Introduction to Bulk RNAseq data analysis

Differential Expression of RNA-seq data - Part 2

Last modified: 28 Nov 2021

Contents

Recap from last week

```
library(DESeq2)
library(tidyverse)
```

```
txi <- readRDS("RObjects/txi.rds")
sampleinfo <- read_tsv("data/samplesheet_corrected.tsv", col_types="cccc") %>%
  mutate(Status = fct_relevel(Status, "Uninfected"))

simple.model <- as.formula(~ Status)

ddsObj.raw <- DESeqDataSetFromTximport(txi = txi,
                                       colData = sampleinfo,
                                       design = simple.model)
```

```
## using counts and average transcript lengths from tximport
```

```
keep <- rowSums(counts(ddsObj.raw)) > 5
ddsObj.filt <- ddsObj.raw[keep,]

ddsObj <- DESeq(ddsObj.filt)
```

```
## estimating size factors

## using 'avgTxLength' from assays(dds), correcting for library size

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
results.simple <- results(ddsObj, alpha=0.05)
```

Exercise 2

So far we have fitted a simple model considering just "Status," but in reality we want to model the effects of both "Status" and "Time Point."

Let's start with the model with only main effects - an additive model with no interaction. The main assumption here is that the effects of Status and the effects of Time Point are indepedent.

Recapitulate the above steps to generate a new DESeq2 object with the additive model. Then we will extract the results table as above.

Load the raw data, remembering to set the factor on the Status so that "Uninfected" will be set as the intercept:

```
txi <- readRDS("RObjects/txi.rds")
sampleinfo <- read_tsv("data/samplesheet_corrected.tsv", col_types="cccc") %>%
                  mutate(Status = fct_relevel(Status, "Uninfected"))
```

```
additive.model <- as.formula(~ TimePoint + Status)
```

Create the model:

```
ddsObj.raw <- DESeqDataSetFromTximport(txi = txi,
                                       colData = sampleinfo,
                                       design = additive.model)
```

Then build the DESeq from the raw data, the sample meta data and the model:

```r
keep <- rowSums(counts(ddsObj.raw)) > 5
ddsObj.filt <- ddsObj.raw[keep,]
```

Filter the data set: You are now ready to run the differential gene expression analysis Run the DESeq2 analysis

1. Run the size factor estimation, dispersion estimation and modelling steps using the `DESeq` command as above.
2. Extract the default contrast using the `results` command into a new object called `results.additive`
   a) What contrast are these results for? If you have constructed the model correctly, then it should be the same as previous `results.simple`
   b) How many genes have an adjusted p-value of less than 0.05

```r
additive.model <- as.formula(~ TimePoint + Status)
ddsObj.raw <- DESeqDataSetFromTximport(txi = txi,
                                       colData = sampleinfo,
                                       design = additive.model)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors

## using counts and average transcript lengths from tximport
```

```r
keep <- rowSums(counts(ddsObj.raw)) > 5
ddsObj.filt <- ddsObj.raw[keep,]
ddsObj <- DESeq(ddsObj.filt)
```

```
## estimating size factors

## using 'avgTxLength' from assays(dds), correcting for library size

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```r
results.additive <- results(ddsObj, alpha=0.05)
```

The default contrast of `results`

The `results` function has returned the results for the contrast "Infected vs Uninfected." Let's have a look at the model matrix to understand why `DESeq2` has given us this particular contrast.

```
model.matrix(additive.model, data = sampleinfo)
```

```
##    (Intercept) TimePointd33 StatusInfected
## 1            1            0              1
## 2            1            0              1
## 3            1            0              1
## 4            1            1              1
## 5            1            1              0
## 6            1            1              1
## 7            1            0              0
## 8            1            0              0
## 9            1            0              0
## 10           1            1              0
## 11           1            1              1
## 12           1            1              0
## attr(,"assign")
## [1] 0 1 2
## attr(,"contrasts")
## attr(,"contrasts")$TimePoint
## [1] "contr.treatment"
##
## attr(,"contrasts")$Status
## [1] "contr.treatment"
```

By default, `results` has returned the contrast encoded by the final column in the model matrix. `DESeq2` has the command `resultsNames` that allows us to view the contrasts that are available directly from the DESeq2 object.

```
resultsNames(ddsObj)
```

```
## [1] "Intercept"                     "TimePoint_d33_vs_d11"
## [3] "Status_Infected_vs_Uninfected"
```

Let's just rename `results.additive` so that we know which contrast results it contains.

```
results.InfectedvUninfected <- results.additive
rm(results.additive)
```

Let's get the top 100 genes by adjusted p-value

```
topGenesIvU <- as.data.frame(results.InfectedvUninfected) %>%
    rownames_to_column("GeneID") %>%
    top_n(100, wt=-padj)
topGenesIvU
```

SAVE SCRIPT

Exercise 3

> If we want a different contrast we can just pass the `results` function the **name** of the contrast, as given by `resultsNames(ddsObj)`. Look at the help page for the `results` command to see how to do this.

1. Retrieve the results for the contrast of d33 versus d11.

```
results.d33vd11 <- results(ddsObj, name= "TimePoint_d33_vs_d11", alpha=0.05)
```

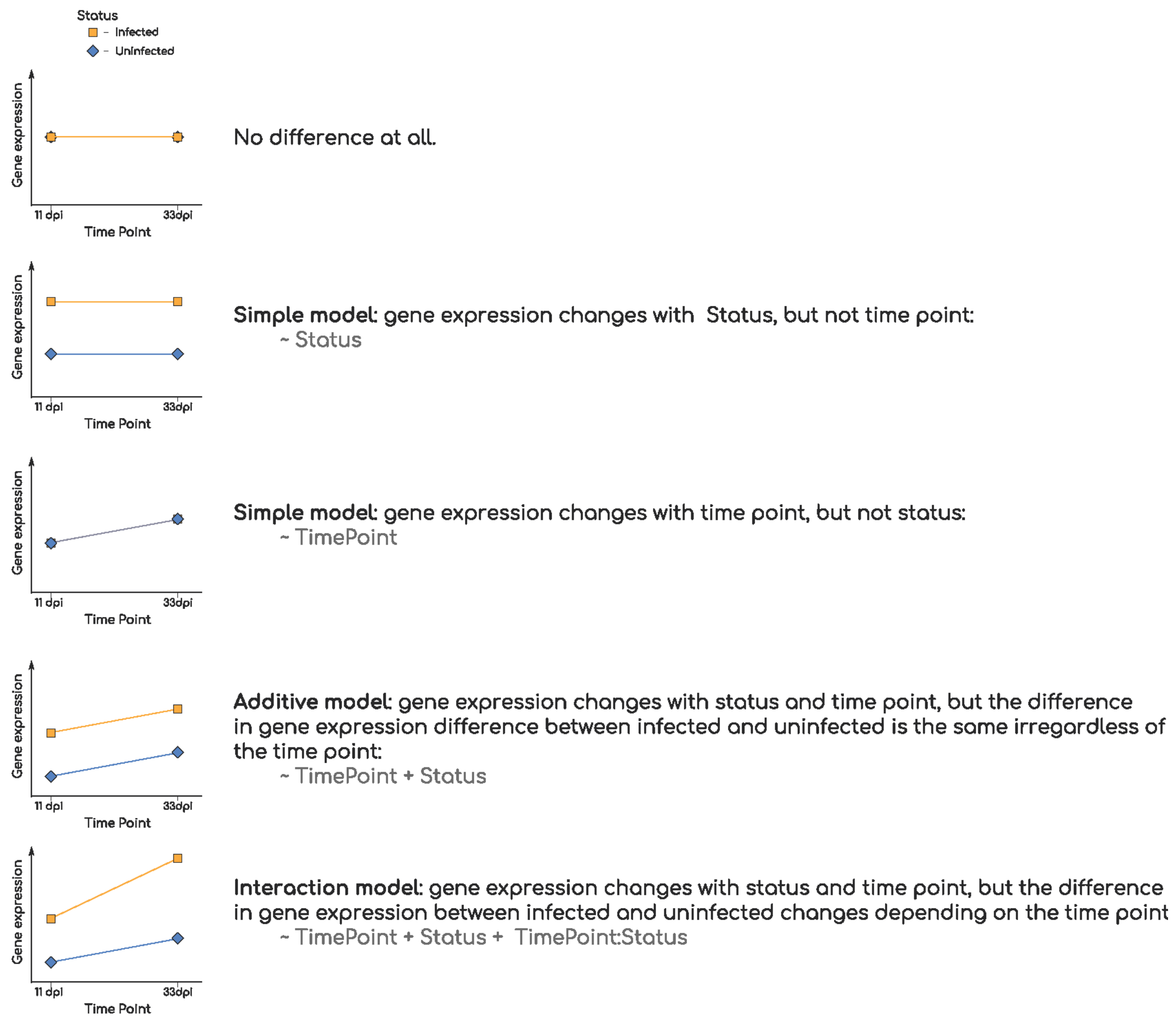
2. How many differentially expressed genes are there at FDR < 0.05?

```
sum(results.d33vd11$padj < 0.05, na.rm = TRUE)
```

```
## [1] 109
```

Should we be using the interaction model?

So far we have modeled gene expression as a function of Status and Time Point with an additive model. Now we are going to look at interaction models and how to decide if we need one.



No difference at all.

Simple model: gene expression changes with Status, but not time point:
~ Status

Simple model: gene expression changes with time point, but not status:
~ TimePoint

Additive model: gene expression changes with status and time point, but the difference in gene expression difference between infected and uninfected is the same irregardless of the time point:
~ TimePoint + Status

Interaction model: gene expression changes with status and time point, but the difference in gene expression between infected and uninfected changes depending on the time point
~ TimePoint + Status + TimePoint:Status

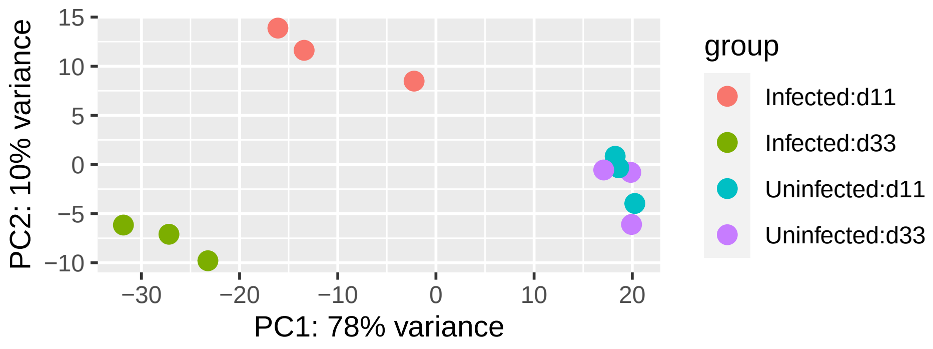
pdf version

Let’s plot a PCA from `vst` transformed data.

```
vstcounts <- vst(ddsObj.raw, blind=TRUE)
```

```
## using 'avgTxLength' from assays(dds), correcting for library size
```

```
plotPCA(vstcounts,  intgroup = c("Status", "TimePoint"))
```



Yes/no Do you think we need an interaction model?

In this case we can, from both the PCA and our understanding of the biology, be fairly certain that the interaction model is the appropriate model to use. This is not always the case and so we need a way to compare two models.

A warning: There are lots of things that you could put into the model, but each extra factor reduces power so you need the simplest appropriate model. How do we know what that is?

SAVE SCRIPT

Comparing two design models

Let's take a simple example to start with.

Suppose we thought that maybe `TimePoint` was irrelevant and really the only differences might be between `Infected` and `Uninfected` groups. We could fit the simpler model and this would give us more degrees of freedom and therefore more power, but how would we know if it was a better model of not?

We can compare two models by using the "likelihood ratio test" (LRT).

To do so we provide the LRT with a simpler model (one with less parameters) than the one currently being used.

Currently `ddsObj` is using the model `~TimePoint + Status`. Here we want to compare to a model without the `TimePoint` parameter: `~Status`, this was our `simple.model` from earlier.

```
ddsObj.LRT <- DESeq(ddsObj, test = "LRT", reduced = simple.model)
```

```
## using pre-existing normalization factors
```

```
## estimating dispersions
```

```
## found already estimated dispersions, replacing these
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
results.Additive_v_Simple <- results(ddsObj.LRT)
results.Additive_v_Simple
```

```
## log2 fold change (MLE): Status Infected vs Uninfected
## LRT p-value: '~ TimePoint + Status' vs '~ Status'
## DataFrame with 20091 rows and 6 columns
##                       baseMean log2FoldChange     lfcSE      stat    pvalue
##                      <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSMUSG00000000001 1102.56094     -0.0110965  0.106195 0.3226536 0.5700173
## ENSMUSG00000000028   58.60055      0.3007930  0.265626 0.0560662 0.8128250
## ENSMUSG00000000037   49.23586     -0.0481414  0.429685 0.0318005 0.8584663
## ENSMUSG00000000049    7.98789      0.4110498  0.656171 0.4373766 0.5083914
## ENSMUSG00000000056 1981.00402     -0.1907691  0.119694 2.7850130 0.0951499
##                          padj
##                     <numeric>
## ENSMUSG00000000001  0.939329
## ENSMUSG00000000028  0.978399
## ENSMUSG00000000037  0.982281
## ENSMUSG00000000049  0.926027
## ENSMUSG00000000056  0.695076
##  [ reached getOption("max.print") -- omitted 6 rows ]
```

header: what are we comparing?

The second line of the results output shows us the test we are doing:

`LRT p-value: '~ TimePoint + Status' vs '~ Status'`

The null hypothesis is that there is no significant difference between the two models, i.e. the simpler model is sufficient to explain the variation in gene expression between the samples. If thats true we might as well use the simpler model and get more power.

If the the adjusted p-value for a gene passes a significance threshold (e.g. padj < 0.05) then we should consider using the more complex model for this gene.

```
sum(results.Additive_v_Simple$padj < 0.05, na.rm=TRUE)
```

```
## [1] 66
```

We can see that for 66 genes the more complex model does fit the data better. Although we have a result for each gene, in practice we should choose one model and apply it to all genes.

Curiously then, this suggests that overall the simple model is more appropriate than the additive model. Let's look into the interaction model.

So what we actually want to test is interaction vs additive and that's down to you!

SAVE SCRIPT

Exercise 4

> When we looked at the PCA it did seem that an interaction model might be warranted. Let's test that.
>
> 1. Create a new DESeq2 object using a model with an interaction between TimePoint and Status. The model formula should be
>
>   `~TimePoint + Status + TimePoint:Status`
>
>   where `TimePoint:Status` is the parameter for the interaction beteween TimePoint and Status.
>
> Note that `*` can be used as shortcut to add the interaction term, e.g. `~TimePoint * Status`, however, writing out in long form is clearer here. > Remember to filter to remove uninformative genes.
>
> 2. Run the statistical analysis using the `DESeq` command and create a new analysis object called `ddsObj.interaction`.
> 3. Use the LRT to compare this to the simpler additive model (`~TimePoint + Status`)
> 4. Extract a table of results using `results`. For how many genes is interaction model a better fit?

```
interaction.model <- as.formula(~ TimePoint * Status)
ddsObj.raw <- DESeqDataSetFromTximport(txi = txi,
                                       colData = sampleinfo,
                                       design = interaction.model)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
## using counts and average transcript lengths from tximport
```

```
keep <- rowSums(counts(ddsObj.raw)) > 5
ddsObj.filt <- ddsObj.raw[keep,]

ddsObj.interaction <- DESeq(ddsObj.filt)
```

```
## estimating size factors

## using 'avgTxLength' from assays(dds), correcting for library size

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

Extracting specific contrasts from an interactive model

If we are settled on using the interaction model, then we need to extract our contrasts with reference to this. That is, we can no longer ask the general question "What is the difference in gene expression between Infected and Uninfected?" but must rather ask two quesions:

- "What is the difference in gene expression between Infected and Uninfected at 11 days post infection?"
- "What is the difference in gene expression between Infected and Uninfected at 33 days post infection?"

If we view the `resultsNames` for the interaction model, we can see the intercept is Uninfected and 11 days post infection:

```
resultsNames(ddsObj.interaction)
```

```
## [1] "Intercept"                     "TimePoint_d33_vs_d11"
## [3] "Status_Infected_vs_Uninfected" "TimePointd33.StatusInfected"
```

The main effect `Status_Infected_vs_Uninfected` is therefore the difference between Infected and Uninfected **at 11 days post infection**.

```
results.interaction.11 <- results(ddsObj.interaction,
                                  name="Status_Infected_vs_Uninfected",
                                  alpha=0.05)
```

To get the results for Infected versus Uninfected at 33 days post infection, we would need to add the interaction term `TimePointd33.StatusInfected`.

SHOW THEM In the help page for `results` it shows us how to do this with a `contrast` in example 3.

```
results.interaction.33 <- results(ddsObj.interaction,
          contrast = list(c("Status_Infected_vs_Uninfected", "TimePointd33.StatusInfected")),
                                    alpha=0.05)
```

Number of genes with padj < 0.05 for Test v Control at day 11:

```
sum(results.interaction.11$padj < 0.05, na.rm = TRUE)
```

```
## [1] 1072
```

Number of genes with padj < 0.05 for Test v Control at day 33:

```
sum(results.interaction.33$padj < 0.05, na.rm = TRUE)
```

```
## [1] 2782
```

We can see that there is a strong difference in the effects of infection on gene expression between days 11 and 33.

SAVE SCRIPT

Exercise 5

Let's investigate the uninfected mice

1. Extract the results for d33 v d11 for Uninfected mice. The the intercept is Uninfected mice at 11 days post infection, so the main effect `TimePoint_d33_vs_d11` is the result that we want.

```
results.d33_v_d11_uninfected <- results(ddsObj.interaction,
                                        name="TimePoint_d33_vs_d11",
                                        alpha = 0.05)
```

How many genes have an adjusted p-value less than 0.05?

```
table(results.d33_v_d11_uninfected$padj < 0.05)
```

```
##
## FALSE  TRUE
## 20043     1
```

Is this remarkable?
Maybe not. Do we really expect vast gene expression differences between the brains of mice that are slightly older than one another? It is possible that there could have been confounding factors, such as changes in enviromental conditions such as temperature or feeding regime, that may have effected gene expression. In which case it was important to set the experiment up with control for both time points.

2. Extract the results for d33 v d11 for Infected mice. The the intercept is Uninfected mice at 11 days post infection, so the main effect `TimePoint_d33_vs_d11` is the result that we want.

```
results.d33_v_d11_infected <- results(ddsObj.interaction,
      contrast = list(c("TimePoint_d33_vs_d11", "TimePointd33.StatusInfected")),
                                        alpha = 0.05)
```

How many genes have an adjusted p-value less than 0.05?

```
table(results.d33_v_d11_infected$padj < 0.05)
```

```
##
## FALSE  TRUE
## 16573  1134
```

Do these results suggest another approach to analysing this data set?

Could we possibly treat the six uninfected samples as a single group with six replicates and then just have 1 factor with 3 levels: Control, d11.Infected, d33.Infected? This is really a biological question and not a statistical one.