Introduction to DGE

View on GitHub

Approximate time: 30 minutes

Learning Objectives

- Executing the differential expression analysis workflow with DESeq2
- Constructing design formulas appropriate for a given experimental design

Differential expression analysis with DESeq2

The final step in the differential expression analysis workflow is **fitting the raw counts to the NB model and performing the statistical test** for differentially expressed genes. In this step we essentially want to determine whether the mean expression levels of different sample groups are significantly different.

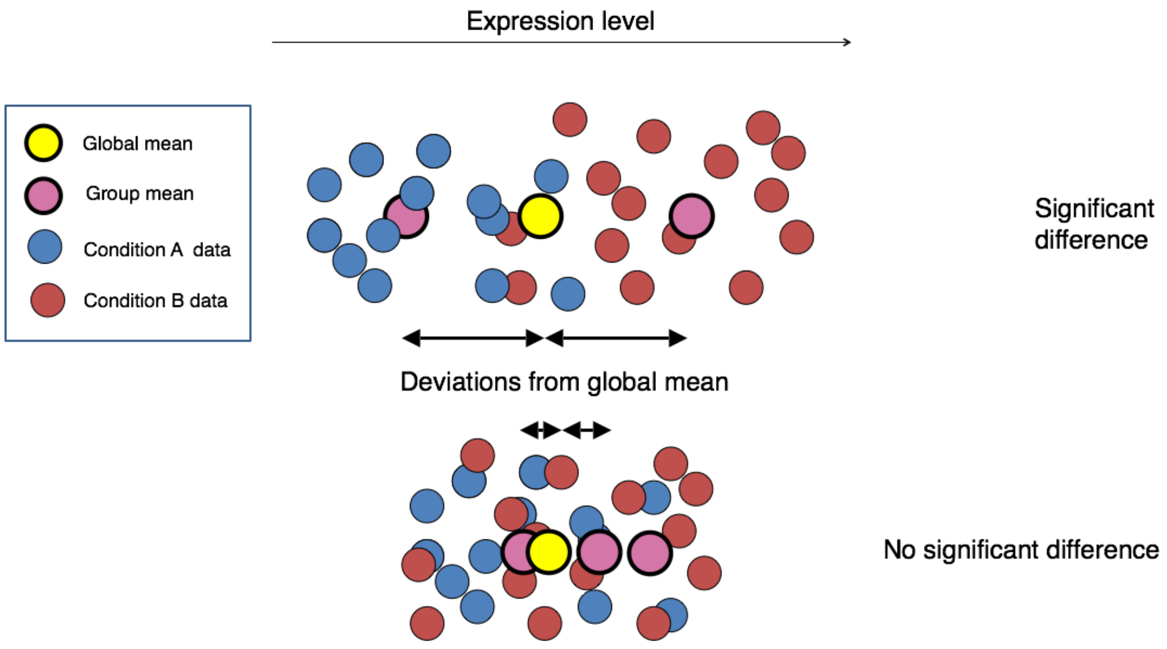
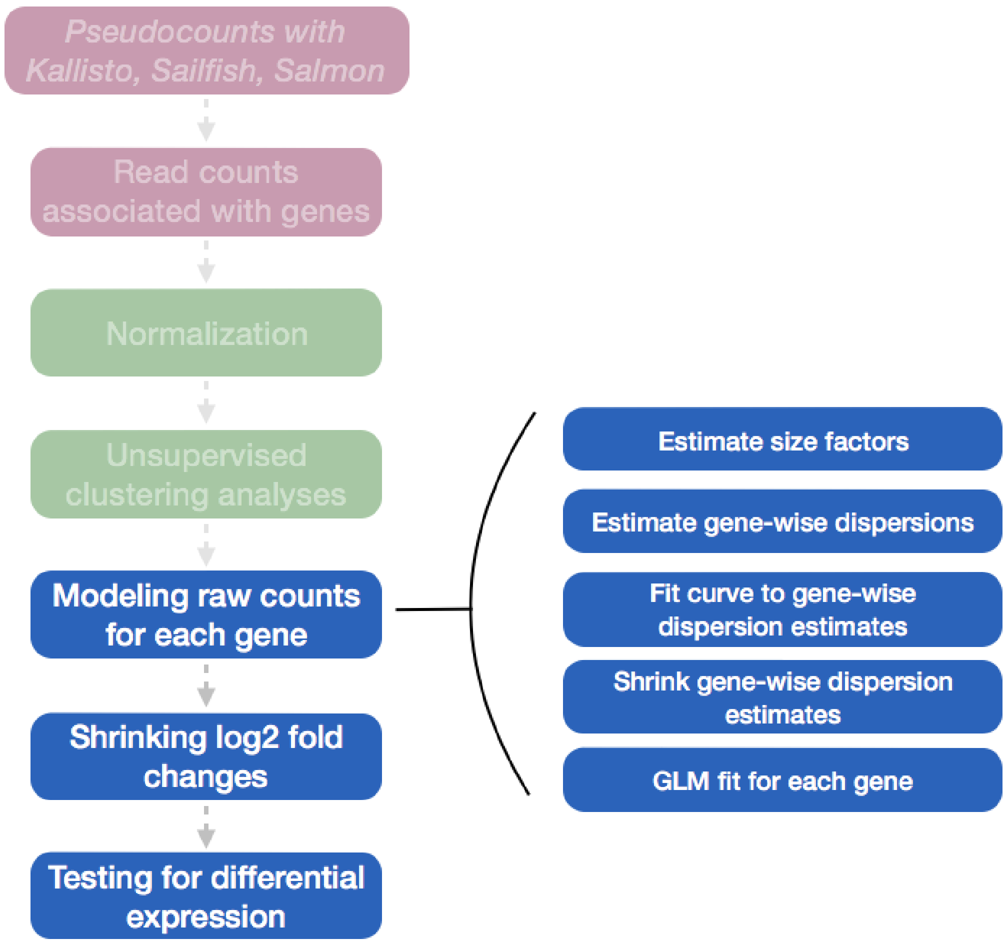


Image credit: Paul Pavlidis, UBC

The DESeq2 paper was published in 2014, but the package is continually updated and available for use in R through Bioconductor. It builds on good ideas for dispersion estimation and use of Generalized Linear Models from the DSS and edgeR methods.

Differential expression analysis with DESeq2 involves multiple steps as displayed in the flowchart below in blue. Briefly, DESeq2 will model the raw counts, using normalization factors (size factors) to account for differences in library depth. Then, it will estimate the gene-wise dispersions and shrink these estimates to generate more accurate estimates of dispersion to model the counts. Finally, DESeq2 will fit the negative binomial model and perform hypothesis testing using the Wald test or Likelihood Ratio Test.


Pseudocounts with Kallisto, Sailfish, Salmon
Read counts associated with genes
Normalization
Unsupervised clustering analyses
Modeling raw counts for each gene
Shrinking log2 fold changes
Testing for differential expression
Estimate size factors
Estimate gene-wise dispersions
Fit curve to gene-wise dispersion estimates
Shrink gene-wise dispersion estimates
GLM fit for each gene

> **NOTE:** DESeq2 is actively maintained by the developers and continuously being updated. As such, it is important that you note the version you are working with. Recently, there have been some rather **big changes implemented** that impact the output. To find out more detail about the specific **modifications made to methods described in the original 2014 paper**, take a look at this section in the DESeq2 vignette.
>
> Additional details on the statistical concepts underlying DESeq2 are elucidated nicely in Rafael Irizarry's materials for the EdX course, "Data Analysis for the Life Sciences Series".

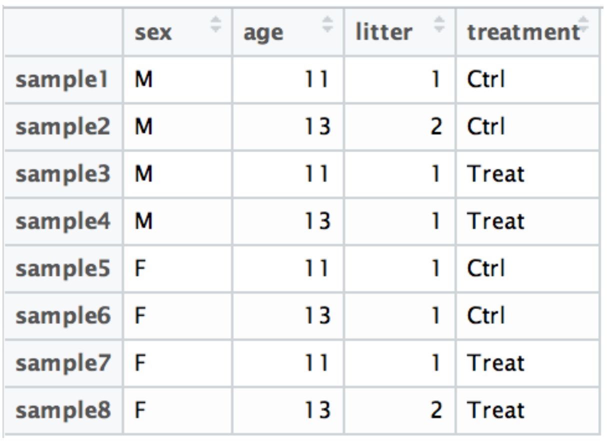
Running DESeq2

Prior to performing the differential expression analysis, it is a good idea to know what **sources of variation** are present in your data, either by exploration during the QC and/or prior knowledge. Once you know the major sources of variation, you can remove them prior to analysis or control for them in the statistical model by including them in your **design formula**.

Design formula

A design formula tells the statistical software the known sources of variation to control for, as well as, the factor of interest to test for during differential expression testing. For example, if you know that sex is a significant source of variation in your data, then `sex` should be included in your model. **The design formula should have all of the factors in your metadata that account for major sources of variation in your data. The last factor entered in the formula should be the condition of interest.**

For example, suppose you have the following metadata:



	sex	age	litter	treatment
sample1	M	11	1	Ctrl
sample2	M	13	2	Ctrl
sample3	M	11	1	Treat
sample4	M	13	1	Treat
sample5	F	11	1	Ctrl
sample6	F	13	1	Ctrl
sample7	F	11	1	Treat
sample8	F	13	2	Treat

If you want to examine the expression differences between treatments, and you know that major sources of variation include `sex` and `age`, then your design formula would be:

```
design <- ~ sex + age + treatment
```

The tilde (~) should always precede your factors and tells DESeq2 to model the counts using the following formula. Note the **factors included in the design formula need to match the column names in the metadata**.

Exercises

1. Suppose you wanted to study the expression differences between the two age groups in the metadata shown above, and major sources of variation were `sex` and `treatment`, how would the design formula be written?
2. Based on our **Mov10** `metadata` dataframe, which factors could we include in our design formula?
3. What would you do if you wanted to include a factor in your design formula that is not in your metadata?

Complex designs

DESeq2 also allows for the analysis of complex designs. You can explore interactions or 'the difference of differences' by specifying for it in the design formula. For example, if you wanted to explore the effect of sex on the treatment effect, you could specify for it in the design formula as follows:

```
design <- ~ sex + age + treatment + sex:treatment
```

Since the interaction term `sex:treatment` is last in the formula, the results output from DESeq2 will output results for this term.

There are additional recommendations for complex designs in the DESeq2 vignette. In addition, Limma documentation offers additional insight into creating more complex design formulas.

> ***NOTE:*** *Need help figuring out **what information should be present in your metadata?** We have additional materials highlighting bulk RNA-seq planning considerations. Please take a look at these materials before starting an experiment to help with proper experimental design.*

MOV10 DE analysis

Now that we know how to specify the model to DESeq2, we can run the differential expression pipeline on the **raw counts**.

To get our differential expression results from our raw count data, we only need to run 2 lines of code!

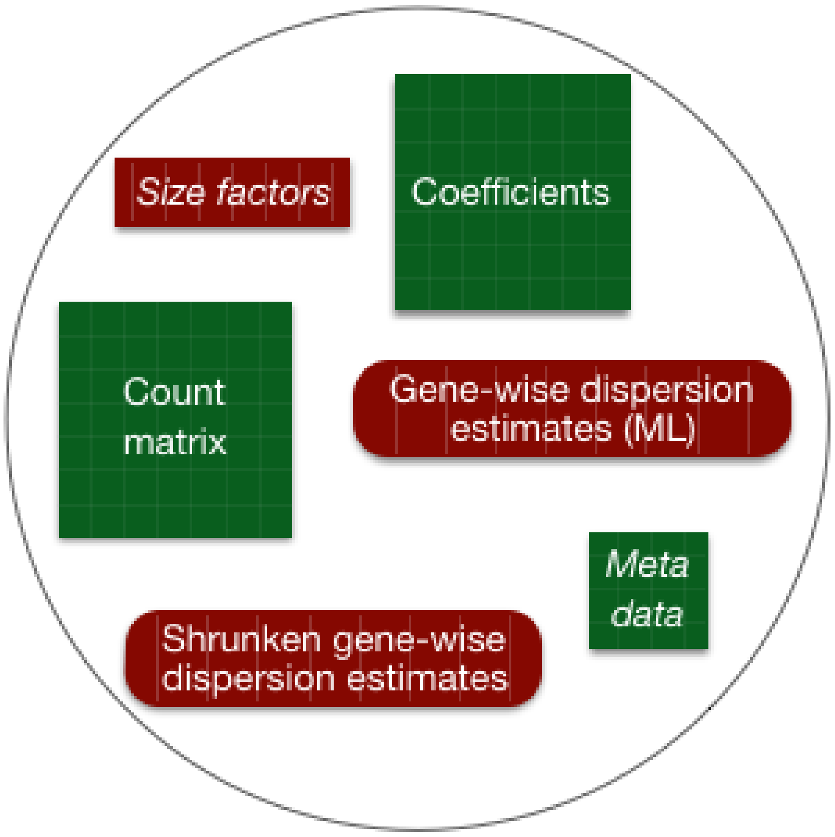
First we create a DESeqDataSet as we did in the 'Count normalization' lesson and specify the `txi` object which contains our raw counts, the metadata variable, and provide our design formula:

```
## Create DESeq2Dataset object
dds <- DESeqDataSetFromTximport(txi, colData = meta, design = ~ sampletype)
```

Then, to run the actual differential expression analysis, we use a single call to the function `DESeq()`.

```
## Run analysis
dds <- DESeq(dds)
```

By re-assigning the results of the function back to the same variable name (`dds`), we can fill in the `slots` of our `DESeqDataSet` object.


Size factors
Coefficients
Count matrix
Gene-wise dispersion estimates (ML)
Meta data
Shrunken gene-wise dispersion estimates

Everything from normalization to linear modeling was carried out by the use of a single function! This function will print out a message for the various steps it performs:

```
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
```

We will discuss what is occurring in each of these steps in the next few lessons, but the code to execute these steps is encompassed in the two lines above.

> **NOTE:** There are individual functions available in DESeq2 that would allow us to carry out each step in the workflow in a step-wise manner, rather than a single call. We demonstrated one example when generating size factors to create a normalized matrix. By calling `DESeq()`, the individual functions for each step are run for you.

Exercise

Let’s suppose our experiment has the following metadata:

	genotype	treatment
sample1	WT	ev

	genotype	treatment
sample2	WT	ev
sample3	WT	ev
sample4	WT	ev
sample5	KO_geneA	ev
sample6	KO_geneA	ev
sample7	KO_geneA	ev
sample8	KO_geneA	ev
sample9	WT	treated
sample10	WT	treated
sample11	WT	treated
sample12	WT	treated
sample13	KO_geneA	treated
sample14	KO_geneA	treated
sample15	KO_geneA	treated
sample16	KO_geneA	treated

How would the design formula be structured to perform the following analyses?

1. Test for the effect of `treatment`.
2. Test for the effect of `genotype`, while regressing out the variation due to `treatment`.
3. Test for the effect of `genotype` on the `treatment` effects. ***
