Differential Expression

Data

In this practical, as before, we will work with RNA-Seq data from Arabidopsis seeds that matured at standard temperature (ST, 22°C day/18°C night) or at high temperature (HT, 25°C day/23°C night) from a publication by Serin et al. (2017), with three biological replicates per condition. In this exercise we will compare two analysis pipelines to get differentially expressed genes.

Pipeline 1: hisat2, stringie & DESeq2

The first pipeline uses the tools hisat2, stringtie and DESeq2. The reads were mapped to the Arabidopsis genome use the hisat2 program and quantified using stringtie and prepDE.py to get the read counts per gene (Pertea et al. 2016).

We will use the DESeq2 (Love et al. 2014) R package to get the differentially expressed genes between the two conditions.

Installation

Make sure DESeq2 is installed. In R Studio run these two lines to do so:

```
source("http://bioconductor.org/biocLite.R")
biocLite("DESeq2")
```

To load the DESeq2 library and a set of custom functions, run: (choose n to update none)

```
library(DESeq2)
source("http://www.bioinformatics.nl/courses/RNAseq/DEseq2Exercise.R")
```

If you get an error message involving the stringi package, try:

```
install.packages("stringi")
```

The read count like before can be loaded by typing the following R command:

```
expression_data=read.table(
"http://www.bioinformatics.nl/courses/RNAseq/ST_vs_HT.csv",
row.names=1, header=TRUE, sep =",", stringsAsFactors=FALSE)
```

Explanation: row.names = 1 indicates that the first column in the file contains the name of the rows. header = TRUE indicates that file contains a header which in this case contains the names of the columns. sep = "," is there to indicate that the column fields are separated by a comma.

Let's look at a bit closer at the dimensions of the loaded RNA-Seq count data by running:

```
dim(expression_data)
```

Two values will be displayed (after the [1]). These represent the number of rows and the number of columns in our dataset. What do you think the rows represent and what the columns? With the class function you can learn that the data type of expression_data is data.frame: class(expression_data)

By running:

```
colnames(expression_data)
```

you will obtain a list with the column names. In this case the columns correspond to RNA-Seq samples, i.e. ST-3 means: standard temperature, replicate 3.

By running:

```
rownames(expression_data)
```

you will obtain a list with all the row names. Row names correspond to the genes.

By running:

```
summary(expression_data)
```

you get a summary of the data in the different columns. In order to get an indication of the total number of counts in each sample you can run:

```
apply(expression_data, 2, sum)
```

With this command we sum all counts per column.

Cleaning

Now we would like to remove all genes that are not very informative. In this specific case we will remove genes that have in none of the samples more than 10 counts (arbitrary chosen threshold).

Run:

```
mx = apply( expression_data, 1, max )
```

That command will create a list containing the maximum read count (over all our 6 samples) for each gene. Next we will make a new table that only contains rows for which the maximum count is greater than 10:

```
expression_data = expression_data[ mx > 10, ]
```

Use dim(expression_data) to determine how many genes you have left in your set.

DESeq2

In order to continue we need make a data-object that DESeq2 can use for performing differential expression analysis.

Run these commands to create a DESeqDataSet data object:

```
condition = factor(c("ST","ST","ST","HT","HT","HT"),c("ST","HT"))
col_data = data.frame(condition)
dds = DESeqDataSetFromMatrix(expression_data, col_data, ~condition)
```

The col_data parameter indicates that first three columns correspond to replicates from the standard temperature and the last three columns correspond to replicates from the high temperature. The last parameter describes the design of the study.

Normalization

Before any comparison can be made between samples the counts have to be normalized. The reason for this is that the counts for a gene not only depend on its expression level but also on the depth of sequencing (total number of reads per experiment).

Run this to calculate the (linear) correction factors for each sample:

```
dds = estimateSizeFactors(dds)
```

The correction (size) factors can be retrieved using:

```
sizeFactors(dds)
```

In order to show you the importance of normalization: Type the following in the console:

```
norm_versus_non_norm( dds, 1, 2, left = 2, right = 8 )
```

This command calls a custom function from DEseq2Exercise.R script. It takes the first and second column (=sample) of the count dataset and generates two scatter plots, each dot represents the count of a gene in the first and second replicate for the standard temperature. The first scatter plot contains the non-normalized and the second plot the normalized counts. Do you see why normalization is important?

Cluster analysis of the samples

It is very important to check whether your samples cluster as you expect them to. You generally expect the gene expression values to be more similar between replicates than between samples from different conditions. In this section you will perform a simple and quick clustering analysis (more on clustering later in the course).

Before we can cluster, we first have to transform the counts to get a more 'normal' distribution; this avoids highly expressed genes dominating the results. Type the following command in the console:

```
rld = rlog(dds)
```

this creates a table in which the normalized counts are transformed to log counts. Compare the distributions of the values using these commands:

```
plot(density(assay(dds)[,1]), main="counts")
plot(density(assay(rld)[,1]), main="log counts")
```

To compare the six samples, we calculate the (euclidean) distances between the samples using their gene expression values. With this command you create a distance matrix:

```
dists = dist(t(assay(rld)))
```

With this command you plot a tree of the distances between the samples:

```
plot(hclust(dists))
```

This tree represents a hierarchical clustering of the samples. Do they cluster as expected?

Gene-specific dispersions

In order to detect differential expression DESeq2 has to estimate the expression variance for each gene. DESeq2 assumes that gene counts within conditions follow the negative binomial distribution. According to this model the variance in expression of a gene depends on its mean expression-level as follows:

$\sigma^2 = \mu + \alpha\mu^2$

The left term is the variance, which depends on the mean μ. In the formula α is called the dispersion. DESeq2 tries to determine the dispersion value for each gene from the normalized count data. It later will use the dispersions to determine the gene-expression variance for each gene so it can test for differential expression.

To estimate the dispersions, run:

```
dds = estimateDispersions(dds)
```

With this command the gene-specific dispersion values are estimated over all samples. Now type the following command in the console:

```
plotDispEsts(dds)
```

You should see a plot of the dispersion versus the expression level for all genes. The black dots are the dispersion values that were calculated per gene from the normalized count data. As you can see DESeq2 fitted a (red) line through the data. This assumes that the dispersion value is a function of the mean expression value.

The dispersion value for each gene is subsequently adjusted towards the red line, giving the blue dots. The black dots with blue circles are dispersion outliers these are not adjusted.

Differential expression.

We have now arrived at the step where we can perform a differential expression analysis. Type the following command in the console:

```
dds = nbinomWaldTest(dds)
```

This command will perform the differential expression tests between our two samples.
To get a table with differential expression values for the genes, type:

```
res = results(dds)
```

To see the top rows from the differential expression table, type the following command in the console:

```
head(res)
```

The padj column contains p-values that are adjusted for multiple testing. BaseMean lists the mean count values for the six samples and the log2FoldChange is the log2 of the fold change (obviously). For a small number of genes no padj could be calculated, these have no value which will cause problems later, so set them to 1 with this command:

```
res$padj = ifelse(is.na(res$padj), 1, res$padj)
```

Now we write an output a table that you can open in Excel later, type in the console:

```
write.table(res, col.names=NA, row.names=T, file ="expressions.tsv",
sep ="\t")
```

With this command we write table res to disk with the row.names and col.names. We use tabs for separating fields (sep = ""). The file that is created is called: expressions.tsv

MA plot

To end this exercise we will make an MA plot. You can make the plot by typing:

```
plotMA(res, main="MA plot",ylim=c(-8,8),alpha=0.01)
```

Each point in an MA plot represents a gene. The x-coordinate corresponds to the mean expression of the gene, and the y-axis corresponds to the log2 fold change between the two conditions/tissues. All red points correspond to genes with that are differentially expressed according to the adjusted p-value threshold (alpha) of 0.01. What can we say about the effect of seed maturation temperature on gene expression based on these results? Are genes actually expressed in seeds?

Can you get a list of the 10 genes with the highest significant fold change?

How many genes are differentially expressed if you take a cut-off for the adjusted p-value of 0.05?

Pipeline 2: Kallisto & Sleuth

We will now use another pipeline to do a differential expression analysis based on the tools kallisto and sleuth (Pimentel et al. 2017) Download the quantification data for each of the samples in the file ***kallisto.zip*** to your D: drive and unzip it to D:. Make sure to get a directory D:that contains the file design.txt and six directories representing each of the samples.

In R Studio, install Sleuth using these commands:

```
source("http://bioconductor.org/biocLite.R")
biocLite("rhdf5")
install.packages("devtools")
devtools::install_github("pachterlab/sleuth")
```

then load the Sleuth library:

```
library(sleuth)
```

Set the working directory to D:(linux style with a '/'):

```
setwd("D:/kallisto")
```

The count data for each sample is in a separate folder in the kallisto directory.

```
base_dir = "."
sample_ids = dir(base_dir,"_")
kal_dirs = sapply(sample_ids, function(id) file.path(base_dir, id))
```

The design of the experiment (two conditions with three replicates each) is described in the design.txt file. We can load that with read.table:

```
s2c = read.table(file.path(base_dir,"design.txt"), header=TRUE,
stringsAsFactors=FALSE)
```

And add the directories for each sample:

```
s2c$path = kal_dirs
```

Check that the table is correct:

```
print(s2c)
```

Now we can let sleuth load and process the count data:

```
so = sleuth_prep(s2c, ~condition,extra_bootstrap_summary=TRUE)
```

With the data in place we can now fit two models on the data. A 'reduced' model that assumes that there is no difference between the conditions:

```
so = sleuth_fit(so, ~1, 'reduced')
```

And the 'full' model that does assume that a transcript is differentially expressed between the conditions:

```
so = sleuth_fit(so, ~condition, 'full')
```

We can determine for each transcript which of the two models is most likely, using a likelihood ratio test:

```
so = sleuth_lrt(so, 'reduced', 'full')
```

The results can be extracted from the sleuth object with:

```
sleuth_table = sleuth_results(so, 'reduced:full', 'lrt',
show_all=FALSE)
```

To get the transcripts for which the full model is significantly more likely:

```
sleuth_significant <- sleuth_table[sleuth_table$qval<=0.05,]
head(sleuth_significant)
```

Let's check what the expression of one of these transcripts looks like:

```
plot_bootstrap(so, "AT1G56600.1")
```

Do you agree that this transcript is differentially expressed between the standard and the high temperature conditions?

And what about one with a high q-value?

```
plot_bootstrap(so,"AT1G01810.1")
```

The likelihood ratio test does not give a fold change for the transcript, just whether it is differentially expressed or not. Sleuth provides another test called Wald test, which returns a 'b' value that "it is analogous to, but not equivalent to, the fold-change." https://pachterlab.github.io/sleuth/docs/sleuth_results.html

```
so = sleuth_wt(so,which_beta="conditionST", which_model="full")
```

To get the results for the Wald test we can use:

```
sleuth_table <- sleuth_results(so, 'conditionST', 'wt', show_all=FALSE)
```

Sleuth has a nice web interface to view the results, try:

```
sleuth_live(so)
```

This should activate a browser with the results.

But… Sleuth thus far only showed results for transcripts, so how can we compare the results with the DESeq2 results?

To do that we should tell sleuth which transcripts belong to the same gene. We first create a table with two columns: transcript_id and corresponding gene_id. Like for the kallisto excersise before, the gene_id can be obtained by removing the numbers after the dot fromthe transcript_id, i.e.: AT1G56600.1 -> AT1G56600

This R code creates the table:

```
sleuth_matrix = sleuth_to_matrix(so, 'obs_norm', 'tpm')
transcripts = rownames(sleuth_matrix)
t2g = data.frame(target_id=transcripts,
gene_id=substring(transcripts,1,9),stringsAsFactors=FALSE)
```

Now we should run the whole sleuth analysis again to get the gene level results:

```
so = sleuth_prep(s2c, ~ condition,target_mapping = t2g,
aggregation_column = 'gene_id')
so = sleuth_fit(so, ~condition, 'full')
so = sleuth_fit(so, ~1, 'reduced')
so = sleuth_w(so,which_beta="conditionST", which_model="full")
so = sleuth_wt(so,which_beta="conditionST", which_model="full")
sleuth_table <- sleuth_results(so, 'conditionST', 'wt', show_all=FALSE)
sleuth_significant <- sleuth_table[sleuth_table$qval<=0.05,]
```

Now we can compare the results of both pipelines, lets first also get the signifant genes from the DESeq2 results:

```
deseq2_significant = data.frame(res[res$padj<=0.05,])
```

And then merge these into one table, keeping only the common genes:

```
both_significant =
merge(deseq2_significant,sleuth_significant,by.x=0,by.y="target_id",all
=FALSE)
```

Now we can determine the number of shared and unique significant genes:

```
nrow(deseq2_significant)
nrow(sleuth_significant)
nrow(both_significant)
```

What do you think, do both methods agree well enough?

References

Love, M. I., W. Huber and S. Anders (2014). "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2." Genome Biology 15(12): 550.

Yi, L., H. Pimentel, N. L. Bray and L. Pachter (2017). "Gene-level differential analysis at transcript-level resolution." bioRxiv.

Pertea, M., D. Kim, G. M. Pertea, J. T. Leek and S. L. Salzberg (2016). "Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown." Nature Protocols 11: 1650.

Pimentel, H., N. L. Bray, S. Puente, P. Melsted and L. Pachter (2017). "Differential analysis of RNA-seq incorporating quantification uncertainty." Nature Methods 14: 687.

Serin, E. A. R., L. B. Snoek, H. Nijveen, L. A. J. Willems, J. M. Jiménez-Gómez, H. W. M. Hilhorst and W. Ligterink (2017). "Construction of a High-Density Genetic Map from RNA-Seq Data for an Arabidopsis Bay-0 × Shahdara RIL Population." Frontiers in Genetics 8: 201.

Links:

https://pachterlab.github.io/sleuth/
https://bioconductor.org/packages/release/bioc/html/DESeq2.html