Workshop from Alignment to Differential Expression

Today we are going to:

- Explore GTF File
- Run featureCounts to get gene counts
- Use edgeR to normalize data and visualization
- Use edgeR to do differential expressed genes
- Differential expressed genes filtering and draw heatmap

Log into BioHPC

First, we will log into the log into a compute node, install a necessary package and then log into a compute node

- Set up a WebGui session on BioHPC
- Log into via the VNC
- Open a terminal window -- you should be in the directory /archive/nanocourse/gene_expr/trainXX
- Copy data from /archive/nanocourse/gene_expr/shared/session2 to your train directory

```
ln -s gene_expr /archive/nanocourse/June2018/trainXX/session2
ls /archive/nanocourse/gene_expr/trainXX
```

Exploring GTF File

- How many lines in the GTF file?
- How would you view the first 5 lines of the GTF file?
- What are the first 5 lines of the file?
- Why might this information be important?
- How many transcripts are there for Tnnt2? (hint: use grep, cut, uniq 9th field (attributes) transcript ids have ENSMUST in transcript_id field semicolon ';' can be used as a delimiter)
- How many transcripts are protein coding (hint: transcript_type)?

Run FeatureCounts to get counts on a file

```
module load subread/1.6.1
featureCounts -p -g gene_name -a session2/gencode.gtf -o heart_e11_5_rep1.cts session2/alignment
s/heart_e11_5_rep1.dedup.bam
```

featureCounts will print statistics on the screen. There is one line in the second box: "Successfully assigned fragments : 376082 (72.3%)"

Depending on how your samples are prepared, this value varies: * If you are using poly-A extraction, we expect this value to be at least 70% * If you are extracting total RNA, this value will be around 50% * When you have a percent less than 40%, you should look into this matter to figure out why

Now Let's take a look at featureCounts results

```
head heart_e11_5_rep1.cts.summary
head heart_e11_5_rep1.cts
```

- What are the counts for the following genes?

- Tnnt2
- Myh6
- Myh7
- Kcng2
- Gja5

Run feature counts on all of the files.

Then we need to combine all the counts into a matrix. We will be using an in-house Perl script "concat_cts.pl" to do this. It takes in the names of all genes (genenames.txt)

There are 2 parameter you need to provide: 1. The output folder -o (here we use the current directory ./) 2. the patter of all of the cts files (.cts)

```
cp session2/genenames.txt .
perl session2/scripts/concat_cts.pl -o ./ session2/counts/*.cts
head countTable.txt
```

Day 2

Before we are ready for differential expression we need to create a design file. Thank full we have already done that for you.

Also lets copy 3 files to your home directory

```
cp session2/genes_simulated_values.tsv /home2/trainXX
cp session2/design_se.txt /home2/trainXX
cp session2/countTable.txt /home2/trainXX
```

Differential expression

Variables and replicates

We are usually interested in *gl erki w*in gene expression. For example, we might want to compare:

- samples treated with a drug to untreated samples
- different tissues from one organism
- samples from different time points
- mutant strains to wild-type strains

or any other difference between samples you can think of. We want to see how genes change in response to these differences.

Each type of difference is called a ***zevmefpi*** , and each possible value of a variable is called a ***gsrhmxmsr*** . For example, if we take samples from three different laboratories, we might have a Laboratory variable, where the conditions are the names of the laboratories.

Question

Each of the bullet points above describes one possible variable. What might be a suitable name for each variable, and what conditions might each variable have? Does the variable have an obvious control condition?

Where we want to test a particular intervention, such as a drug treatment or a mutation, we usually call the untreated, wild type or 'normal' condition the ***gsrxvsp***condition, but not all variables fit this pattern; here, Tissue and Time do not have obvious control conditions.

Because biological samples vary considerably, we need to take multiple samples for each condition so we can attempt to measure this variation. If we take six drugged samples and six untreated samples, we say we have six ***vi t pmgexi w***of each condition.

(It is not necessary for the same number of replicates to be taken for each condition, but it is a good idea to aim for this, so each condition has consistent data.)

Use RStudio on demand

- Set up a RStudio session on BioHPC

Exercise

Suppose we have six replicates of two conditions of a treatment variable, Control and Treated (so twelve samples in all). We measure the abundance of ten genes in each of these twelve samples and get the following results:

```
gene_names = paste("Gene", 1:10)
genes_sim<-read_tsv("genes_simulated_values.tsv") %>% mutate(Gene=factor(Gene, levels=gene_name
s))
```

Plot the gene abundance by gene

```
ggplot(genes_sim, aes(Condition, Abundance, colour=Condition)) + geom_point() + facet_wrap(Gene
~., ncol=5, scales='fixed') + theme_bw() + ylim(0,NA) + guides(colour=FALSE)
```

- Which of these genes would you say are differentially expressed?
- Which of them look biologically interesting?
- Make sure to pay attention to the abundance values. changing scales to be 'free' to see more appropriate scales for each gene individually. Which genes look differentially expressed now?

Absolute and relative differences

Clearly some of our genes are differentially expressed. Gene 1 increases with treatment; Gene 7 decreases. If genes increase in expression, we say they are *yt vi kypexi h*; if they decrease, they are *hs{ r vi kypexi h*.

We can also see that some genes have higher abundances *mn ki r i vep* than others. For example, the absolute abundances for Gene 1 are higher than those for Gene 8.

But differences in absolute abundance may obscure relationships between genes. For example, for both Gene 1 and Gene 8, the Treated abundance is *hsyf pi* the Control abundance - a 2-fold increase. We say the treatment is associated with a *jsph gl erki* of 2 for these two genes.

Fold change is a useful measure, because genes with similar fold changes may be similarly regulated; for example, a set of 3-fold downregulated genes may be repressed by some other 3-fold upregulated gene.

Fold change values are multipliers; if the control gene has abundance 1:

- a 2-fold increase is $1*2$
- a 3-fold increase is $1*3$
- no change is $1*1$

We can represent decreases in a similar way:

- a 2-fold decrease is $1 *536d(sv(* 50.5$
- a 4-fold decrease is $1 *538d(sv(* 50.25$

Let's see how this works in practice.

Exercise

Calculate the gene means and Fold changes

```
genes_means <- genes_sim %>%
  group_by(Gene, Condition) %>%
  summarise(Mean_Abundance = mean(Abundance)) %>%
  spread(Condition, Mean_Abundance) %>%
  mutate(Absolute_Difference = Treated - Control,
         Fold_Change = Treated / Control) %>%
  ungroup()
```

This plot shows the same genes as before, but now with the means for each condition calculated and shown as horizontal lines. We also have a table listing the mean abundance values for each condition, the absolute difference between these means, and the fold change between these means.

```
ggplot(genes_sim, aes(Condition, Abundance, colour=Condition)) + geom_point() + stat_summary(fu
n.y="mean", geom="errorbar", aes(ymax = ..y.., ymin= ..y..), width=0.4) + facet_wrap(Gene~., nco
l=5, scales="fixed") + theme_bw() + ylim(0,NA) + guides(colour=FALSE)
```

Click the variable `genes_means` in `Environment` section. Tjis table can be sorted by clicking on the column names. Try sorting the table above by Absolute Difference or Fold Change and answer the following questions:

1. Which genes have very similar fold changes? How similar are the absolute differences for these genes?
2. Which genes have a two-fold decrease in expression in the Treatment condition?
3. Which genes show little change? How similar are the abundances and absolute differences for these genes? What are the fold change values?

Log fold changes

We are often interested in visualising and filtering our gene sets based on fold change. For example, we might want to select all of the genes that are more than 2-fold up- or down-regulated, and we might want to investigate whether genes with similar fold changes in either direction are regulating each other.

However, look at this plot of fold changes from our gene set:

```
genes_means_logs <- mutate(genes_means, Log2_Fold_Change = log2(Fold_Change), GeneNum=1:10)
```

```
ggplot(genes_means_logs, aes(Fold_Change, y=GeneNum, label=1:10, colour=Log2_Fold_Change)) +
  geom_text() +
  scale_x_continuous(name="Fold Change", breaks=seq(0,5,0.5),limits=c(0,NA)) +
  scale_y_reverse(name="Gene", limits=c(11,0)) +
  scale_colour_gradient2(low="blue", mid="gray60", high="red", limits=c(-2.5,2.5), breaks=c(-2.5
, 0, 2.5), labels=c(" Down", " No change", " Up")) +
  theme_minimal() +
  theme(axis.text.y=element_blank(), panel.grid.major.y=element_blank(), panel.grid.minor.y=elem
ent_blank(),
        legend.title=element_blank())
```

(The vertical positions don't mean anything here, they are just to separate the genes.)

The downregulated genes get clumped together between 0 and 1, and it is hard to see which genes might have 2-fold changes in either direction, because the scale is not symmetric; 2-fold upregulation is at 2, but 2-fold downregulation is at 0.5. It is also awkward that 'no change' is a '1-fold change'.

It would be useful if this plot was symmetric around 0, so that 0 represents no change and a 2-fold downregulated gene has the same magnitude as a 2-fold upregulated gene.

We can achieve this by calculating the logarithms of our fold change values. Logarithms are the inverse of exponents:

- $262 = 2^3 = 8$ (2 raised to the power of 3 equals 8)

- the inverse of this is $log2(8)=3$ (the base 2 logarithm of 8 is 3).

Just as we can raise any number to a power, we can use any base for logarithms; for example, as $10^2=100$, $log10(100)=2$.

However, we typically use log2 fold changes (often abbreviated to log2FC) for gene expression. Here are our genes on a log2 scale:

```
ggplot(genes_means_logs, aes(Log2_Fold_Change, y=GeneNum, label=1:10, colour=Log2_Fold_Change))
 +
  geom_text() +
  scale_x_continuous(name="Log2 Fold Change", breaks=seq(-2,2,0.5), limits=c(-2.5,2.5)) +
  scale_y_reverse(name="Gene", limits=c(11,0)) +
  scale_colour_gradient2(low="blue", mid="gray60", high="red", limits=c(-2.5,2.5),
                         breaks=c(-2, -1, 0, 1, 2), labels=c(" -2", " -1", "  0", "  1", "  2"))
+
  theme_minimal() +
  theme(axis.text.y=element_blank(), panel.grid.major.y=element_blank(), panel.grid.minor.y=elem
ent_blank(), legend.title=element_blank())
```

Now our gene expression is symmetric; 'no change' is 0, and, for example, the 2-fold downregulated (log2FC=-1) Genes 7 and 10 are similarly distant from 0 as the 2-fold upregulated (log2FC=1) Gene 8.

This plot shows how linear fold changes correspond to logarithmic fold changes:

```
loglin<-data.frame(FoldChange=c(0.125, 0.25, 0.5, 1, 2, 4, 8)) %>% mutate(Log2FoldChange=log2(Fo
ldChange))

ggplot(loglin, aes(FoldChange, Log2FoldChange)) + geom_point() + theme_bw() + ylab("Log2 Fold Ch
ange") + scale_y_continuous(breaks=seq(-3,3,0.5), limits=c(-3,3)) + xlab("Fold Change") + scale_
x_continuous(breaks=seq(0,8), minor_breaks=seq(0.5,7.5), limits=c(0,NA))
```

The x-axis here shows fold changes, and the y-axis shows log2 fold changes, calculated by the formula $y=log2(x)$.

Most publications use log fold changes , so build your intuition for them by answering these questions:

- What is the log2 fold change value for a 2-fold upregulation
- What fold change corresponds to a log2 fold change value of -2?
- The fold change for an 8-fold upregulation is 8; what is the fold change for an 8-fold downregulation?
- The log2 fold change for an 8-fold up-regulation is 3; what is the log2 fold change for an 8-fold down-regulation?

Variation in expression

So far we have seen that genes can have small or large overall abundances, and that genes can have small or large relative differences in abundance between conditions (fold changes). But we also need to consider ***zevrexsr*** in gene expression.

Consider Genes 1 and 3 from our dataset:

```
ggplot(genes_sim %>% filter(Gene %in% c("Gene 1", "Gene 3")), aes(Condition, Abundance, colour=C
ondition)) + geom_point() + facet_wrap(Gene~., ncol=5) + theme_bw() + ylim(0,NA) + guides(colour
=FALSE) + stat_summary(fun.y="mean", geom="errorbar", aes(ymax = ..y.., ymin= ..y..), width=0.4)
```

It seems clear that Gene 1 is differentially expressed, but it is not clear whether Gene 3 is. The abundances for Gene 3 are spread over a much wider range, and the means for each condition are within the range for the other condition. We are much less confident that the differences in the means is reflecting some true biological difference. Perhaps if we took another set of samples, the means would be much more similar.

Actually, we can test this. The abundances for Gene 1 and Gene 3 are not from real genes, but are simulated; both genes are programmed to have the same mean values in each condition (Control=100, Treatment=200) but different amounts of variation (Gene 3 is 10 times as varied as Gene 1).

Here's what happens if we simulate 5 genes like Gene 1, and 5 genes like Gene 3:

```
library(tibble)

generep_names = paste("Gene ", rep(c(1,3),each=5), c("A","B","C","D","E"), sep="")
genereps <- tibble(
              Gene=factor(rep(generep_names, each=12), levels=generep_names),
              Condition=rep(rep(c("Control","Treated"), each=6), 10),
              Abundance=c(rnorm(6, 100, 10), rnorm(6, 200, 10), # Gene 1A
                          rnorm(6, 100, 10), rnorm(6, 200, 10), # Gene 1B
                          rnorm(6, 100, 10), rnorm(6, 200, 10), # Gene 1C
                          rnorm(6, 100, 10), rnorm(6, 200, 10), # Gene 1D
                          rnorm(6, 100, 10), rnorm(6, 200, 10), # Gene 1E
                          rnorm(6, 100, 100), rnorm(6, 200, 100), # Gene 3A
                          rnorm(6, 100, 100), rnorm(6, 200, 100), # Gene 3B
                          rnorm(6, 100, 100), rnorm(6, 200, 100), # Gene 3C
                          rnorm(6, 100, 100), rnorm(6, 200, 100), # Gene 3D
                          rnorm(6, 100, 100), rnorm(6, 200, 100) # Gene 3E
                         )
              )
```

Plot

```
ggplot(genereps, aes(Condition, Abundance, colour=Condition)) + geom_point() + facet_wrap(Gene
~., ncol=5, scales='fixed') + theme_bw() + ylim(0,NA) + guides(colour=FALSE) + stat_summary(fun.
y="mean", geom="errorbar", aes(ymax = ..y.., ymin= ..y..), width=0.4)
```

Although the Gene 1 means vary a little, they are much more consistent than the Gene 3 means, which vary a lot, and sometimes are not very different at all. The Treated mean may even be lower than the Control mean in a few cases.

Also, all the abundances for each condition in Gene 1 are well separated, whereas Gene 3's abundances always overlap, and their ranges often include both condition means. We can be confident that the treatment affects Gene 1, but it is not clear that it truly affects Gene 3.

To say whether a gene is differentially expressed, we must take variation into account.

Statistical distributions

We can quantify the variation in abundances for a gene by assuming the abundances are drawn from some known statistical distribution. We can then fit that distribution to our data, and then compare distributions for different conditions to see how similar they are.

For now, we'll assume that our data is drawn from the familiar normal distribution; we'll need to use other distributions later, but the same principles will apply.

The plot below shows the normal distribution that generates the data for Gene 1, along with a set of points that are drawn from the distribution:

```
gene1mean<-100
gene1sd<-10
replicates<-6
dataInput1<-data.frame(x=rnorm(replicates, gene1mean, gene1sd))

ggplot(dataInput1, aes(x)) + geom_point(aes(y=0)) + stat_function(fun=dnorm, n=101, args=list(me
an=gene1mean, sd=gene1sd), lty="dashed") + theme_bw() + scale_x_continuous(limits=c(0,200), brea
ks=seq(0,200,10)) + scale_y_continuous(breaks=seq(0,0.2,0.005))
```

You can adjust the number of replicates here to generate more or less data from this distribution.

What does it mean to draw from the distribution here? The normal distribution is a probability distribution, which means it expresses what the probability of getting each value of x is. The area under the curve sums to 1.

The mean value for Gene 1 is 100, which is where the peak of the curve is found, at a y value of about 0.04. This means that about 4% of the values in the sample will be close to (though not exactly) the mean.

Similarly, the curve passes through 80 and 120 on the x-axis at 0.005 on the y-axis, so we might expect about 0.5% of the data to be around 80, and another 0.5% to be around 120. More distant values from the mean are less likely.

The normal distribution has two parameters, the mean (which controls the location of the bell curve) and the standard deviation (which determines the width of the bell curve). You can explore this in the plot below. As you change the mean and the standard deviation (left sliders), you may need to adjust the range of the x axis and the height of the y axis (right sliders) in order to see the entire bell curve.

Fitting distributions

We have seen that we can think of our gene abundances as being drawn from some distribution with particular parameters. If we knew the true underlying distributions, we could say explicitly what the differences between our genes were.

Here is a plot that shows the actual distributions for our genes and plot for gene 1

```
gene_data<-data.frame(control_mean=c(100, 100, 100, 100, 30, 10, 100, 50, 50, 20),
                      control_sd = c(10, 3, 100, 10, 5, 2, 10, 10, 10, 2),
                      treated_mean=c(200, 100, 200, 100, 10, 10, 50, 100, 200, 10),
                      treated_sd  =c(10, 3, 100, 50, 2, 2, 10, 10, 20, 2))

one_gene<-gene_data[1,]
pop_curves<- data.frame(
    x=rep(seq(-500,1000, 0.1), 2),
    y=c(dnorm(seq(-500,1000, 0.1), one_gene$control_mean, one_gene$control_sd),
        dnorm(seq(-500,1000, 0.1), one_gene$treated_mean, one_gene$treated_sd)
    ),
    Treatment = rep(rep(c("Control", "Treatment"), each=15001))
  )

ggplot(pop_curves, aes(x, y, colour=Treatment)) + geom_line() + theme_bw() +
    xlim(c(min(one_gene$control_mean - 4*one_gene$control_sd, one_gene$treated_mean - 4*one_gene
$treated_sd),
           max(one_gene$control_mean + 4*one_gene$control_sd, one_gene$treated_mean + 4*one_gene
$treated_sd)))
```

ith this information, we can clearly see whether the distributions differ, and what their differences are. (Where only one curve appears, the genes have the same expression - one curve has been drawn on top of the other.)

1. What is the true difference between Gene 1 and Gene 3?
2. Genes 2 and 4 had similar means. How do their distributions differ?

But when we study gene expression, we don't know what the parameters of these distributions are. We only have our expression measures. We have good reason to assume that our data is drawn from some probability distribution. But we have to estimate the parameters of the distribution from our data.

Comparing distributions

So what use is all of this distribution fitting? How does it help us to find differentially expressed genes? Once we have fitted distributions for each condition, we can compare the fitted distributions to see how likely it is they have come from the same population distribution.

We calculate a ***(t (zepyi*** , which is the probability that two fitted distributions have come from the same population distribution. The p value is related to (although not exactly the same as) the amount of overlap between the two distributions; if the distributions have very little overlap, the p value will be low; if they overlap a lot, the p value will be high.

So if the p value is low, we can say the fitted distributions are likely to have come from two different population distributions, and so we can claim that our gene is differentially expressed.

How low? It is conventional to set a p value threshold of 0.05, or 0.01, but these are just conventions; there is nothing magical about them that makes genes with p values of 0.049 significant but genes with p values of 0.051 insignificant.

Practical Differential Expression

Install edgeR for gene differential expression analysis

Install edgeR if not there already

```
source("https://bioconductor.org/biocLite.R")
biocLite("edgeR")
```

For pheatmap package, click "package"->"install" Type 'a' if program asks you if you want to update packages. Install from:CRAN Packages: pheatmap Check "Install dependencies" Click "Install"

Click "Yes" if prompt window asks you if you want to use a personal library.

After everything finished, load the libraries using

```
library("edgeR")
library("pheatmap")
```

Load Data

```
#Read data matrix and sample file

cfile<-read.table("countTable.txt",header=T,row.names=1)
coldata<-read.table("design_se.txt",header=T,sep="\t")

head(counts)
head(coldata)

#Reorder the counts columns to match the order of sample file
cfile = cfile[c("heart_e11_5_rep1", "heart_e11_5_rep2", "heart_p0_rep1", "heart_p0_rep2")]

#It is good to set your control group label as the baseline. Especially you are going to use int
ercept
group = relevel(factor(coldata$SampleGroup),ref="heart_e11_5")
cds = DGEList(cfile,group=group)
```

Pre-filtering the low-expressed genes

Filter for keeping a gene if cpm (counts per million) exceeds 1 in at least 2 samples.

```
cds = cds[ rowSums(cpm(cds)>=1) >= 2, ,keep.lib.sizes=FALSE]
```

Exploratory analysis and some vizulization

Use cpm() function to get log2 transformed normalized counts

```
rld <- cpm(cds, log=TRUE)
```

Calculate the distance between sample pairs and do hierarchical clustering

```
sampleDists = dist(t(rld))
sampleDists
plot(hclust(sampleDists))
```

Use heatmap to show sample correlation**

```
pheatmap(cor(rld))
```

Use MDS plot to check the relationship of replicates.

```
points = c(15,16)
colors = rep(c("red","blue"),4)
plotMDS(cds, col=colors[group], pch=points[group])
legend("bottomleft", legend=levels(group), pch=points, col=colors, ncol=2)
```

Make a design matrix for samplegroups

```
samplegroup <- factor(coldata$SampleGroup)
design<-model.matrix(~samplegroup)
design
```

Normalize data and estimate dispersion. What is the norm.factors per sample?

```
cds = calcNormFactors(cds)
cds
```

Estimate the genewise dispersion estimates over all genes, allowing for a possible abundancetrend. The estimation is also robustified against potential outlier genes.

```
cds$samples
cds <- estimateDisp(cds,design)

cds$common.dispersion

plotBCV(cds)
```

The square root of the common dispersion gives the coefficient of variation of biologicalvariation. Here the common dispersion is found to be 0.0089, so the coefficient of biologicalvariation is around 0.0945.

Now proceed to determine differentially expressed genes. Fit genewise glms: glmFit() and glmLRT() to test for differential expression. What are the top 10 differential genes sorted by logFC?

```
fit <- glmFit(cds, design)
lrt <- glmLRT(fit)
res <- topTags(lrt, n=dim(cfile)[1],sort.by="logFC")
res[1:10,]
```

What are the total number of genes significantly up-regulated or down-regulated at 5% FDR is summarized

```
summary(decideTests(qlf))
```

Make a dataframe with column for genes

```
res_df = cbind(gene_name = rownames(res), data.frame(res))
write.table(res_df,"edgeR.res.tsv",quote=F,sep="\t",row.names=F)
head(res_df)
```

Filter for genes that are logFC >= 1 & FDR <= 0.01

```
res_filt = res_df[(abs(res_df$logFC)>=1 & res_df$FDR<=0.01),]
```

Draw a heatmap of differential expressed genes meeting filter criteria.

```
res_filt_rld = rld[rownames(rld) %in% res_filt$gene_name,]
pheatmap(res_filt_rld,scale="row",show_rownames = F)
```

Write out filtered genes

```
write.table(res_filt,"edgeR.filtered.tsv",quote=F,sep="\t",row.names=F)
```

Write out values of genes that a

```
write.table(res_filt,"edgeR.filtered.tsv",quote=F,sep="\t",row.names=F)
```