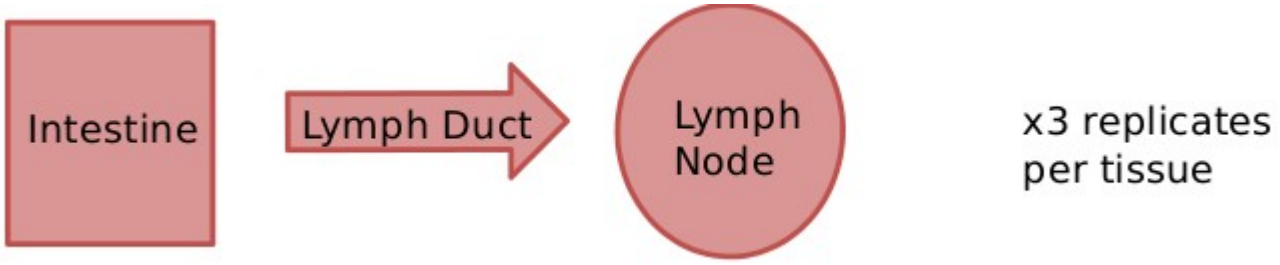
RNA-seq high level analysis, practical session 1

Introduction

This project centres on a novel and unpublished RNA-seq data-set. The data includes nine samples and three sample groups. The data is centred one the question: What happens to dendritic cells as they migrate from the small intestine through the lymph duct and into the lymph node? This migration is a natural immune response, occuring when dendritic cells in the gut “sense” a threat to the body (e.g. a bacterial infection). They then migrate through the lymph duct to the lymph node – where they stimulate an immune response. The data is for CD103 single positive dendritic cells isloated from mice. The data has already been aligned to the mm10 genome using tophat2, and read counts have been calculated using htseq count with and version 87 of the mouse transcriptome. FPKMs were generated using string-tie.

The experimental design is:



Our main research questions are:

1. Has the experiment worked?
2. Do the cells from each stage of the migration differ from each other transcriptionally?
3. What pathways and regulators might be involved in the migration and response?
4. Can we identify any genes specific to each stage of the migration?

The required files can be found here:

- counts_matrix.csv
- FPKM_matrix.csv

Task 1 – differential expression using DESeq2

Your first task is to determine the differential expressed genes between each tissue (i.e. intestine vs lymph duct and also lymph duct vs lymph node). In addition you will be required to annotate the output files with information from Ensembls biomart.

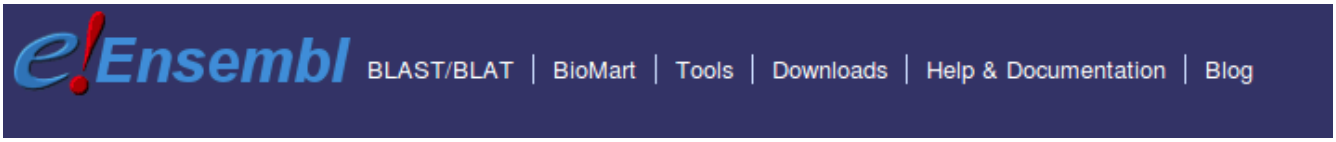
Use and modify as appropriate the following R code to obtain the differential expression values:

```
#Multi-core settings:
library("BiocParallel")
register(MulticoreParam(2))

# Import data
countdata <- read.table("PATH TO YOUR COUNTS MATRIX", header=TRUE, row.names=1)


# Convert to matrix
countdata <- as.matrix(countdata)
head(countdata)

# Filter for genes with a median >= 1 read
countdata <- subset(countdata,apply(countdata, 1, median) >= 1)

# Assign conditions – one per sample.  Note these must be in the same order as the columns of the counts matrix:
condition <- factor(c(rep("WHAT IS THE FIRST GROUP?", 3), rep("WHAT IS THE SECOND GROUP?", 3), rep("WHAT IS THE
THIRD GROUP?", 3)))

# Analysis with DESeq2

library(DESeq2)

# Create a coldata frame and instantiate the DESeqDataSet. See ?DESeqDataSetFromMatrix
coldata <- data.frame(row.names=colnames(countdata), condition)
dds <- DESeqDataSetFromMatrix(countData=countdata, colData=coldata, design=~condition)
dds

# Run the DESeq pipeline
dds <- DESeq(dds)


#Output the normalised read counts matrix:
resdata <- as.data.frame(counts(dds, normalized=TRUE))
#Convert to a data-frame, with a column at the start called "IDs" for the gene IDs
resdata <- data.frame(IDs = row.names(resdata), resdata)
#Write results
write.table(resdata, file="PATH TO OUTPUT YOUR NORALISED COUNT MATRIX",row.names=FALSE, sep="\t", quote = FALSE)


#Output the differential expression file for Intestine vs Lymph Duct:
res <- results(dds, c("condition","lymph_duct","intestine"))
# Order by adjusted p-value
res <- res[order(res$padj), ]
# Convert to a data-frame, with a column at the start called "IDs" for the gene IDs
resdata <- data.frame(IDs = row.names(as.data.frame(res)), as.data.frame(res))
# Write results
write.table(resdata, file="PATH TO OUTPUT YOUR FIRST DIFFERENTIAL EXPRESSION FILE TO",row.names=FALSE, sep="\t",
quote = FALSE)

#Output the differential expression file for Lymph Duct vs Lymph Node:

HOW DO YOU DO THIS? HINT  - ITS REALLY EASY.
```

Task 2 – downloading annotations from biomart

Next, you need to annotate your output files with some extra information for each gene. This is useful later on when you try and interpret your plots and data. Firstly, you will need to obtain the correct annotations from Ensembl. Remember, your genome version is mm10, and transcriptome version in 87.

Go-to the ensembl website and click on the biomart tab: https://www.ensembl.org/index.html



Biomart includes gene annotations for most genomes. Have a quick look to see the kinds of things that you can download.

However, note we have not used the latest version of the transcriptome to align and process our data. We have used version 87. To obtain annotations for version 87 you will need to access the ensembl archive site. Go back to the amin ensembl page. The archive link is hidden away in the bottom right.

Once you are in the correct archive site, go-to biomart and download the following annotations:

- Gene ID
- Associated Gene Name
- Chromosome/scaffold name
- Gene Start (bp)
- Gene End (bp)
- Strand
- Gene type

Click “results” then download as a TSV.

Task 3: Annotating and combining your files

Now that you have your annotations downloaded, you need to combine your various expression, differential expression and annotation files into one master file. This makes handling your data easier and more convenient at later stages. Because you will only need to consider one file for all plots and analysis.

R is perfect for combining many files together. As it allows you to join files based on a common ID column. We will use the merge() function. We wish to merge the following files, using the Ensembl Ids:

- The FPKM file provided
- Your annotations file
- Your two differential expression files generated from DESeq 2

Firstly we must open our files in R using the read.table() method. It is important to store each file under a different variable name. You may need the sep="\t" parameter to open the annotations file:

```
FPKMs <- read.table("PATH TO YOUR FPKM FILE", header=TRUE)
annotations <- read.table("PATH TO YOUR ANNOTATIONS FILE", header=TRUE)
DE_intestine_vs_lymph_duct <- read.table("PATH TO YOUR DESEQ2 FILE COMAPARING INTESTINE TO LYMPH DUCT", header=TRUE)
DE_lymph_duct_vs_lymph_node <- read.table("PATH TO YOUR DESEQ2 FILE COMAPARING LYMPH DUCT TO LYMPH NODE", header=TRUE)
```

Next we need to consider the header names for each table. There are two issues:

1. The merge() function requires the data.tables being merged to have the same column name for the ID column. In the two DESeq2 differential expression files this will be "IDs", however in the annotations file this will be "Gene.ID", and in the FPKMs it will be "ID".
2. The two DESeq2 differential expression files have identical column headers, though they are of different comparisons. When we merge the files we will not easily be able to tell which column belongs to which comparison.

To get around these issues, you will need to re-name several of the column headers in several of your files. To do so use the colnames() function. You can call your columns anything you like, but as a rule of thumb try to keep them concise, descriptive and free of spaces (use _). Here's an example:

```
colnames(annotations) <- c("IDs","symbol","chr","start","stop","strand","type")
```

For more information please see: https://howtoprogram.xyz/2016/10/02/r-rename-column-data-frame-change-column-names/

Now we are ready to merge the files. The merge() function merges files pairs of files, by a common ID column. For example:

```
masterFile <- merge(FPKMs, annotations, by="IDs")
```

You will need to make several consecutive merges, to make a single master file. Finally, you will need to save your master table to a file using:

```
write.table(masterFile, file="PATH YOU WANT TO SAVE YOUR MASTER FILE AS",row.names=FALSE, sep="\t", quote = FALSE)
```

For more information please see: https://www.statmethods.net/management/merging.html

Task 4: filtering your files to include only coding genes:

RNA-seq gives us expression information in a genome wide and largely unbiased way. This means it not only includes expression levels for coding genes, but also non-coding genes. For this study, in order to simplify the analysis, we only wish to investigate the effects of dendritic cell migration on coding genes.

We will have to remove the non-coding genes from the master file, using the subset() function. The subset function lets us choose rows of a table based on the value of a column (using a logical function). The annotations file includes a column for “gene type”, we wish to select only rows with the entry “protein_coding”. Use R to load the master file into a table, and the subset method:

```
masterFile <- read.table("PATH TO YOUR MASTER FILE", header=TRUE)
masterFileKnownCoding <- subset(masterFile, NAME OF YOUR GENT YPE COLUMN == “protein_coding”)
write.table(masterFileKnownCoding, file="PATH TO OUTPUT YOUR KNOWN CODING MASTER FILE TO",row.names=FALSE, sep="\t", quote = FALSE)
```

You can use the subset() method to make other useful files. Can you make and save the following files:

- Only the significant genes (p.adj < 0.05 and absolute log2fold >1)
- Only the significantly upregulated genes (i.e. p.adj < 0.05 and positive fold change >1)
- Only the significantly downregulated genes (i.e. p.adj < 0.05 and negative fold change < -1)
- Note, you will need to make one of each file type for each of your two comparisons.

Task 5: Selecting only certain columns from a table:

Often RNA-seq high level analysis software require an input of lists of gene Ids or names, without any additional information. This is true for many of the pathway analysis software we will be using towards the end of the week. These tools require lists of the significant gene Ids, or gene names. Your next task is to use R to make these files.

There are two useful methods to do this in R. Firstly simply select the column in the table by name, and save it to a new variable. For example:

```
masterFileKnownCodingSig <- read.table("PATH TO YOUR SIGNIFICANT GENES MASTER FILE", header=TRUE)
masterFileKnownCodingSigIDs <- masterFileKnownCoding$Ids
```

The second method is to directly select columns by column number. E.g. the first column is number 1. For example:

```
masterFileKnownCodingSig <- read.table("PATH TO YOUR SIGNIFICANT GENES MASTER FILE", header=TRUE)
columns_to_select <- c(COLUMN NUMBER)
masterFileKnownCodingSigIDs <- masterFileKnownCodingSig[,columns_to_select]
```

The advantage of the first method is that its less fiddly, and so harder to make mistakes with. The advantage of the second method is that you can select multiple columns, for example:

```
columns_to_select <- c(1,2,3,7,8,9)
columns_to_select <- c(1:9)
```

Use one of the above methods to make separate IDs only and separate gene name only files for the significant genes, significantly upregulated genes and significantly downregulated genes.

Task 6: Try to make some plots of your own, which describe that data in a biological context.

This task is a free-for-all, you can use any method you like (even microsoft excel), to try and make some plots of the data. Make anything you like, but the aim is to try to explore the dataset with the biological aim of the experiment in mind (see Introduction if you've forgotten this). We will discuss your plots at the next session, and then explore common methods for exploring RNA-seq data. Only perform this task if you feel that you have time. There is no pressure.

GENERAL WORD OF WARNING: ALWAYS CHECK YOUR OUTPUT FILES TO SEE IF THEY ARE WHAT YOU THINK THEY ARE.

SOME ADVICE: NAME YOUR FILES SENSIBLY, AND KEEP THEM NEATLY ORGANISED IN FOLDERS.

RNA-seq high level analysis, practical session 2

Introduction

Now that we have processed and annotated our data, we can start to mine it. Mining data is simply a way of revealing hidden patterns that are of interest to us. Typically this is through an iterative process of asking a question, parsing a file, performing a statistical analysis, and plotting a result. In this section we will ask several different questions of the data, each of which will involve a new plot type. These plots are highly typical of RNA-seq analysis. Though we have two different comparisons (intestine vs lymph duct, and lymph duct vs lymph node) we will for now consider these only separately.

During this session we will make the following plots:

- Expression distribution density plots
- PCA scatter plots
- MA plots
- Volcano plots

You will make several of these plots for both of the differential comparisons.

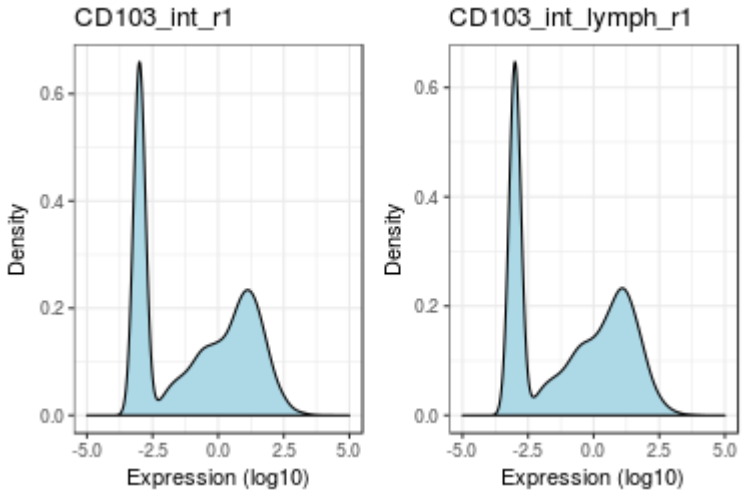
You will need the following R libraries:

```
require(ggplot2)
require(Rmisc)
require(ggfortify)
```

Task 1: An expression distribution plot for each sample

Question: globally how similar are our samples to each other? Has our experiment worked technically?

Here we aim to show for each sample the distribution of expression values. Generally these are expected to look highly similar across all samples and sample groups, and so are useful for assessing the quality of the experiment technically. Outlier samples with atypical distributions of expression values can be indicative of technical issues with that sample − such as insufficient quantities of RNA during library preparation. As you have 9 samples, you should produce 9 plots here. Here is an example of two.



Use and modify as appropriate the following R code to generate the plots:

```
#We read the expression matrix file:
x <- read.table("PATH TO YOUR MASTER FILE", header=TRUE)

#We generate the plot:
png("PATH TO SAVE YOUR PLOT TO.png", height=750, width=750, pointsize=10)
ggplot(x, aes(x=x$COLUMN NAME FOR THE SAMPLE YOU WISH TO PLOT))
 + geom_density(fill="lightblue")
dev.off()
```

Now you need to generate 8 more!

How do the plots look? You can improve them by logging the x-axis (to the base 10). This “squishes” the plot, making it easier to look at. However, as you cannot log a zero value - and many genes are unexpressed, we need to add an infinitesimal value to each genes expression value before we can log. Try swapping the following piece of code into your own code:

```
x=log10(x$COLUMN NAME FOR THE SAMPLE YOU WISH TO PLOT+0.001)
```

Next, we can add extra visualisation options to the ggplot. Here's a few examples for you to play with. Note these must be placed after the ggplot() method, but before dev.off():

```
 + xlim(-5, 5)
 + xlab("Expression (log10)")
 + ylab("Density")
```

```
 + ggtitle("A TITLE FOR YOUR PLOT, IDEALLY THE SAMPLE NAME")
 + theme_bw()
 + theme(legend.position="none")
```

You are doing great. Our final step is to use ggplots to generate the nine plots as a single png. For this we use the multiplot() method. Use the following code as a guide to understanding multi-plots. Then, generate a single multiplot png containing your nine plots:

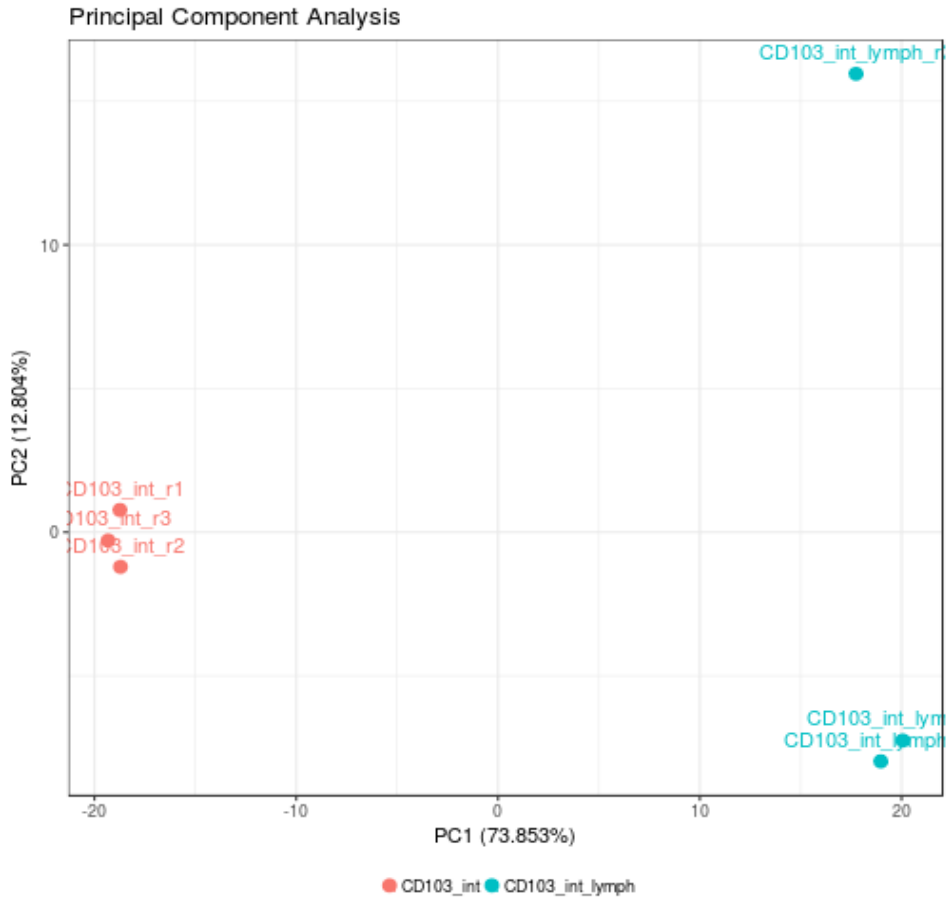
```
png("PATH TO SAVE YOUR MULTI-PLOT TO.png", height=750, width=750, pointsize=10)
plot1 <- ggplot(x, aes(x=x$COLUMN NAME FOR THE FIRST SAMPLE YOU WISH TO PLOT)) + geom_density(fill="lightblue")
plot2 <- ggplot(x, aes(x=x$COLUMN NAME FOR THE SECOND SAMPLE YOU WISH TO PLOT)) + geom_density(fill="lightblue")
plot3 <- ggplot(x, aes(x=x$COLUMN NAME FOR THE THIRD SAMPLE YOU WISH TO PLOT)) + geom_density(fill="lightblue")
p <- multiplot(plot1,plot2,plot3,cols= NUMBER OF COLUMNS THAT YOUR INDIVIDUAL PLOTS ARE DIVIDED INTO ON THE
print(p)
dev.off()
```

Well done. Now write a couple of sentences interpreting these plots. What do they show? Are there any outlier samples? Why are there several peaks on the plots? We can discuss this in more detail later.

Task 2: Principal Component Analysis (PCA) plots

Question: How do the samples vary from each other? Is this variation related to our sample groups or to individual outlier samples? Can we detect obvious differences between our samples?

It is useful in RNA-seq to get an idea of how different our samples are globally. For this we use Principal Component Analysis (PCA). In simple terms, PCA is a technique that uses all expression values to find underlying variables (known as principal components) that best differentiate your samples. Principal components are dimensions along which your data points are most spread out. Generally we expect sample groups to separate and replicates to cluster together. Groups not separating suggests little difference between groups, and replicates not clustering suggests sample heterogeneity. The principal component 1 is the dimension that best separates the samples. Here's an example of a PCA plot:



Your next task is to make a PCA plot for your dataset. PCA is a statistical method, and can be applied to a matrix of expression values. Once applied PCA returns a list of coordinates for each sample for each principal component. There will be as many components as there are samples, but the components do not necessarily relate to a particular sample. They are hypothetical dimensions that best separate your samples. The components are numbered in order of how well they separate the samples – based on the variation between them, with PC1 being the dimension that best separates your samples.

Luckily R has a pre-implemented PCA method called prcomp(). We will use this to plot a scatter plot of PC1 vs PC2. Modify the following code to generate the plot:

```
#We read the master file – note the use of row.names = 1:
x <- read.table("PATH TO YOUR MASTER FILE", header=TRUE, row.names = 1)

#We select only the columns with the expression values (as the PCA method only requires the expression values):
FPKM_columns <- c(COMMA SEPARATED LIST OF ALL THE FPKM COLUMNS)
x <- x[,FPKM_columns]

#We parse the expression matrix and perform PCA:
x <- as.matrix(sapply(x, as.numeric))
xx <- prcomp(t(x))

#We plot the PCA scatter plot of PC1 vs PC2:
png("PATH TO OUT{UT YOUR PCA SCATTER PLOT TO.png", width = 500, height = 500, pointsize=5)
ggplot(xx, aes(x=PC1,y=PC2)) + geom_point(size=3, shape=21)
dev.off()
```

As mentioned before there will be as many principal components as there are samples. Explore these briefly by making similar plots for other components (e.g. change x=PC1 to x=PC3, etc).

PCA considers genes with higher expression to be more important than genes with lower expression. This can mean that the analysis can be driven by a small number of very highly expressed genes, which is misleading. To work around this a PCA plot can usually be improved by logging the expression data. This “squishes” the expression values. Try replotting the PCA with this piece of code, just after you select the FPKM columns:

```
x <- log10(x+1)
```

Next, we want to colour the dots by the sample group (tissue). This can help us to interpret the plot. For this we make use of ggplots “fill” and “colour” parameters. If you supply a list of sample groups to these parameters ggplots will automatically colour the samples by sample group. Note: the list of sample groups MUST be the same length and in the same order as the FPKM columns. i.e. the list must have nine entries, for our data. It will then link your sample groups list back to your samples. This is the same idea as our conditions list in the DESeq2 step from last session. Modify the following code to colour the dotsin your PCA plot:

```
y <- c("CD103_int","CD103_int","CD103_int","CD103_lymph_duct","CD103_lymph_duct","CD103_lymph_dict")
ggplot(xx, aes(x=PC1,y=PC2, fill = y, colour = y)) + geom_point(size=3, shape=21)
```

Excellent. Next we want to add some extra visualisation options. Try some of the following:

```
+ geom_text(aes(label=colnames(x)),hjust= 0.5, vjust=-1)
+ ggtitle("Principal Component Analysis")
+ theme_bw()
+ theme(legend.position="bottom", legend.title = element_blank())
+ xlab(paste("PC1 (",summary(xx)$importance[2]*100,"%)",sep=""))
+ ylab(paste("PC2 (",summary(xx)$importance[5]*100,"%)",sep=""))
```

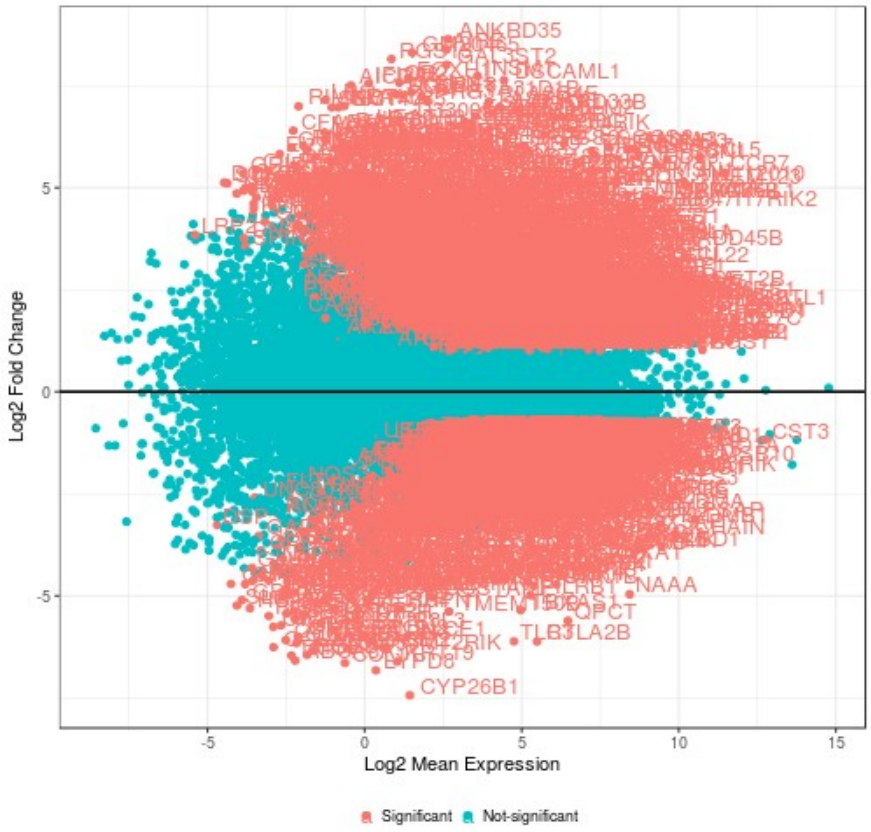
What do these extra options do? Make sure that you understand what the x and y labels mean, and how they are getting the information from the PCA data.

Finally, write a couple of sentences interpreting the plot. What does it show? Are there any outlier samples? What does the plot mean biologically? We can discuss this in more detail later.

Task 3: MA plots

Question: Is there a relationship between the expression level and fold change? Are more highly expressed genes more likely to be differentially expressed? Can we detect differential expression in lowly expressed genes?

It is useful in RNA-seq to get a feel for the relationship between expression level, significance and fold change. The three are strongly related. Lets think more about this. Each read is a biological observation, so more reads means more observations, and thus the more statistical power we have. Highly expressed genes will typically have more reads than lowly expressed genes. So, we can better detect changes in expression at highly expressed genes. To explore this relationship we use MA plots. These plot mean expression per gene, vs the fold change for that gene. We then colour non significant and significant genes differently. See below for an example.



To make this plot firstly we will have to do some parsing. We need to get the mean expression values across all samples. Try the following code:

```
#We read the master file – note the use of row.names = 1:
x <- read.table("PATH TO YOUR MASTER FILE", header=TRUE, row.names = 1)

#We create a separate table of only the expression values:
FPKM_columns <- c(COMMA SEPARATED LIST OF ALL THE FPKM COLUMNS)
```

```
xFPKMs <- x[,FPKM_columns]

#We generate a mean expression column but save it into original table (clever eh?):
x$row_means <- rowMeans(xFPKMs, na.rm = FALSE, dims = 1)
```

Make sure that you understand how this works. Here we call the column which contains the mean expression values per gene “row_means”, but you can change this to anything you like. Try calling it “mean_expression”.

Next we can try the plot, modify the following code:

```
#We plot the MA:
png("PATH TO OUT{UT YOUR MA PLOT TO.png", height=500, width=500, pointsize=10)
ggplot(data=x, aes(x=mean_expression, y=YOUR LOG2 FOLD CHANGE COLUMN)) + geom_point(size=1.8)
dev.off()
```

Once again, we can improve the plot by logging the mean expression values, try:

```
ggplot(data=x, aes(x=log10(YOUR MEAN EXPRESSION COLUMN + 0.001), y=YOUR LOG2 FOLD CHANGE COLUMN))
```

Great. Our next step is to colour the dots by significance. There are many ways to do this. In this example we will create a new column in the master file that flags significance as “true” or “false”, then we will supply this column to the ggplot “colour” parameter. Modify the following code into your own:

```
x$sig_flag <- as.factor(x$YOUR ADJUSTED P VALUE COLUMN > 0.05 | abs(x$YOUR FOLD CHANGE COLUMN) < 1.0)
ggplot(data=x, aes(x=log10( YOUR MEAN EXPRESSION COLUMN + 0.001), y=YOUR LOG2 FOLD CHANGE COLUMN,
colour=sig_flag))
```

Make sure that you understand how this works. Its the same idea as the PCA plot.

Now we are ready to add some data labels. It can be useful to see which genes are highly expressed or have large fold changes. We can use the extra “grob” geom_text() to add data labels. Try the following:

```
+ geom_text(data=x,aes(label=rownames(x))
```

One issue is that we generally don't care about the names of the non-significant genes. There's enough information on the plot already without having them confuse things further. Next we will re-plot only with labels for the significant genes. To do so we will supply the geom_text function with a list of only the significant gene Ids. It will automatically know which dots to label. Modify the following code:

```
sigGenes <- subset(x, x$ADJUSTED P COLUMN < 0.05 & abs(x$LOG2 FOLD CHANGE COLUMN) > 1.0)
 + geom_text(data=sigGenes,aes(label=rownames(sigGenes)),hjust=-0.1, vjust=-0.1, size=4.5)
```

Notice we added some new parameters too: hjust, vjust and size. These can be used to control the location of the text realtive to the dot, and the text size. Try changing these values and see what happens.

Now we can try tweaking the visuals, as before. Try some of the following:

```
+ geom_hline(aes(yintercept = 0), colour = "black", size = 0.75)
+ xlab("Log2 Mean Expression")
+ ylab("Log2 Fold Change")
+ theme_bw()
+ theme(legend.position="bottom",legend.title = element_blank())
```

```
+ scale_colour_discrete(breaks=c("FALSE", "TRUE"),labels=c("Significant", "Not-significant"))
```

Make sure you know what each of these is doing, and why. You can modify any of the axis labels or legend lables as you see fit. Note: you will have to make two plots – one for each of your fold change columns. You can use the same mean values column each time.

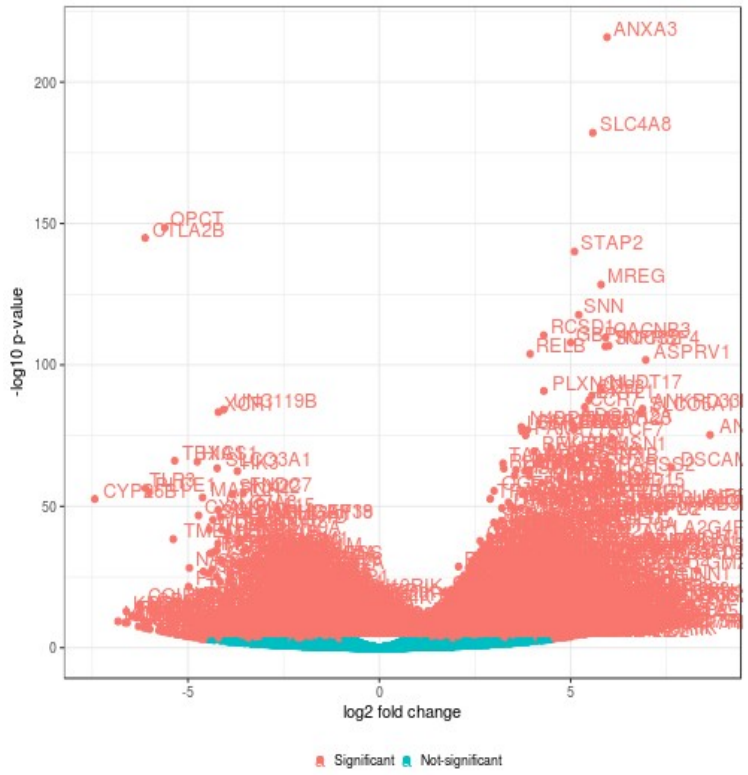
Finally, write a couple of sentences interpreting both of your plots. What does it show? Are more highly expressed genes more likely to be significant? Does the expression level effect the fold change? We can discuss this in more detail later.

Task 4: Volcano plots

Question: Is there a relationship between the p-value and fold change? Are genes with a greater fold change more likely to be differentially expressed?

In a volcano plot we want to get a feel for the relationship between fold change and p-value. In essence genes with larger fold changes are changing more, and so should be more significant. Sometimes it is possible to see different populations of genes: those with low fold change but high significance, and those with high fold change, but lower significance. This can tell us about the behaviour of the genes biologically. We can talk about this more later.

A volcano plot is quite similar to an MA plot. The difference being log2 fold change is plotted on the x-axis (instead of the y-axis) and instead of mean expression, we plot -log10 p-value (on te y-axis). See below for an example volcano plot:



Why do we use -log10 p? Well, if we plot p-value smaller numbers will be more significant and the plot will look odd. P-values can range from 1 to 1x10 to the power of minus infinity. Effectively, all of the significant genes will be squished on the y axis from 0.05 to 0, whereas the non-significant genes will “luxuriate” in a range of 0.05 to 1 (19x more space). As we are interested mainly in the significant genes, we want to reverse this.

Modify your MA plot code to produce a volcano plot. Try it with and without -log10 for your p-value. Get a feel for what -log10 does.

Finally, write a couple of sentences interpreting both of your volcano plots. What does it show? How does fold change relate to the p-value? We can discuss this in more detail later.

RNA-seq high level analysis, practical session 3

Introduction

This session is a continuation of session 2. We will use the same data to produce some more plots, and expand our knowledge of ggplots and our understanding of the data.

During this session we will make the following plots:

- Bar charts showing the number of up and downregulated genes
- Heatmaps of the significant genes
- Heatmaps of the non-significant genes
- Violin plots of the 10 most up and downregulated genes

You will make several of these plots for both of the differential comparisons.

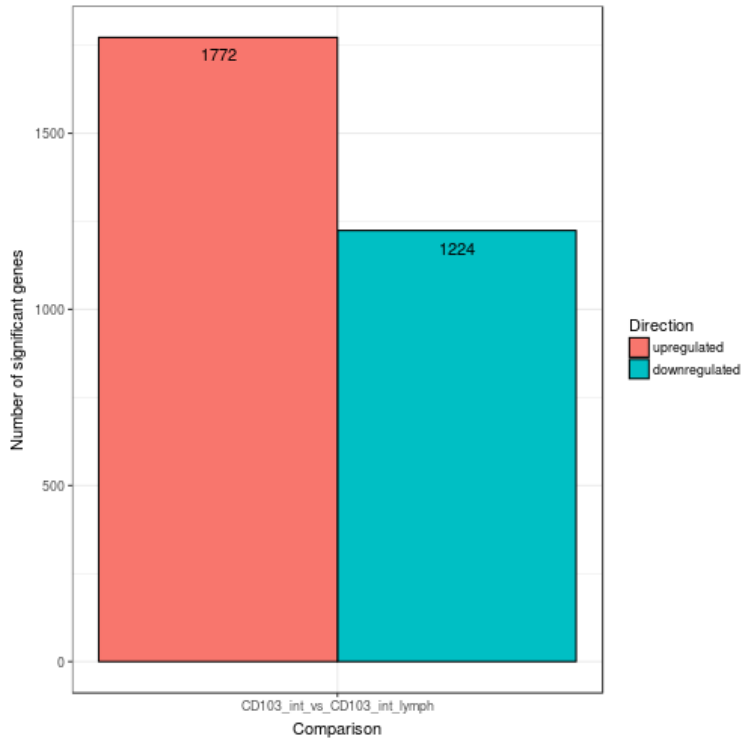
You will need the following R libraries:

```
require(ggplot2)
require(RColorBrewer)
require(fastcluster)
require(reshape2)
require(amap)
```

Task 1: Bar charts of the number of significantly up and down-regulated genes

Question: How many genes have been up or down-regulated? Are more genes up or down-regulated?

Never underestimate how important the simple things are. Simply knowing, and being able to communicate how many genes are changing tells us a LOT about our experiment. Remember, people are very visual. A bar chart of two numbers will communicate far better than just showing the numbers.



Our first task is to make a bar chart, showing the number of up and down-regulated genes. Our first step is to use R to count the number of up and downregulated genes Open (using R) your files of up and down-regulated genes that you created in task 4 of session 1. Use the following method to count the number of rows in each file (i.e. the number of significant genes). Store the counts into an appropriately named variable. You should end up with 4 count variables (two for each differential comparison).

```
x <- read.table("PATH TO YOUR UPREGULATED GENES FILE.csv", header=TRUE, row.names = 1)
upregulated_count <- nrow(x)
```

Our first problem is that we can't individually supply the number of signficantly up and down-regulated genes to ggplots, in order to make our bar chart. Ggplots is quite strict about the formats it accepts, and particularly so when making bar charts. Lucky us! We will need to make a table of our results using the data.frame() method.

Lets think a bit harder about what we need to create our bar chart in ggplots. We want to plot our two

bars as separate colours (red for up and blue for down-regulated). We also want ggplots to recognise that the two bars belong to the same comparison (this will become useful later on). We therefore need to supply to ggplots:

x axis = the comparisons
y axis = the number of genes
fill colour = the direction

We need to make a table of our results where each row can communicate this information. i.e. we need three columns.

Column 1= name of the comparison
Column 2 = the direction of change (up or down-regulated)
Column 3 = the number of significant genes in that comparison, with that direction

Here is some example code to make a new data.frame() with the type of columns and information you that need to make your bar chart. Modify it to include your own data. Note: you will need to make one data.frame for each differential comparison.

```
up_count = 1456
down_count = 3482

comparison = c("a_vs_b", "a_vs_b")
direction = c("upregulated", "downregulated")
number_of_sig_genes = c(up_count, down_count)
dat = data.frame(comparison, direction, number_of_sig_genes)
```

For more information on data.frames() see here: http://www.r-tutor.com/r-introduction/data-frame

You can name your columns anything you like. But its best to keep it appropriate. Make sure that you understand the format of your data.frame(), and why it needs to be this way, before you continue.

Now we are ready to plot. Modify the following code to generate your bar charts:

```
#We plot:
png("PATH TO OUTPUT YOUR PLOT  TO.png", height=500, width=500, pointsize=10)
ggplot(data=dat , aes(x=comparison, y=number_of_sig_genes, fill=direction)) + geom_bar(colour="black", stat="identity", position =
"dodge")
dev.off()
```

What do the parameters “stat” and “position” do? Try to modify them to see what happens to the plots. For more type: ?geom_bar into the R console.

Now we are ready to tweak the visuals. Try the following:

```
+ ylab("Number of significant genes")
+ xlab("Comparison")
+ scale_fill_discrete(name = "Direction")
+ theme_bw()
+ geom_text(aes(label=value), position=position_dodge(width=0.9), vjust=2)
```

Make sure that you know what each of the above does before moving on.

Now, we can improve our bar charts by merging them into a single plot. This will also test how well you understand the data.frame() that you created, and how it relates to ggplots. Your task is to make a single plot with four bars, two for each comparison. Both the upregulated bars MUST be red, and both the downregulated bars MUST be blue. The two bars for each comparison must be side by side. Its actually very easy.

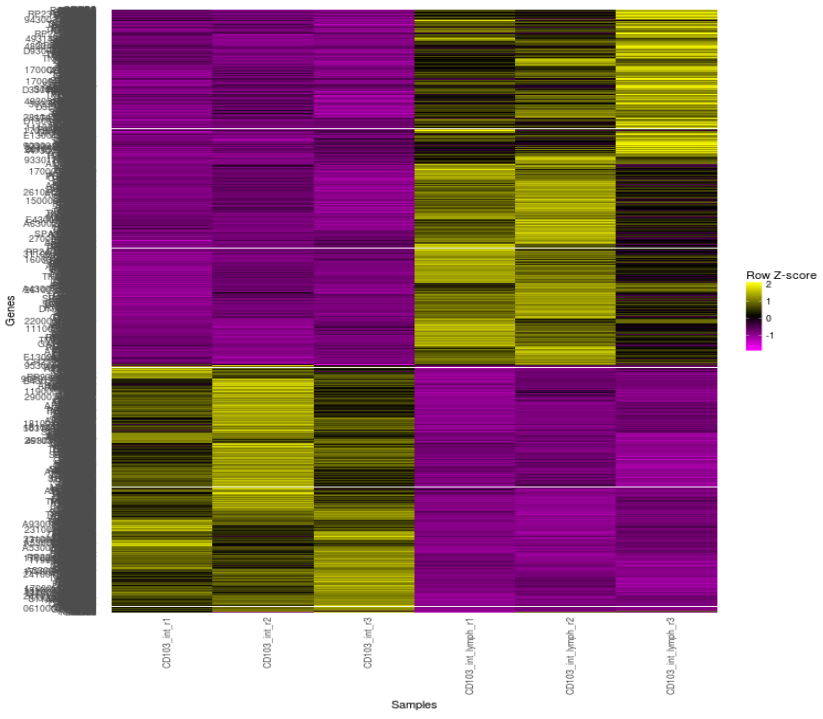
Hint: make a data-frame with 4 rows.

Finally, write a couple of sentences interpreting your bar chart. What does it show? Biologically, which part of the migration (intestine to lymph duct, or lymph duct to lymph node) has the largest change in transcription? Can you relate this to the PCA plot from last session?

Task 2: Clustered heatmaps of the significant genes

Question: For the significantly different genes, do the expression values for the replicates look consistent? Do the sample groups being compared look different? Are there any hidden expression profiles in the data?

Heatmaps are probably the single most useful plot type in RNA-seq high level analysis. A heatmap is effectively a table of expression values where the value is replaced with a colour (or colour scale). The intensity of the colour (or colour scale) represents the expression value. The columns of a heatmap are typically the samples, and the rows the genes. In all other plot types it is necessary to "compress" the data in some way, either by averaging or some other statistical conversion. In a heatmap there is no conversion, you can quickly see every single gene, in every single sample. That said, there are two issues with heatmaps. Firstly with 100s or 1000s of genes it can be hard to visually tell individual genes apart. Secondly the eye cannot accurately tell several colour intensities apart, or how far apart they are fro each other. For these reasons, typically heatmaps are used to show the general pattern of expression, across all of your significant genes, and so are highly useful for examining patterns of expression across many conditions. Here is an example of a heatmap:



Before we use ggplots to make our heatmaps (one for each differential comparison), we need to do some parsing in R. Firstly we need to load our version of the master file that contains only the significant genes. For now we are only interested in the significant genes. We can plot both up and down-regulated genes together on the same heatmap.

```
x <- read.table (file= "PATH TO YOUR SIG GENES MASTER FILE", header=TRUE,row.names = 1)
```

The heatmap generating function in R can often crash due to high memory usage, if there are too many rows (genes). We will need to ensure there are fewer than 5,000 rows (genes) in our file, and if not

select 5,000 at random. To do so we use an R if statement to test if there are more than 5,000 rows in our master file. If so, we then use the function sample() to select 5,000 random genes. Use the following piece of code:

```
#If there are more than 5,000 significant genes, we select 5,000 at random:
if (nrow(x) > 5000) {
x <- x[sample(1:nrow(x), 5000,replace=FALSE),]
}
```

For more information please see: https://www.datamentor.io/r-programming/if-else-statement
and: https://www.rdocumentation.org/packages/base/versions/3.4.3/topics/sample

Make sure that you are comfortable with this piece of code before proceeding.

Now we come to our first problem. Unlike our previous plots - where we used one column from our master file to be the x-axis, or one column to be the y-axis, or one column to be the colour, in our heatmap we need to use 6 columns to be the x-axis, and the gene Ids to be our y-axis. As previously mentioned, ggplots is quite strict about the format that it accepts.

GG plots effectively sees the x-axis and y-axis values supplied as a coordinate. This can either be a continuous or a discontinous coordinate. So for the MA scatter plots the x and y coordinates for each dot were the expression level and the fold change. For the bar charts the x and y coordinates for each bar were the comparison and the number of significant genes. For our heatmaps we need to supply:

x axis = the samples (i.e. the samples are the coordinates for the x-axis)
y axis = the genes (i.e. the genes are the coordinates for the y-axis)
fill colour = the value for that gene in that sample (i.e. colour for position x/y in the heatmap)

We need to make a table of our results where each row can communicate this information. i.e. we need three columns.

Column 1= name of the sample
Column 2 = the name of the gene
Column 3 = the expression value for that gene in that sample.

To do this we need to make two steps. Firstly, select only the columns from our significant genes master file that we need. Secondly, use the highly useful R function melt() to convert the data into the above format.

Firstly, select only the 6 columns in the significant genes master file that contain the FPKM values for the samples used in your differential expression comparison. Modify the following code, that we first saw in part 5 of session 1:

```
#We select only the columns with the expression values:
FPKM_columns <- c(COMMA SEPARATED LIST OF THE RELEVANT FPKM COLUMNS)
x <- x[,FPKM_columns]
```

Secondly, apply the melt function to your FPKMs. Melt will magically convert your data.frame into the format required for ggplots. In order for melt to include the gene Ids, you will need to first convert your data.frame to a matrix. Modify the following code:

```
#We melt the data:
x.m <- melt(as.matrix(x))
```

For more information please see: https://www.statmethods.net/management/reshape.html

Make sure that you understand the melt function, and what it does before you move on. Write a short description of the melt function (one paragraph).

Great, now you are ready to make the plot. Modify the following code to generate a heatmap:

```
#We plot:
png("PATH TO OUTPUT YOUR HEATMAP TO.png", width = 750, height = 750, pointsize=10)
ggplot(x.m, aes(x = Var2, y = Var1, fill = value)) + geom_tile()
dev.off()
```

Now we come to our second problem. You may notice that your heatmap doesn't look very good. Its all blue, and almost uniform in colour. Why is this? This is because you have plotted expression level. For every gene, all one the same scale. Think, if one gene on the plot has an expression of 1 million, the brightest blue will represent expression of 1 million. This will make it impossible to see any difference in a gene with expression of 10 that goes up by two fold. i.e. on a scale of 0 to 1 million a change from 10 to 20 will be imperceivable. To get around this problem, we need to scale the expression values for each gene. To do so we convert the values to z-scores. Z-scores effectively standardise the size of the expression (across all genes), whilst maintaining the proportion of change between samples for each gene. Please see for more information: http://stattrek.com/statistics/dictionary.aspx?definition=z%20score

To scale our data in R, we can use the scale() function. One issue is that scale() only scales by column, in our case we want to scale by row (gene). To get round this we use the transpose function t(), to flip the rows and columns. We scale, and then flip back. Add the following piece of code into your own, and modify so that x.scale is melted instead of x.

```
#We scale the data:
x.scale <- t(scale(t(x)))
```

Now, your plot should look a little better, but still not quite right. The genes on the plot are still in the same order as in the master file. i.e. The up and down-regulated genes are all jumbled up. We will now need to add a clustering step. Clustering in R re-orders the rows of a data.table to place the most similar rows close together, and the most different far apart. We can then use the clustered order to plot a nicer and more structured looking heatmap. To cluster the data we will use the hclust() function, with the distance method “spearman”. Modify your own code to include the following step, after the scaling but before melt:

```
row.order <- hclust(Dist(x.scale, method ="spearman"), method="average")$order
x.scale.clustered <- x.scale[row.order,]
```

Make sure that you can appreciate what the above code does, particularly the $order and row.order parts.

Now your plot should look nice. You can now try some extra visualisation methods. Firstly change the colour of the plot. Try modifying you code to include the following:

```
#We set the colour range:
hm.palette <- colorRampPalette(c("magenta", "black", "yellow"))

+ scale_fill_gradientn(colours = hm.palette(100), name="Row Z-score")
```

You can try some colours of your own. For example the “classic” red, white blue.

Next try the following ggplot options:

```
+ ylab('Genes') + xlab('Samples')
+ theme_bw()
+ theme(axis.text.x = element_text(angle = 90, hjust = 1), axis.ticks=element_blank(),plot.background =
element_blank(),panel.grid.major = element_blank(),panel.grid.minor = element_blank(),panel.border = element_blank())
```

Make sure that you know what each of these extra options does before moving on.

If you haven't already, plot a second heatmap, for the significant genes from the second differential comparison. Note: you may have to change the sample columns that you select.

Finally, write a couple of sentences interpreting your two heatmaps. Are the replicates similar at the significant genes? Are the sample groups very different? How does it relate to the bar chart in task 1? Did you notice that heatmaps look simple, but are actually quite complex?

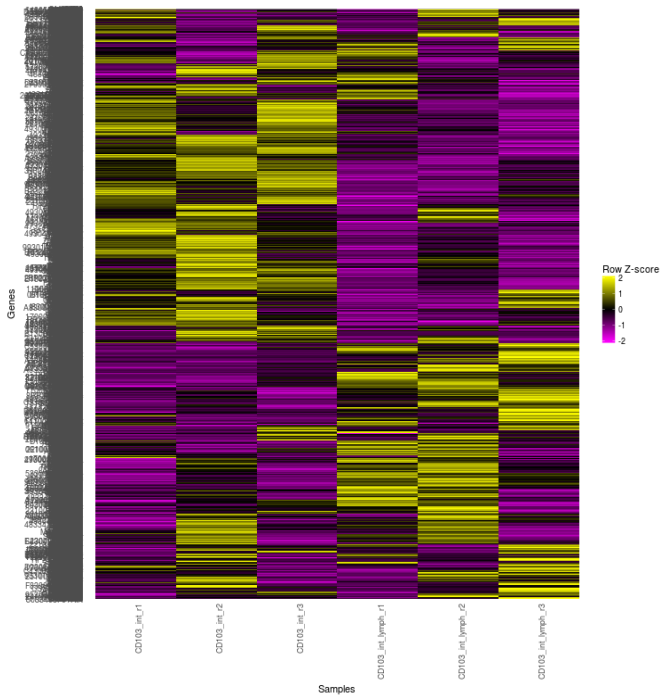
Well done. This task has maybe been a bit tricky to understand. You have certainly earned a short break. Are you back? Ok, lets continue, it gets easier - we promise.

Task 3: Clustered heatmaps of the non-significant genes

Question: Have we set an appropriate cut-off for significance? Do non-significant genes look non-significant?

Though a p value cut-off of 0.05 is widely used and accepted in biology it is a fairly arbitrary value. A p-value cut-off can actually be anything that you like. Keeping in mind that the lower the cut-off the lower the chance of a false positive result (which is good), but the higher the chance of a false negative results (which is bad). With 60,000 genes in an RNA-seq experiment we want to make sure we strike a good balance between false positives and false negatives.

A useful and practical method for seeing whether we have achieved this is to plot a heatmap of a few thousand non-significant genes. If we have chosen a good cut-off for significance we should see a random pattern amongst these genes. See below for an example:



Your next task is to plot a heatmap of a few thousand non-significant genes, for each of the two comparisons. You should have two heatmaps at the end of this task. You should already have the skills to do this. Please refer to your code from session 1 (task 4) and from session 3 (task 2) to aid you.

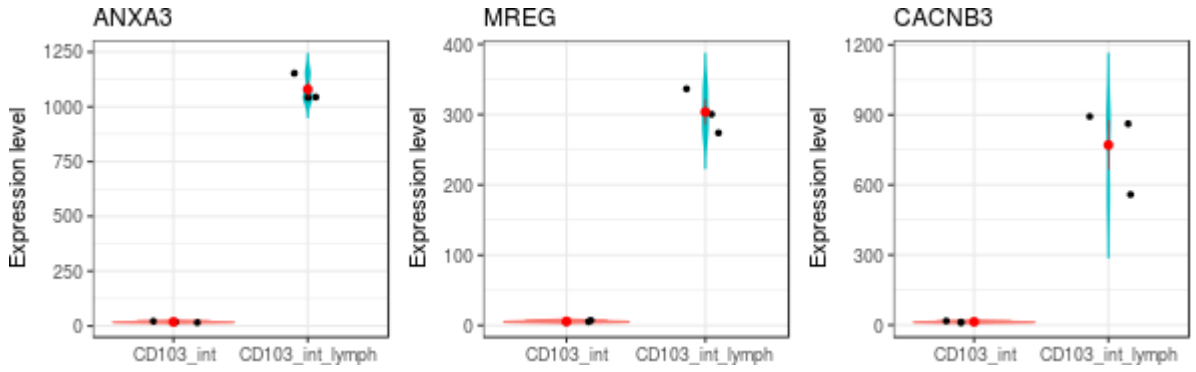
Once you have finished your plots, write a couple of sentences interpreting them. Do you think our threshold for significance is appropriate for this data?

Task 4: Violin and jitter plots of the top differential genes

Question: What are the most highly up and down-regulated genes? Do they look convincing when we look at them individually?

So far we have made several different plots that look at many genes at once. I.e. a global view of transcriptional change. However it is also extremely useful to consider individual genes. Particularly, the genes that change the most. Under the logic that the genes that change the most are often more central to biological system being investigated than the genes that change the least.

To view individual genes we will use violin plots. Violin plots are highly similar to box-plots, except that they show the shape of the distribution, rather than just a box indicating where most samples lie. It can also very useful to see the individual samples. To do this, we use a jitter plot. A jitter plot is simply a scatter plot where the x-axis is discontinuous. As violin and jitter plots show very similar information, and have exactly the same axis, it is possible (and highly useful) to plot both together. Please see below for an example:



Firstly we will find our most significantly up and downn-regulated genes. Using R open your master file of significantly upregulated genes for one of your comparisons. Next sort the rows by ascending p-value (so the lowest p-value is at the top – i.e. the most significant). To do this we use the order() function. Modify the following code:

```
table <- read.table (file= "/PATH TO YOUR SIG GENES MASTER FILE", header=TRUE)
table_ordered<- table[order(table$FOLD,decreasing=FALSE),]
```

In this code we are applying a function to the rows of a table. The square brackets are a way of applying functions to the rows or columns of a table. For example:

table[rows,columns]
table[function(),] = applies the function to all the rows of the table
table[,function()] = applies the function to all the columns of the table

Now that we have sorted our file we can find out what the top 3 most upregulated genes are. The 3 most highly upregulated genes will be found on rows 1-3. If we select row 1, then the gene symbol column, we can get the gene name. Modify the following code to get the names of the three most

highly upregulated genes, storing each in a new variable:

```
gene1 <- table_ordered[ROW NUMBER,COLUMN NUMBER FOR GENE NAME]
```

Excellent. Lets think about what the x and y axis are for our jitter plot. The x axis will be the sample group, and the y-axis the expression level of the gene. Each dot is a sample.

We therefore need to supply to ggplots a table in the following format:
Column 1 = sample group that the sample is in
Column 2 = expression level for the gene in each sample
Rows = samples

This format is very different from our master file. We will need to do some parsing.

1. Firstly we need to trim the sorted master table to include only the top three genes. This makes things more manageable. Make a trimmed table containing only the first three rows.
2. Our trimmed file contains lots of columns that we don't need. Trim your table to remove all of the columns that are not expression values for the relevant samples.
3. Excellent your table is starting to look good, except that it is the wrong way round. Use the t() function to transpose your table. Note: Annoyingly the t() function will convert your data.table to a matrix. You will need to use the as.data.frame() method to cast back into a table.
4. Nearly there. All you need now is a column in your trimmed table saying what each sample group each sample (row) belongs to. Modify the following code to append this column:
   ```
   table_ordered_trimmed$sample_groups <-
   c("CD103_int","CD103_int","CD103_int","CD103_int_lymph","CD103_int_lymph","CD103_int_lymph")
   ```

Well done. Take a moment to compare your original sorted table to your new trimmed one. Appreciate how you have parsed the table into a new format. As a bioinformatician you will do a lot of parsing.

Now we can plot. Modify the following code to make your the plot for your top gene:

```
#We plot:
png("PATH TO OUTPUT YOUR PLOT TO.png",width=600,height=600)
ggplot(table_ordered_trimmed, aes(x=sample_groups,y=table_ordered_trimmed[,1], colour=sample_groups, fill=sample_groups)) +
geom_jitter(size=1, colour="black", width=0.25)
dev.off()
```

Make sure that you understand what the geom_jitter() parameters size, colour and width do before moving on.

Now, we can add the violin plot. A very useful function of ggplots is that where two plots have identical axis, you can plot them on the same image. We will now do this with our jitter plot. Add the following code to your own:

```
 + geom_violin(trim=FALSE)
```

Magic! The violin method converts the dots into a violin plot. Now you can add some extra visuals:

```
 + xlab("")
 + ylab("Expression level")
 + ggtitle(gene_name_1)
 + stat_summary(position=position_dodge(0.75), colour = "red", size = 0.25, geom="pointrange")
```

```
 + theme_bw()
 + theme(legend.position="none",plot.title = element_text(size=12))
```

Well done. Next make 9 more violin plots. 3 for the downregulated genes and then 6 more for the second differential comparison. If you are feeling brave, plot them all onto the same plot using multiplot. This was covered in session 2 (task 1).

Once you are happy with your violin plots, write a couple of sentences interpreting your plots. There might not be too much to say. You could try googling the genes to see what they do.

RNA-seq high level analysis, practical session 4

Introduction

During the previous two sessions we have focused on individual comparisons. I.e. how does lymph duct differ from intestine, or lymph node from lymph duct. It is also very interesting to see how our comparisons compare. i.e. how does the dendritic cell migratory signature differ as the cells progress from the intestine to the lymph duct and finally to the lymph node. Comparing several differential comparisons can often be the most illuminating and exciting part of a project.

During this session we will make the following plots:

- Heatmap of the genes that are significant in either comparisons
- Heatmaps of the genes that are significant in both compsrisons
- Heatmaps of each differential expression profile
- Venn diagrams overlapping the significant genes from each comparison
- A fold vs fold scatter plot

You will need the following R libraries:

```
require(ggplot2)
require(RColorBrewer)
require(fastcluster)
require(reshape2)
require(amap)
```

May the force be with you.

Task 1: Create a master file of the genes that are significant in either comparison

When dealing with multiple differential comparisons, it is often useful to make a file which contains all of the interesting genes. In this case, all of the significant genes. This helps to keep things tidy.

In this task you need to create a master file of the genes that are significant in either intestine vs lymph duct OR lymph duct vs lymph node. This will be used to make a heatmap and in tomorrows pathway analysis practical.

Task 2: Create a master file of the genes that are significant in both comparisons

When dealing with multiple differential comparisons, a second useful file is one which contains only the most variable genes. i.e. those that differ in all comparisons.

In this task you need to create a master file of the genes that are significant in both intestine vs lymph duct OR lymph duct vs lymph node. This will be used to make a heatmap and in tomorrows pathway analysis practical.

Task 3: Make a heatmap of the genes that are significant in either comparison

Question: What do all of the significant genes from each comprison look like across all samples? What general patterns of expression can we observe?

This is usually the most useful plot in a RNA-seq high level analysis. As it lets you see what is going on at all of the interesting genes, in all of your samples.

In this task you need to create a clustered heatmap of the genes in the master-file from task 1. Once you have generated the heatmap, write a few sentences interpreting the plot. Whats going on? Can you see genes with different expression profiles, across the 9 samples? Are there genes required for migration? How does the heatmap relate to the PCA plot from session 1?

Task 4: Make a heatmap of the genes that are significant in both comparisons

Question: What do the most variable genes look like across all samples?

This is also a helpful plot, as it can help us identify candidate useful genes. In this task you need to create a clustered heatmap of the genes in the master-file from task 2. Once you have generated the heatmap, write a few sentences interpreting the plot. Whats going on? How does this differ from the previous task. Which heatmap is better and why? We will discuss this later on.

Task 5: Create master files for each possible differential expression profile

A differential expression profile is simply a pattern of expression across samples. For example: genes that go down in expression from intestine to lymph duct, then increase from lymph duct to lymph node. With two sample groups there can only be two interesting profiles: genes that go up and genes that go

down. However with more sample groups there can be a wider range of profiles. Each profile contains genes that all have a similar behaviour and so are likely functionally related. Determining the genes in each differential expression profiles can help us to further subset our significant genes in a biologically meaningful context. In many cases the whole experiment is centred on finding the genes in just one profile. Each profile is essentially a different biological question.

Your task is to create a separate master file for the genes in each possible profile. Here are the first three profiles. Note there will be 8 profiles of interest:

- Significantly upregulated intestine to lymph duct AND significantly upregulated lymph duct to lymph node
- Significantly upregulated intestine to lymph duct AND significantly downregulated lymph duct to lymph node
- Significantly upregulated intestine to lymph duct AND not significant lymph duct to lymph node

Task 6: Make separate heatmaps for the genes in each differential expression profile

Question: What do the genes with each pattern of expression look like? Are they convincing?

Now that we have identified genes with different types of differential expression profile, it is useful to plot them. Plotting each individually makes each profile stand out more, and so is visually more convincing.

Your task is to create a separate clustered heatmap for each of the differential expression profiles generated in task 5. Once you have generated the 8 heatmaps, write a few sentences describing how overall (not individually) these 8 plots relate to the heatmap generated in task 3.

Task 7: Venn diagram of the significant genes from both comparisons, with overlap statistics.

Question: Is it coincidence that many of the up-regulated genes from intestine to lymph duct are also down-regulated from lymph duct to lymph node, or is it biologically deliberate?

Though we now know that many genes that are upregulated in the lymph duct are switched off again in the lymph node we don't strictly know that this is a “deliberate function of the cell”. For example there might be two systems in play. One is the upregulation of genes from the intestine to lymph duct, and the other the downregulation of genes from the lymph duct to the lymph node. It could be the case that just by total random chance some of the upregulated genes might also be the same genes as some of the downregulated genes. And so, the upregulated then downregulated genes might not really be a biological system of their own, and so might not be very interesting. In order to test this we use a venn diagram and a hypergeometric test. The venn diagram gives us the numbers to use in the test, and the test tells us whether the overlap is random.

Your next task is to create a venn diagram showing the overlap between the significantly upregulated genes from intestine to lymph duct AND the significantly down-regulated from lymph duct to lymph node. To do so we will take a short break from R, and try the highly useful web-tool venny. We could

do this in R (feel free to try later on), but it is useful to learn about venny:

http://bioinfogp.cnb.csic.es/tools/venny/

Use Venny to make the Venn diagram. It should be fairly intuitive. Note: once you have made the diagram, you can click on (or near to) the numbers on the venn diagram to get a list of the genes. You can also right click on the image and “save as” to save the image.

Next we can perform the statistics using R. To do so we use the phyper() function. Please see the following for more informa12tion on this function:

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Hypergeometric.html

Here is an example to get you started. Please substitute your own values in to get the p-value:

```
#Total number of genes in the genome = 1233
#Number of upregulated genes = 134
#Number of downregulated genes = 153
#Number of overlapping genes = 88

#We run the test
phyper(88, 134, 1233-134, 153, lower.tail = FALSE)
```

Great. Now that you have your venn diagram and p-vlaue, write a couple of sentences explaining what the result means. We can discuss in more detail later.

Task 8: More venn diagram of the significant genes, with overlap statistics.

If you feel that you have time, your final task is to create several more Venn diagrams with statistics. Try all combinations of significant genes. i.e. one Venn and p-value for each differential expression profile. Once completed make a table of the p-values. Write a couple of sentences commenting on the biological meaning of the overlap p-values.

RNA-seq high level analysis, practical session 5

Introduction

Now that we have a good understanding of the behaviour of the data across all the interesting genes and all of the samples, our next task is to get a feel for what these genes actually do biologically, and also what might be regulating them. More specifically are there any patterns in the types of genes that are changing that might give us an idea of what biology is being effected by dendritic cell migration.

During this session we use the following methods / tools:

- Ontologicial enrichment using DAVID
- Transcription factor binding site enrichment using OPOSSUM

You will try each of these methods on each of the differential expression profiles that you have generated.

Task 1: Using DAVID to determine enriched gene ontologies

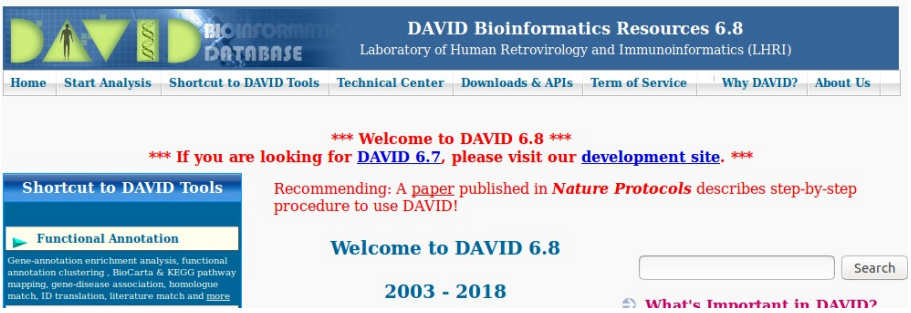
Question: What do the significant genes do biologically? What biological functions are enriched amongst these genes?

With a gene list of several hundred or even thousand genes it is not possible to read the literature on every one. To get round this we use ontology analysis to give us a snapshot of the functions of our genes of interest.

Choose one of your differential expression profiles. Make a file of only the gene Ids. Upload this list of genes to DAVID, and perform a functional annotation analysis. Save the results to a txt fle or spreadsheet.

https://david.ncifcrf.gov/

1) Go-to the DAVID website and click functional annotation:

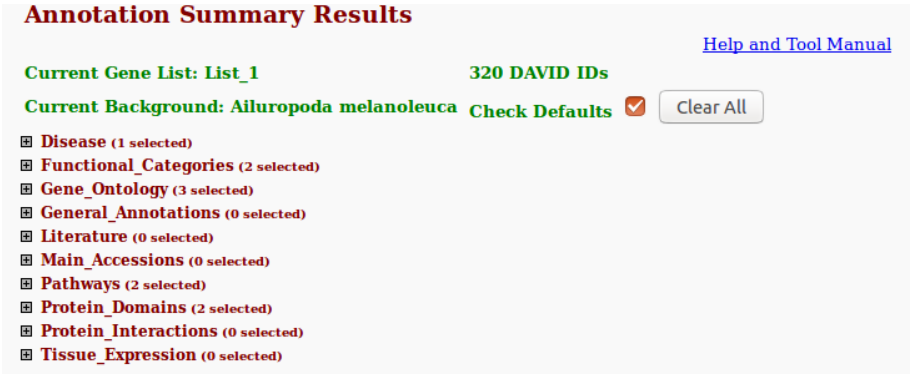


2) Click upload and paste in your list of genes. Just the gene names (or Ids). Select the appropriate type of identifier that you have used. This will be either “OFFICIAL_GENE_SYMBOL” or “ENSEMBL_ID”. Select gene list, then submit.

3) You might have to choose the species type (mus musculus – mouse). You might also (annoyingly) have to select the background to be mouse as well. Otherwise DAVID might compare your data to something else.

4) Click functional annotation chart to see your results. Save the results into a file.

Once completed try playing about with some of the settings. For example, try using different databases:



Once you are happy with the databases being used, re-run DAVID on each of the other 7 differential expression profiles. Save the results for each.

Next summarise each of the sets of results. To do so, make a table of the top 5 most highly enriched gene ontologies for each differential expression profile. Show the gene ontology name, and the FDR value on the table. Write a few sentences, briefly summarising what each of the differential expression profiles does biologically. Broadly speaking, does this make sense in relation to dendritic cell migration?

Task 2: Using OPOSSUM to determine enriched transcription factor binding sites

Question: What might ultimately be regulating the significant genes. Do the promoters of the significant genes contain an enrichment of any transcription factor binding sites?

Where DAVID asks what does a list of genes do, OPOSSUM asks what might be upstream of these genes. Often it is extremely useful to get an idea of what master regulators might be controlling the biological system that we are looking at. OPOSSUM and similar tools (such as HOMER) use the logic that if the promoters of your significant genes have an over-abundance of binding sites for a given transcription factor, then its likely that that transcription factor is regulating these genes in your biologcial system.

Your task is to run OPOSSUM on your gene list. Go-to the OPOSSUM 3 website and select “Single Site Analysis (SSA)” for mouse. Paste in your gene list and click “submit” (its at the bottom). Just use the default settings for now. Note: OPOSSUM can take some time to run. You might want to run it several times simultaneously. Note: OPOSSUM only allows for lists of < 2000 genes. If you find that one of your lists is larger than this, take 1,999 random genes from your list instead.

http://opossum.cisreg.ca/oPOSSUM3/

Once finished save the results. Try playing around with the settings or try a different gene list. Take note of how the results differ. Now run OPOSSUM on each of the other 7 differential expression profiles. Save the results for each.

Next summarise each of the sets of results. To do so, make a table of the top 3 most highly enriched transcription factors for each differential expression profile. Show the transcription factor name, the z-socre and the fisher score. Write a few sentences, briefly summarising what transcription factors might regulate each profile. Broadly speaking, does this make sense in relation to dendritic cell migration? Do these transcription factors agree with the results with DAVID?

CONGRATULATIONS ON FINISHING THIS SERIES ON RNA-SEQ HIGH LEVEL ANALYSIS.

For your interest, here are four further highly useful methods:

Using TRRUST to determine enriched upstream regulators

Question: What might ultimately be regulating the significant genes. Are the significant genes consistent with activation of any specific transcription factors?

Where DAVID asks what does a list of genes do, TRRUST asks what might be upstream of these genes. Often it is extremely useful to get an idea of what master regulators might be controlling the biological system that we are looking at. For this, tools like TRRUST and IPA are very useful. Such tools rely on a comprehensive database of known (from literature) master regulators and the genes that they regulate.

Your task is to run TRRUST on your gene list. Go-to the website, select "search" and fill in the form for "2. Find key regulators for query genes". Note: this method only accepts lists of less than 500 genes. If your list is too large, try selecting 500 random genes from your list. Or if possible the 500 most significant by p-value.

http://www.grnpedia.org/trrust/

Save the results once finished. Note for this task, you only need to run TRRUST on one list of genes. Just to give you an idea of how it works and what it does.

Using consensus path DB to determine networks of genes that interact with each other

Question: How might the significant genes interact with each other? Do the significant genes form any clear pathway?

So far today we have considered the possible functions and regulators of our gene lists, but not how individual genes might interact with each other in a pathway. Tools such as consensus path DB, string and KEGG use databases of known (from literature) gene to gene interactions to construct possible and likely pathways from your list of significant genes.

Goto the consensus path DB website. Click "gene set analysis" and then "induced network modules". Paste your gene list in and select the appropriate identifier. This will be "ensembl" or "gene symbol", then click "proceed". Next click "find induced modules". This will build a gene interaction network.

http://cpdb.molgen.mpg.de/

*Note: this method will struggle to run with a gene list greater than 100 genes. As before, if your list has more than 100 genes, select 100 genes at random from your list, or the 100 most significant.

Once the network is on your screen, save it as an image. Try re-running it using different settings. Note for this task, you only need to run it on one list of genes. Just to give you an idea of how it works and what it does.

Using Gene Set Enrichment Analysis (GSEA) to determine enriched gene ontologies

Question: Is there a difference in the biological functions between the up and down regulated genes from a comparison?

GSEA uses similar databases to DAVID however it uses a very different algorithm to determine enrichment. Where DAVID uses a hypergeometric test, GSEA detects skew towards up or down-regulation based on the p-values. As you must provide p-values to GSEA you can only use it on gene lists generated from a differential comparison, and not those from a differential expression profile.

In GSEA you need to supply not only the gene list, but also the p-values and fold changes.

http://software.broadinstitute.org/gsea/index.jsp

Using HOMER to determine de-novo enriched transcription factor binding motifs

Homer is very similar to OPOSSUM, however it can also be used to find DE-NOVO transcription factor binding sites.

http://homer.ucsd.edu/homer/