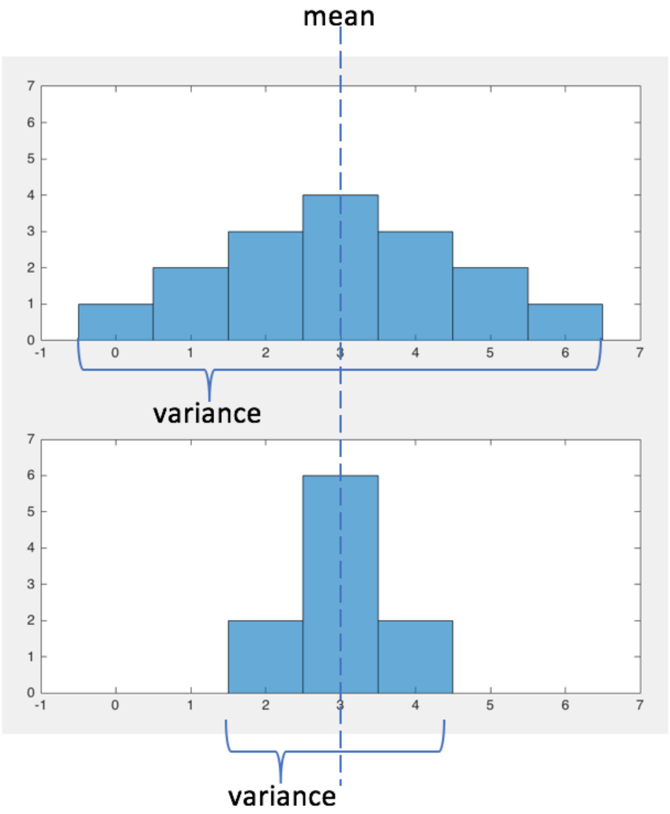


Figure 2: Two distributions with the same mean but different variance.

The covariance also increases in magnitude as the variance of each of the two datasets increases. Correlation values can be negative or positive, indicating whether the values of two variables increase or decrease together. Figure 3 shows some examples of correlation.

Coordinate transformations

In a two dimensional space described by coordinates, a point in space is described by X and Y such that $\mathbf{v} = [x_1, y_1]$. For example, the vector $v_1 = [1\ 2]^T$ represents a point in the 2 dimensional space as shown in Figure 4 (a). An alternative coordinate system described by the coordinates X$'$ and Y$'$, has a different column vector describing the same point $\mathbf{v}' = [x'_1, y'_1]$, shown in Figure 4 (b).

The two coordinate systems are $T\mathbf{v} = \mathbf{v}'$, related to the orthogonal transform matrix T. An orthogonal matrix is the kind of matrix which performs rotated-axis coordinate transforms. Thus, we can make a new coordinate system by using a transformation matrix T, which relates the two coordinates vectors by matrix multiplication. There are many types of transformations but we are particularly interested in transformations

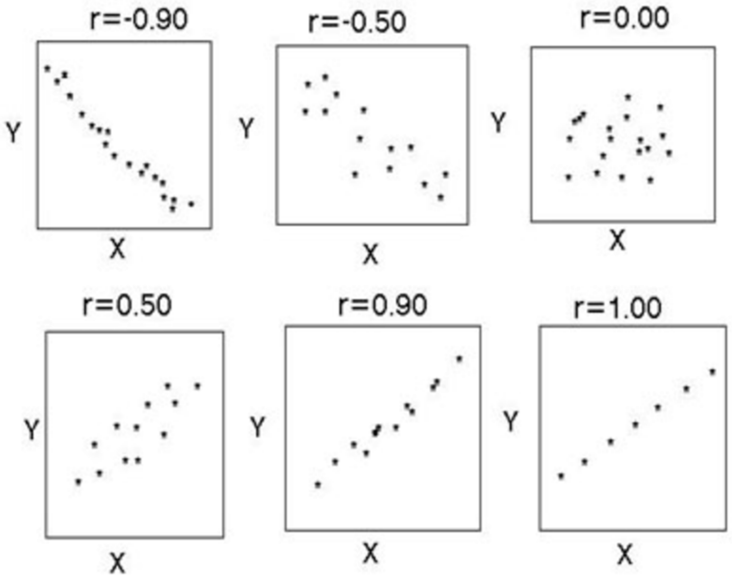


Figure 3: Examples of correlation

which rotate the coordinate axis. These are performed by matrices which have the property called orthogonality.

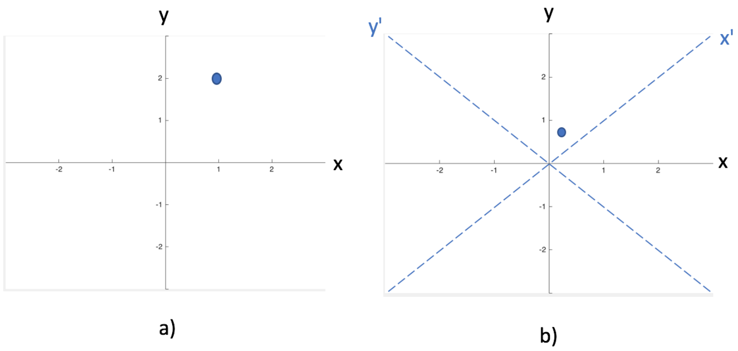


Figure 4: Example of a coordinate transform

Eigenvalues and Eigenvectors

When a transformation matrix maps a vector to a multiple of itself, then the vector is called an Eigenvector. The amount by which the vector is multiplied (stretched) is the associated Eigenvalue:

$$Tx = \lambda x \tag{5}$$

where λ are the Eigenvalues and x are the Eigenvectors.

In general terms, PCA uses covariance to encode the structure in the data and then eigenvectors to devise a new set of coordinates that best reveals the structure by finding the appropriate set of directions. One result from linear algebra is that if the eigenvectors are placed next to each other to construct an orthogonal matrix that performs a coordinate transformation. It is important to mention that Eigenvectors are placed in descending order of their corresponding eigenvalue to ensure that the first components encode most of the variance in the data. The transpose of this matrix of Eigenvectos is an orthogonal matrix which performs a rotated-axis coordinate transformation. We can transform our data matrix, D, to the new coordinates, D_{PCA}:

$$D_{PCA} = W^T D$$

For example:

The matrix: $\begin{pmatrix} 1 & 3 \\ 2 & 2 \end{pmatrix}$ has eigenvalues 4 and -1 and the eigenvectors $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 3 \\ -2 \end{pmatrix}$

such that

$$\begin{pmatrix} 1 & 3 \\ 2 & 2 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 4\begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 3 \\ 2 & 2 \end{pmatrix}\begin{pmatrix} 3 \\ -2 \end{pmatrix} = -1\begin{pmatrix} 3 \\ -2 \end{pmatrix}$$

The orthogonal matrix using these Eigenvector is:

$$W^T = \begin{bmatrix} 1 & 1 \\ 3 & -2 \end{bmatrix}$$

2.1 Exercise: PCA for the expression of two genes

We will use R and the package stats to perform PCA. We will use an example data which represents several measurements of the expression of two genes, x and y, with the following values:

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

We start by create a matrix of points in 2-d space (gene expression data) by using the following syntax:

```
#n number of samples
name_gene_1 <- c(values_in_sample1, ..., _value_in_sampleN)
#m number of genes
name_gene_2 <- c(gene_1, gene_2, ..., gene_m)
samples <- c(sample1, ..., sampleN)
# expression matrix
Exp <- data.frame(gene1 = name_gene1, ..., geneM = name_geneM)
```

Then, we plot these two genes (see Figure 5a) using the command:

```
plot(x, y)
```

Trends are already apparent because data is simple but this is not usually the case. We then perform an statistical analysis using Principal Component Analysis. Before starting with PCA, it is best to first have centered the data with mean zero. This is, calculate the mean of each of the two variables and substracted to obtain centered data (shown in Figure 5b).

Now, let us calculate the covariance matrix. Covariance matrix for two variables:

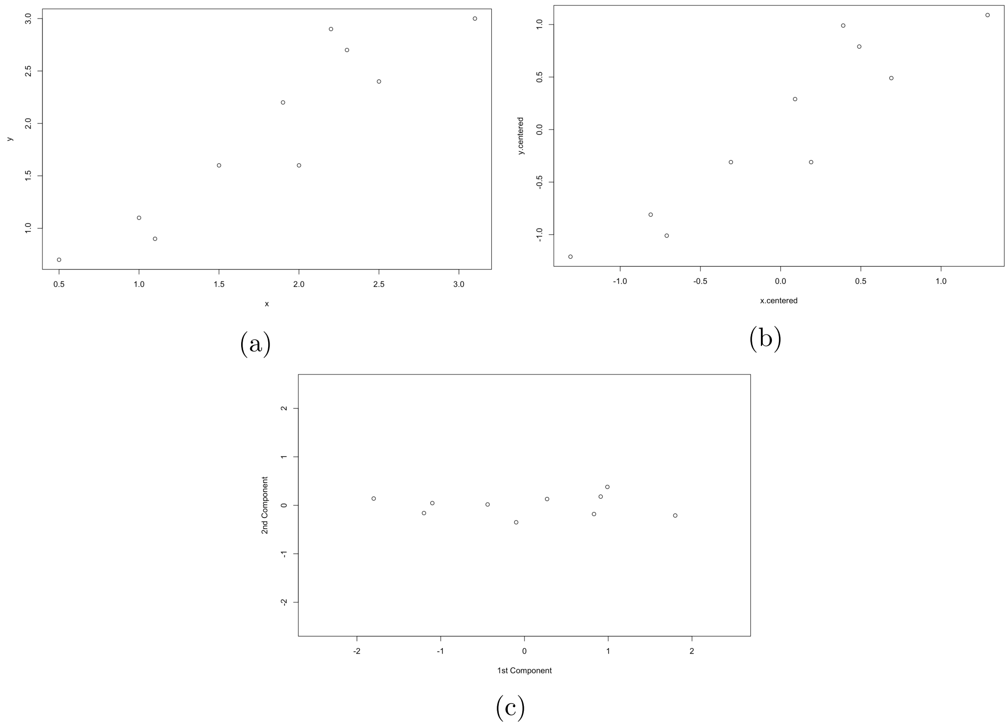
$$\begin{bmatrix} \mathrm{Cov(x,x)} & \mathrm{Cov(x,y)} \\ \mathrm{Cov(y,x)} & \mathrm{Cov(y,y)} \end{bmatrix}$$

and Covariance matrix for our data:

$$\begin{bmatrix} 0.016 & 0.615 \\ 0.615 & 0.716 \end{bmatrix}$$

The Eigenvalues of this matrix are: 1.284 and 0.0490. Eigenvalues gives the relative variance of the data in the direction defined by the Eigenvectors. From the values we can inferred that most variation is in one direction. To calculate the Eigenvalues in R use type:

```
eigen(covariance_matrix)
```



(a)

(b)

(c)

Figure 5: Plot: in (a) shows the expression of two genes, (b) the expression of the same two genes after centering the data (expression has zero mean), (c) gene expression is plot in the new coordiantes (PCA)

The corresponding eigenvector are then placed in a matrix in descending order of eigenvalue:

$$\begin{bmatrix} 0.6778734 & -0.7351787 \\ 0.7351787 & 0.6778734 \end{bmatrix}$$

The transpose of this Eigenvectos will perform the coordinate transformation:

$$W^T = \begin{bmatrix} 0.6778734 & 0.7351787 \\ -0.7351787 & 0.6778734 \end{bmatrix}$$

This is an orthogonal matrix which performs a rotated-axis coordinate transformation. We can transform our data matrix so that the data is represented in the new coordinates:

$$D_{PCA} = W^T D$$

which in our example is:

$$D_{PCA} = \begin{bmatrix} 0.6778734 & 0.7351787 \\ -0.7351787 & 0.6778734 \end{bmatrix} \begin{bmatrix} 0.69 & -1.31 & 0.39 & 0.09 & 1.29 & 0.49 & 0.19 & -0.81 & -0.31 & -0.71 \\ 0.49 & -1.21 & 0.99 & 0.29 & 1.09 & 0.79 & -0.31 & -0.81 & -0.31 & -1.01 \end{bmatrix}$$

$$= \begin{bmatrix} 0.83 & -1.8 & 0.99 & 0.27 & 1.8 & 0.91 & -0.099 & -1.1 & -0.44 & -1.2 \\ -0.18 & 0.14 & 0.38 & 0.13 & -0.21 & 0.18 & -0.35 & 0.046 & 0.018 & -0.16 \end{bmatrix}$$

The we can plot our data in the new coordinates, as shown in Figure 6, where each coordinate is called principal component. The first coordinate aligns with the direction in the expression space where has the most variation. Subsequent coordinates would align with directions with descending degrees of variation. This is why we are careful to order according to the size of the eigenvalues. Thus, PCA is capturing as much variation in the first component as possible, then the same for the second coordinate, and so on. In the case of our data, all the meaningful variation sees to have been captured with the first coordinate, or the first principal component. Specially compared to the second component which would seem to be random scatter. So we have reduced the dimensionality of our data from two to one. In cases when dealing with thousands of genes, PCA might be able to capture most of the variation of the data in with only two or three principal components. Thus making it easier to visualise it, which is one of the main motivations of performing PCA.

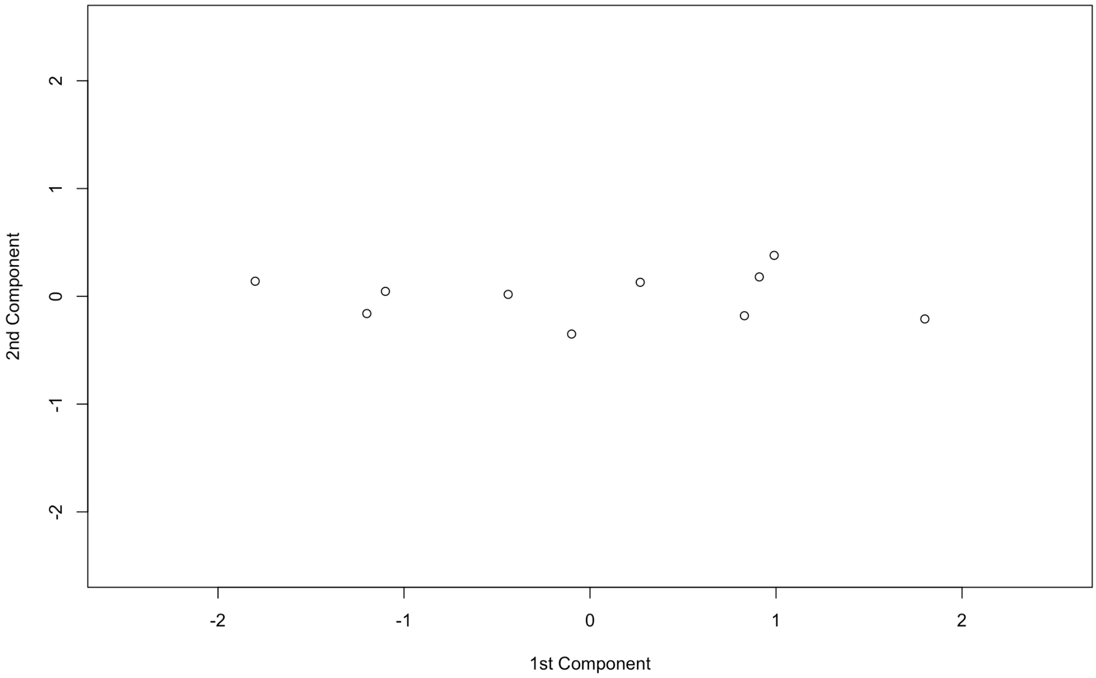


Figure 6: Plot of two genes in the new coordinates

2.2 Exercise: PCA using example data

In this example we will use a publicly available dataset to explore expression data. We will use the stats package in R for computing PCA and will show how to visualise it using the function ggbiplot, which was implemented by Vince Q Vu [6].

Gene expression data is usually stored in a tab delimited text file. The extension of such files could be .csv, .soft, .xls(x), etc. Use Excel, R or MATLAB to open and preview the file. It is important to mention that gene expression values must be normalised before PCA plotting.

The dataset used in this example is from a study on non-alcoholic fatty liver disease, published in 2015 by Wruck et al. [7]. The transcriptomics data was extracted from nine from patients that were recruited in the Multidisciplinary Obesity Research project at the Medical University of Graz, Austria, or at the Interdisciplinary Adipositas Center at the Kantonsspital St Gallen, Switzerland. Each sample is described in table 2) in terms of gender, age, BMI, percent of steatosis and steatosis grouping.

Table 2: Details of the samples for microarray data for a study in fatty liver [7]

ID	gender	age	BMI	% steatosis	steatosis grouping
H0004	f	54	47	10	obese, low steatosis
H0007	f	33	51	40	obese, high steatosis
H0008	m	61	46	40	obese, high steatosis
H0009	f	48	49	5 - 10	obese, low steatosis
H0011	f	58	45	70	obese, high steatosis
H0012	f	50	35	0	obese, low steatosis
H0018	f	35	41	30 - 40	obese, high steatosis
H0021	m	49	41	0	no steatosis
H0022	m	45	49	40	obese, high steatosis

The gene expression matrix, gene annotation and sample annotation can be found in the following files:

- **gene-expression-table.txt**: gene expression table. This table is available as reference but for the simplicity we will use the following files which contain the expression data separately from the gene annotation, the names of samples, and the steatosis groups per sample.

- **gene-expression.txt**: numerical matrix for the gene expression values, where the columns represent genes and the rows represent the samples. This is the transpose of the expression matrix because the function requires the rows of the input matrix to be observations and the columns features, which means rows to be the gene expression profiles (samples) and columns to be the genes.

- **gene-annotation.txt**: string vector containing the gene names for the expression data

- **samples.txt**: name of each sample

- **groups.txt**: name of the steatosis group for each sample

Now, we can load the data into R to begin the analysis. We can either load the gene expression table by typing:

```
#Expression data is saved in a tabular txt file
#The data is in a numerical matrix with no headers
ExpData <- read.delim(FileName, header = FALSE, sep = '\t',
stringsAsFactors = FALSE)
Annotation <- read.delim(FileName, header = TRUE, sep = '\t',
stringsAsFactors = FALSE)
samples <- read.delim(FileName, header = TRUE, sep = '\t',
stringsAsFactors = FALSE)
genes.class <- read.delim(FileName, header = TRUE, sep = '\t',
stringsAsFactors = FALSE)
```

Next, calculate the principal components using *prcomp* and plot the first two components. Then plot the first two components using *ggbiplot* [6]:

```
## Computing the principal components using prcomp
        genes.pca <- prcomp(gene_expression_data)

# Plot the components using ggbiplot
        library(ggbiplot)

        g <- ggbiplot(genes.pca, obs.scale = 1, var.scale = 1,
                      groups = genes.class$groups, ellipse = TRUE,
                      circle = FALSE, var.axes = FALSE) +
          scale_color_discrete(name = '') +
          theme(legend.direction = 'horizontal', legend.position = 'top')

        print(g)
```

ggbiplot produces a plot which is shown in Figure 7. Each dot is a gene expression from a sample in each category (group) from a patient, and is coloured by its type. The two axis are the first two principal components and the numbers represent the percentage of variance that is captured by each component. Typically, the first three component captures the most variance, whereas following components capture only a small percentage of variance. Dots of the same type tend to cluster together which means that samples of the same type have similar expression profiles. The distance on the dots on each axis should not be treated equally (as each component captures a different percentage of variance). Thus, the difference on the first component should be taken into more consideration. Furthermore, the ellipse in the figure represents the normal data ellipse for each group for the details of 68%.

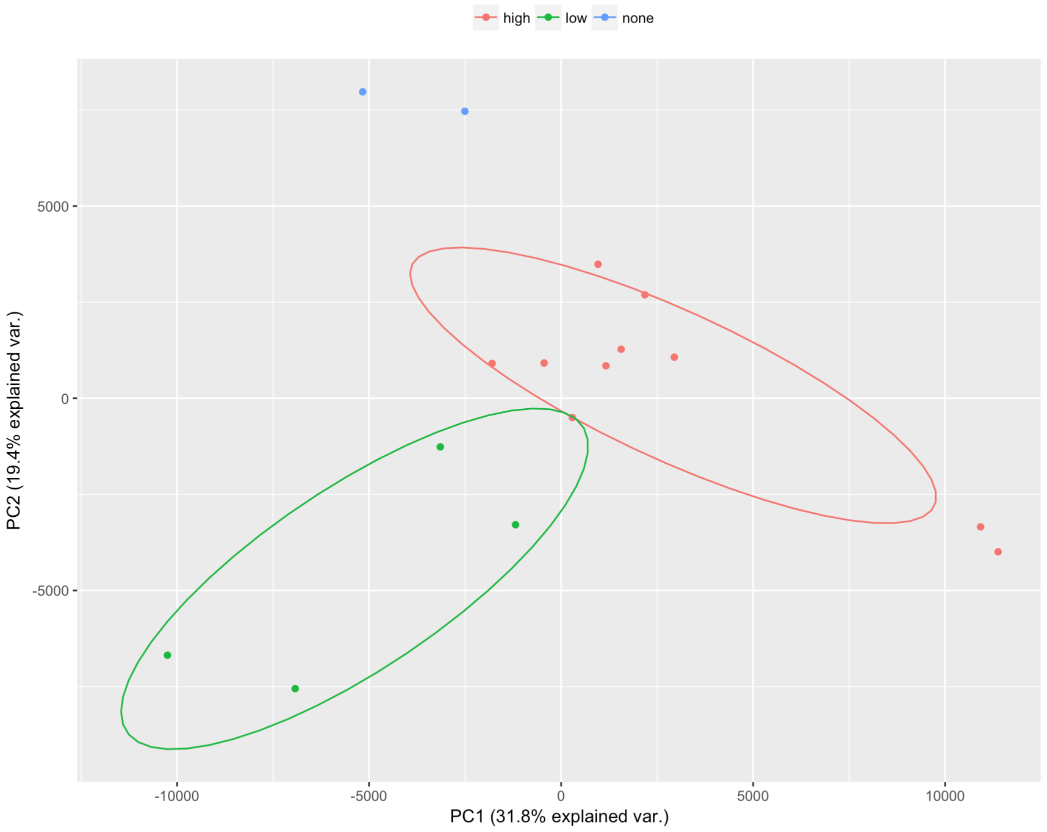


Figure 7: The first two principal components for gene expression data in a study on Fatty Liver. The plot was generated using the R package ggbiplot [6]

Extra exercise: plotting the three principal components

Plot the three principal components for the expression data. Figure 8 shows an example of such plot.

Summary

In summary, PCA is a method of revealing underling trends in large amounts of data. PCA reduces high dimensional data to just a few principal components which hopefully capture most of the variation of the data and allows inferring meaningful structure.

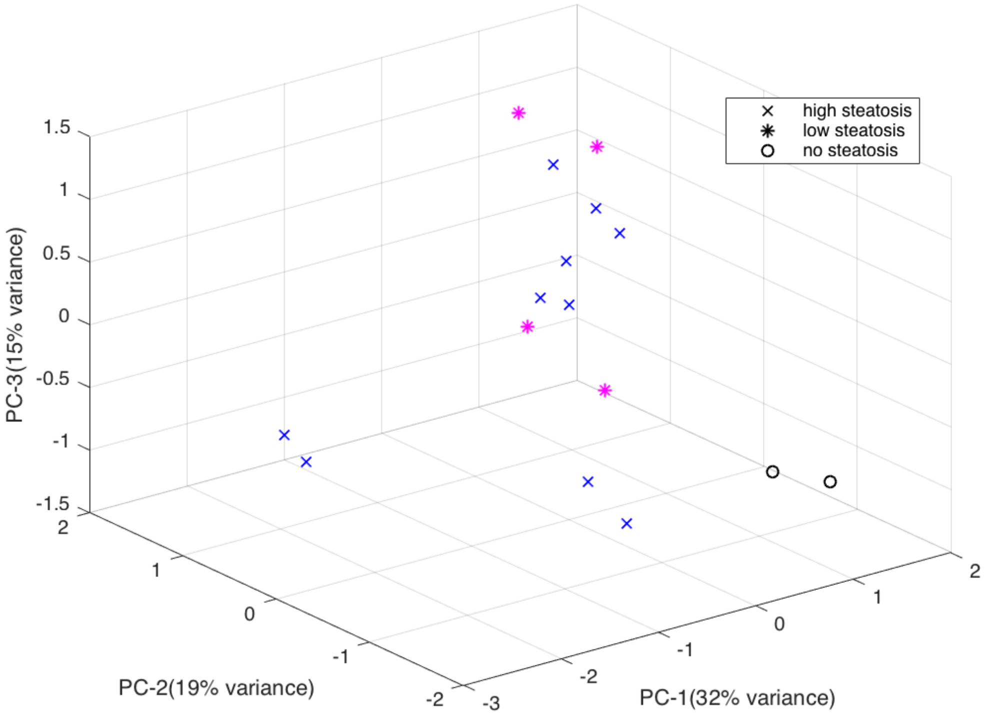


Figure 8: The first three principal components for gene expression data in a study on Fatty Liver [7]. Note: the plot was generated using MATLAB, can you plot the three components in R

A new coordinate system is constructed by rotating the axes (each representing a gene). The first new coordinate, or first principal component, is the direction in which the data varies most, then the second component, and so on. PCA allows to select a few new variables which contain most of the variation of the data which can also be visualised.

Some of the benefits of using PCA benefits are:

- A powerful tool to visualise high dimensional data
- Shows quantified difference among observations
- Used to assess data quality and discover relationships between data points
- Some software to compute PCA is available in MATLAB and R (using package stats)

3 Hierarchical Clustering

Hierarchical Clustering is another way to visualise high dimensional data. It clusters observations by distance and builds a hierarchical structure. It gives more detailed information of the differences among clusters, for example, what genes contributes the most to the differences between two clusters.

Hierarchical clustering uses a distance metric (typically Euclidean but could be correlation, Hamming distance, etc.) between each pair of genes to create a hierarchical tree-like structure of the data. Then it uses a linkage function to calculate the distance between clusters. For more details please see [1].

Figure 9 shows an example of clustergram from gene expression data. The clustergram is made of a heat map in the middle and dendograms in the left and top, with row and column labels on the right and bottom (depending on the number of genes and samples) and a scale bar. Each column is a sample expression profile, and each row represents a gene. The colours suggests relative expression values, where red indicates high expression values and blue indicates low expression values. Ideally, samples of the same type will cluster together, e.g., all control samples will cluster together and all cases as well.

3.1 Exercise: Hierarchical clustering for expression data

Compute hierarchical clustering on the Fatty Liver [7]. Since MATLAB is a commercial software, we will use a free to use wrapper for MATLAB's clustergram function. This wrapper is available from MultiPEN (`https://github.com/TGAC/MultiPEN/`, [4]). The wrapper provided in MultiPEN reads the expression data provided as a tabular file and plots the hierarchical clustering image on screen, which is also saved as a png image. To use the wrapper just use MultiPEN in a terminal using the following syntax:

Syntax

```
MultiPEN HierarchicalClustering Output Expression Threshold Title
```

Description

MultiPEN: This is the path to the binary executable of MultiPEN

Output: Specify directory to save the output image, default is: *output/MultiPEN/stats/*

Expression: The expression data is in tabular format where the rows represent the features (e.g. genes) and the colums are the samples.

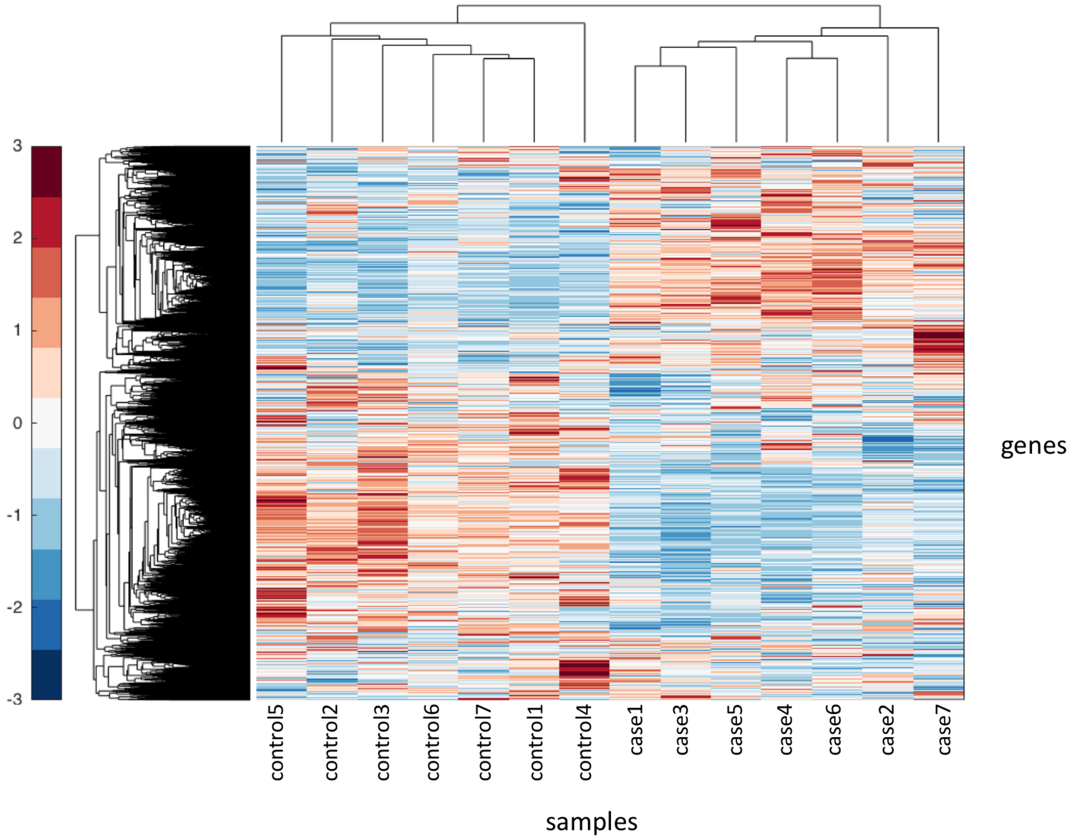


Figure 9: Example of hierarchical clustering using the MATLAB function clustergram

Threshold: To filter expression values. For example, for gene expression, it is common practice to discard genes with counts smaller than 100. This is an optional input argument.

Title: Specify the title to be displayed in the plot. This is an optional input argument.

Exercise

Run the the wrapper in MultiPEN to perform hierarchical clustering in the expression data for the Fatty Liver data [7] used in previous exercise. For more information on running MultiPEN visit `https://github.com/TGAC/MultiPEN/` [4].

4 Interaction Network

4.1 Exercise: Building a network using PSICQUIC

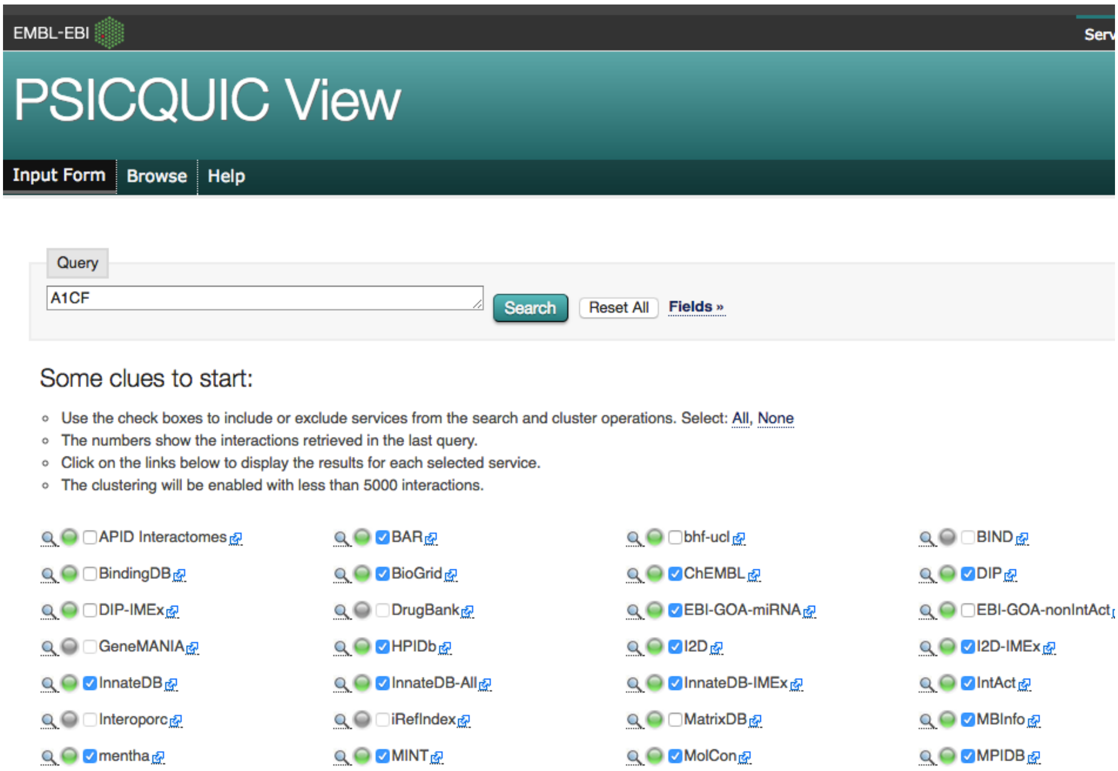
Build Protein-Protein Interaction network with PSICQUIC, `http://www.ebi.ac.uk/Tools/webservices/psicquic/view/home.xhtml`

PSICQUIC is a tool developed by Pablo Porras at EBI and it is in principle a meta-server that runs queries of PPI from multiple primary resources. It is available as:

1. Web-based tool http://www.ebi.ac.uk/Tools/webservices/psicquic/view/home.xhtml
2. R/Bioconductor http://bioconductor.org/packages/release/bioc/html/PSICQUIC.html
3. Perl or Python client script to run web queries through API

Using PSICQUIC web interface

1. Paste list of genes into the search query and select source db (see Figure 10).



A1CF

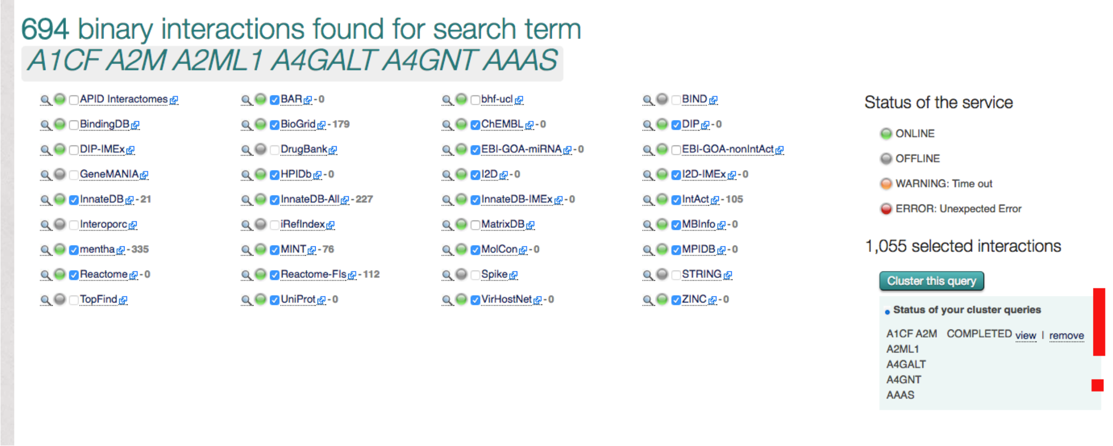
Some clues to start:

- Use the check boxes to include or exclude services from the search and cluster operations. Select: All, None
- The numbers show the interactions retrieved in the last query.
- Click on the links below to display the results for each selected service.
- The clustering will be enabled with less than 5000 interactions.

□ APID Interactomes	☑ BAR	□ bhf-ucl	□ BIND
□ BindingDB	☑ BioGrid	☑ ChEMBL	☑ DIP
□ DIP-IMEx	□ DrugBank	☑ EBI-GOA-miRNA	□ EBI-GOA-nonIntAct
□ GeneMANIA	☑ HPIDb	☑ I2D	☑ I2D-IMEx
☑ InnateDB	☑ InnateDB-All	☑ InnateDB-IMEx	☑ IntAct
□ Interoporc	□ iRefIndex	□ MatrixDB	☑ MBInfo
☑ mentha	☑ MINT	☑ MolCon	☑ MPIDB

Figure 10: Query of gene using PSICQUIC web interface

2. After initial search the service informs about number of interactions detected in given resource (see Figure 11).



694 binary interactions found for search term

A1CF A2M A2ML1 A4GALT A4GNT AAAS

□ APID Interactomes	☑ BAR - 0	□ bhf-ucl	□ BIND
□ BindingDB	☑ BioGrid - 179	☑ ChEMBL - 0	☑ DIP - 0
□ DIP-IMEx	□ DrugBank	☑ EBI-GOA-miRNA - 0	□ EBI-GOA-nonIntAct
□ GeneMANIA	☑ HPIDb - 0	☑ I2D - 0	☑ I2D-IMEx - 0
☑ InnateDB - 21	☑ InnateDB-All - 227	☑ InnateDB-IMEx - 0	☑ IntAct - 105
□ Interoporc	□ iRefIndex	□ MatrixDB	☑ MBInfo - 0
☑ mentha - 335	☑ MINT - 76	☑ MolCon - 0	☑ MPIDB - 0
☑ Reactome - 0	☑ Reactome-FIs - 112	□ Spike	□ STRING
□ TopFind	☑ UniProt - 0	☑ VirHostNet - 0	☑ ZINC - 0

Status of the service

● ONLINE
● OFFLINE
● WARNING: Time out
● ERROR: Unexpected Error

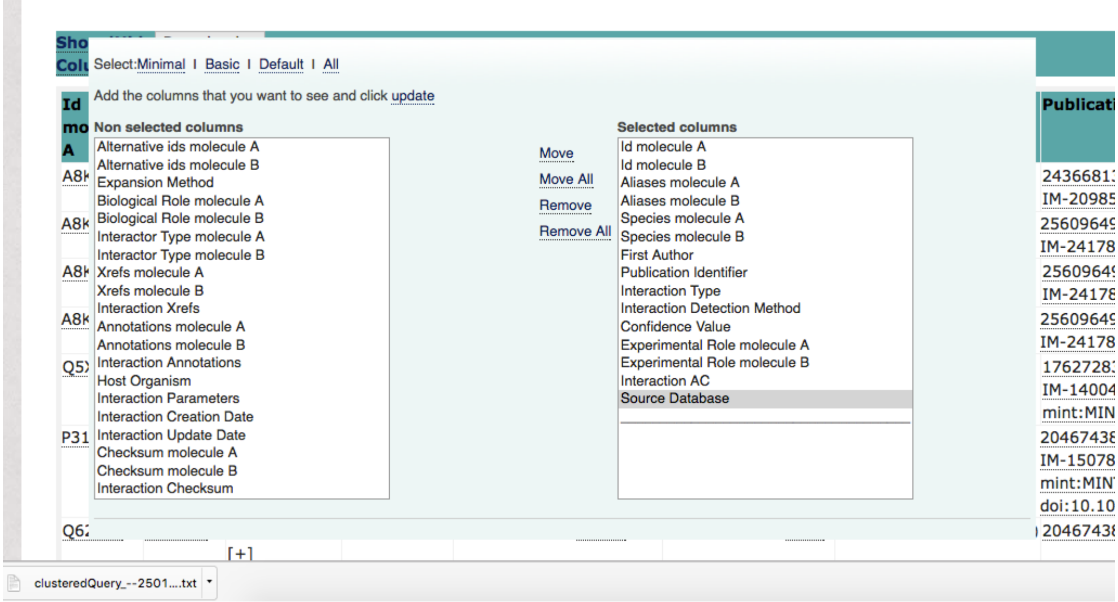
1,055 selected interactions

Status of your cluster queries

A1CF A2M A2ML1 A4GALT A4GNT AAAS COMPLETED view | remove

Figure 11: Result of the interactions found for search term in PSICQUIC

3. Now we can view results, customise list of displayed columns and download results (see Figure 12).



Select:Minimal | Basic | Default | All

Add the columns that you want to see and click update

Non selected columns		Selected columns
Alternative ids molecule A	Move	Id molecule A
Alternative ids molecule B	Move All	Id molecule B
Expansion Method	Remove	Aliases molecule A
Biological Role molecule A	Remove All	Aliases molecule B
Biological Role molecule B		Species molecule A
Interactor Type molecule A		Species molecule B
Interactor Type molecule B		First Author
Xrefs molecule A		Publication Identifier
Xrefs molecule B		Interaction Type
Interaction Xrefs		Interaction Detection Method
Annotations molecule A		Confidence Value
Annotations molecule B		Experimental Role molecule A
Interaction Annotations		Experimental Role molecule B
Host Organism		Interaction AC
Interaction Parameters		Source Database
Interaction Creation Date		
Interaction Update Date		
Checksum molecule A		
Checksum molecule B		
Interaction Checksum		

Figure 12: Customising view of results provided by PSICQUIC

4. Some of the resources e.g., Intact, Mentha, Reactome provide evidence scores that could be used for quality control purpose. Similarly as with StringDB is it advisable to use interactions with weight > 0.8 in order to decrease number of spurious

interactions. This cut-off is arbitrary and should be adjusted for specific application.

4.2 Compile the network with STRINGdb

Type the following code in an R script:

```
# Get StringInteractome Network

#INPUT
# fileName - table with column ’name’
# Uncomment next two lines and add values accordingly
# speciesName = ’human’
# speciesCode = 9606  #homo sapiens

# networkFileName = "network.csv"   # output file name

# Load file with list of genes
# Read data table
# with at least a column "name" for the list of genes
inputData <- read.delim( fileName, header = TRUE,
     sep = ’\t’, stringsAsFactors = FALSE)
geneList <- inputData$name

# begin compiling network
library(STRINGdb)
string_db <- STRINGdb$new( version="10", species = 9606,
 score_threshold=0, input_directory="" )
mapped <- string_db$map( inputData,  "name", removeUnmappedRows = TRUE )

#get interactions
inter<-string_db$get_interactions(mapped$STRING_id)

#annotate source and target nodes
from <- gsub("9606.","",inter$from)
to <- gsub("9606.","",inter$to)
#divide combined_score values by 1000 to have
#scores in the range [0,1]
network <- cbind(from,to,inter[16]/1000)
threshold <- 0.6   # select some relevant threshold
subNetwork <- network[network$combined_score > threshold,]
```

```
#edit STRING_id (speciesCode.ENSPxxxxx) to remove speciesCode.
stringID <- gsub(paste(speciesCode, ".", sep = ""), "", mapped$STRING_id)
drops <- "STRING_id"
mapped$STRINGID <- mapped$STRING_id
mapped <- mapped[!(names(mapped) %in% drops)]

# end compiling network

#write two files:
#1) all network edges and
#2) edges above specified threshold
cat(sprintf(’\nSaving network (edges) to file: %s’, fileName))
cat(’. . .’)
fileName <- paste(networkFileName, ’.txt’, sep = "")
write.table(mapped, fileName, sep = ’\t’, col.names = TRUE,
 row.names = FALSE, quote = FALSE)
cat(sprintf(’Done!’))

cat(sprintf(’\nSaving network for threshold: 0.60 in file: %s’, fileName))
cat(’. . .’)
fileName <- paste(networkFileName, ’score_0.60.txt’, sep = "")
write.table(subNetwork, fileName, sep = ’\t’, col.names = TRUE,
 row.names = FALSE, quote = FALSE)
cat(sprintf(’Done!’))
```

Part II
Statistical Approaches with Sparsity

5 Feature Selection using logistic regression

MultiPEN provides a framework for a combined analysis when different type of omics data available for a given study, specifically when data from RNA sequencing or microarray and metabolomics is available. This combined analysis aims to identify two aspects: first, the genes and metabolites that are key for the differences between two conditions, e.g., healthy vs disease, or patients under different treatments; and second, the pathway and biological processes involved in each condition.

In this work, we set the problem of finding the genes and metabolites that are key to discriminate between conditions as a problem of machine learning, where two conditions, i.e., control and cases, are described by a set of features (i.e., genes and metabolites) and their values (i.e., expression) for the conditions. We apply a penalised logistic regression approach to perform feature selection.

5.1 Exercise: Features selection from expression data using MultiPEN

We will use logistic regression for feature selection. Based on a penalised regularised logistic approach [5].

Perform feature selection for the gene expression data on the Fatty Liver study [7]. Using the feature selection function provided by MultiPEN (`https://github.com/TGAC/MultiPEN/`, [4]) which uses a penalised logistic regression approach.

Syntax

```
MultiPEN FeatureSelection Output Expression Interactions Class lambda
DecisionThreshold NumIterations
```

Description

MultiPEN: This is the path to the binary executable of MultiPEN

Output: Specify directory to save the output image, default is: *output/MultiPEN/stats/*

Expression: The expression data is in tabular format where the rows represent the features (e.g. genes) and the colums are the samples.

Interactions: The interaction matrix where the ith interaction (row) is represented as: [source target score] where *source* and *target* are names (symbolID for genes and CHEBI IDs for metabolites) of the connected nodes and *score* is a number in the range [0,1] representing the interaction confidence (where 1 corresponds to the maximum level of confidence).

Class: For each sample (one sample per row) specify if control (0) or case (1).

lambda: This is the lambda parameter that optimises the logistic regression problem for your specific data. Different lambdas can be tested using cross validation, then selecting the value that provides better results (in terms of the size of the largest connected component, accuracy or area under the curve).

DecisionThreshold: The decision threshold is set to 0.5 by default. However, if want to test another value specify it here.

NumIterations: Maximum number of iterations for the optimisation solver. Default value is 100.

Feature Selection Output Files

Feature selection produces seven output files:

MultiPEN-Rankings_lambdaX.txt: Ranking of features for the corresponding lambda X. This file contains the following columns and (n+5) rows (where n is the number of samples):

Column	Column Name	Description	Example (row 4 in Figure 13)
1	name	Feature name	PPOX
2	weight	Weight (in the range [-1,1])	0.00290391
3	ranking	Ranking according to the absolute weight, where ranking 1 corresponds to the most significant feature for the model	3
4	foldChange	Fold change to determine the expression change from control to cases	1.1735
5	higherIn	The average expression is higher in case or control	case
6	sample_1	First sample	case1
. . .	. . .	. . .	. . .
n + 5	sample_n	Last sample	control7

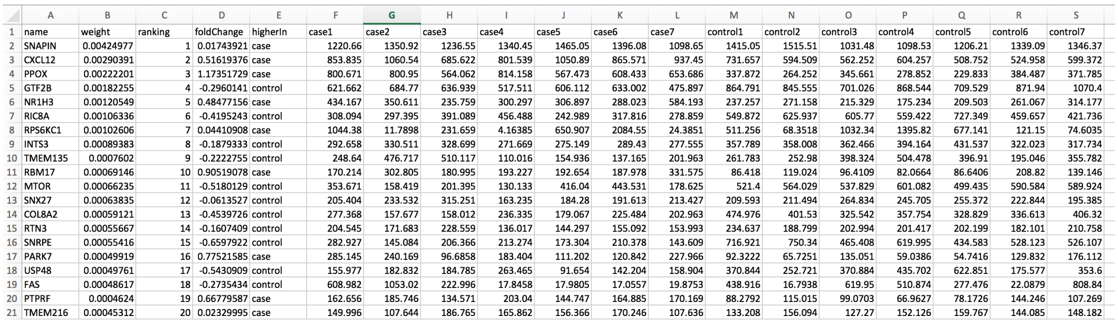
MultiPEN-Rankings_lambdaX_genes-higher-in-cases.txt: Ranking of features which includes only features with higher expression in cases samples.

MultiPEN-Rankings_lambdaX_genes-higher-in-control.txt: Ranking of features which includes only features with higher expression in control samples.

MultiPEN-vts_lambdaX.txt: Intercept term (logistic regression model).

MultiPEN-performance_feature-selection_lambdaX.txt: file that contains statistics on the performance of feature selection: largest connected component (LCC), area under the curve (auc), accuracy, true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

MultiPEN-feature-selection_config.txt: Contains the information of the parameters used: lambda, number of iterations and decision threshold.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	name	weight	ranking	foldChange	higherIn	case1	case2	case3	case4	case5	case6	case7	control1	control2	control3	control4	control5	control6	control7
2	SNAPIN	0.00424977	1	0.01743921	case	1220.66	1350.92	1236.55	1340.45	1465.05	1396.08	1098.65	1415.05	1515.51	1031.48	1098.53	1206.21	1339.09	1346.37
3	CXCL12	0.00290391	2	0.51619376	case	853.835	1060.54	685.622	801.539	1050.89	865.571	937.45	731.657	594.509	562.252	604.257	508.752	524.958	599.372
4	PPOX	0.00222201	3	1.17351729	case	800.671	800.95	564.062	814.158	567.473	608.433	653.686	337.872	264.252	345.661	278.852	229.833	384.487	371.785
5	GTF2B	0.00182255	4	-0.2960141	control	621.662	684.77	636.939	517.511	606.112	633.002	475.897	864.791	845.555	701.026	868.544	709.529	871.94	1070.4
6	NR1H3	0.00120549	5	0.48477156	case	434.167	350.611	235.759	300.297	306.897	288.023	584.193	237.257	271.158	215.329	175.234	209.503	261.067	314.177
7	RIC8A	0.00106336	6	-0.4195243	control	308.094	297.395	391.089	456.488	242.989	317.816	278.859	549.872	625.937	605.77	559.422	727.349	459.657	421.736
8	RPS6KC1	0.00102606	7	0.04410908	case	1044.38	11.7898	231.659	4.16385	650.907	2084.55	24.3851	511.256	68.3518	1032.34	1395.82	677.141	121.15	74.6035
9	INTS3	0.00089383	8	-0.1879333	control	292.658	330.511	328.699	271.669	275.149	289.43	277.555	357.789	358.008	362.466	394.164	431.537	322.023	317.734
10	TMEM135	0.0007602	9	-0.2222755	control	248.64	476.717	510.117	110.016	154.936	137.165	201.963	261.783	252.98	398.324	504.478	396.91	195.046	355.782
11	RBM17	0.00069146	10	0.90519078	case	170.214	302.805	180.995	193.227	192.654	187.978	331.575	86.418	119.024	96.4109	82.0664	86.6406	208.82	139.146
12	MTOR	0.00066235	11	-0.5180129	control	353.671	158.419	201.395	130.133	416.04	443.531	178.625	521.4	564.029	537.829	601.082	499.435	590.584	589.924
13	SNX27	0.00063835	12	-0.0613527	control	205.404	233.532	315.251	163.235	184.28	191.613	213.427	209.593	211.494	264.834	245.705	255.372	222.844	195.385
14	COL8A2	0.00059121	13	-0.4539726	control	277.368	157.677	158.012	236.335	179.067	225.484	202.963	474.976	401.53	325.542	357.754	328.829	336.613	406.32
15	RTN3	0.00055667	14	-0.1607409	control	204.545	171.683	228.559	136.017	144.297	155.092	153.993	234.637	188.799	202.994	201.417	202.199	182.101	210.758
16	SNRPE	0.00055416	15	-0.6597922	control	282.927	145.084	206.366	213.274	173.304	210.378	143.609	716.921	750.34	465.408	619.995	434.583	528.123	526.107
17	PARK7	0.00049919	16	0.77521585	case	285.145	240.169	96.6858	183.404	111.202	120.842	227.966	92.3222	65.7251	135.051	59.0386	54.7416	129.832	176.112
18	USP48	0.00049761	17	-0.5430909	control	155.977	182.832	184.785	263.465	91.654	142.204	158.904	370.844	252.721	370.884	435.702	622.851	175.577	353.6
19	FAS	0.00048617	18	-0.2735434	control	608.982	1053.02	222.996	17.8458	17.9805	17.0557	19.8753	438.916	16.7938	619.95	510.874	277.476	22.0879	808.84
20	PTPRF	0.0004624	19	0.66779587	case	162.656	185.746	134.571	203.04	144.747	164.885	170.169	88.2792	115.015	99.0703	66.9627	78.1726	144.246	107.269
21	TMEM216	0.00045312	20	0.02329995	case	149.996	107.644	186.765	165.862	156.366	170.246	107.636	133.208	156.094	127.27	152.126	159.767	144.085	148.182

Figure 13: Example of output file for feature selection

Using MultiPEN for feature selection

In the command line, navigate to the folder where the binary for MultiPEN is located, i.e., binary-Linux/MultiPEN_v001_Linux/. Then create variables for the paths to standalone application, output directory and input files by typing:

```
Output="ExampleOutputs/"
Expression="ExampleInputs/X.txt"
Interactions="ExampleInputs/E.txt"
Class="ExampleInputs/Y.txt"
lambda=0.001
```

Next, run feature selection with the following command:

```
MultiPEN FeatureSelection $Output $Expression $Interactions $Class $lambda
```

5.2 Cross Validation

It is a technique to determine whether a model can be generalised to other similar databases (it measure the accuracy of a model).

Approach:

- Divide the dataset into training and test datasets
- Fit the model to the training set

- Use test set to evaluate goodness of fit

Theory

- Signal is correlated across tests and training sets
- Noise is uncorrelated across tests and training sets

We will use the wrapper provided in MultiPEN to perform cross validation.

Syntax

```
MultiPEN CrossValidation Output Expression Interactions Class
lambdas Folds NumIterations
```

Description

MultiPEN: This is the path to the binary executable of MultiPEN, i.e., binary-Linux/MultiPEN_v001_OS/.

OutputDirectory: Specify directory for output files.

ExpressionData: The expression data is in tabular format where the rows are the features (genes and/or metabolites) and the columns are the samples.

Interactions: The interaction matrix where the ith interaction (row) is represented as: [source target score] where *source* and *target* are names (symbolID for genes and CHEBI IDs for metabolites) of the connected nodes and *score* is a number in the range [0,1] representing the interaction confidence (where 1 corresponds to the maximum level of confidence).

SampleClass: For each sample specify if control (0) or case (1).

lambdas: Set of lambdas to test for cross validation. If you are wanting to test more than one lambda, specify the lambdas by using the notation (include the quotation mark symbols): “[lambda1 lambda2 . . . lambdaN]”. For example, if we want to try two lambdas, namely 0.02 and 0.2, we would specify it with: “[0.02 0.2]”.

Folds: Specify the number of partitions for cross validation.

NumIterations: Maximum number of iterations for the optimisation solver. Default value is 100.

Cross Validation Output Files

Cross Validation produces one output file:

cross-validation_stats.txt: Statistics for tests which include, for each lambda, the size of the largest connected component (LCC), the standard deviation of the largest connected component (std_LCC), the number of selected features (selected, i.e., features which weights are different to zero), area under the curve (AUC), and the standard deviation of the area under the curve (std_AUC).

Example

In the command line, navigate to the folder where the binary for MultiPEN is located, i.e., MultiPEN_v001_Linux/. Then create variables for the paths to stand-alone application, output directory and input files by typing:

```
Output="ExampleOutputs/"
Expression="ExampleInputs/X.txt"
Interactions="ExampleInputs/E.txt"
Class="ExampleInputs/Y.txt"
Folds=3
NumIter=3000
```

Note that in this example we are using the example files provided with the application. All the files used for the example are located in the folder: ExampleInputs/.

Next, test Cross Validation for lambdas: ”[0.000001 0.00001 0.0001 0.001 0.01 0.1 1 10]”. Type the following command:

```
MultiPEN CrossValidation $Output $Expression $Interactions $Class
"[0.000001 0.00001 0.0001 0.001 0.01 0.1 1 10]" $Folds $NumIter
```

References

[1] MATLAB and MathWorks. Compute hierarchical clustering.

[2] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[3] Michael E Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, and Trey Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, Feb 2011.

[4] Perla Troncoso-Rey and Wiktor Jurkowski. Multipen, 2017.

[5] Nikos Vlassis and Enrico Glaab. Genepen: analysis of network activity alterations in complex diseases via the pairwise elastic net. *Stat Appl Genet Mol Biol*, 14(2):221–224, Apr 2015.

[6] Vinve Q. Vu. Ggbiplot.

[7] Wasco Wruck, Karl Kashofer, Samrina Rehman, Andriani Daskalaki, Daniela Berg, Ewa Gralka, Justyna Jozefczuk, Katharina Drews, Vikash Pandey, Christian Regenbrecht, Christoph Wierling, Paola Turano, Ulrike Korf, Kurt Zatloukal, Hans Lehrach, Hans V. Westerhoff, and James Adjaye. Multi-omic profiles of human nonalcoholic fatty liver disease tissue highlight heterogenic phenotypes. *Scientific Data*, 2:150068 EP –, 12 2015.