# Using the BioConductor package 'limma'

**Linear Models for Microarray Analysis (limma)**

http://bioconductor.org/packages/release/bioc/html/limma.html

'limma' provides a comprehensive framework for analysing gene expression data from both microarray and RNA-Seq experiments. In this session, we will illustrate the steps involved in setting up an appropriate analysis model and interpreting the results, with an emphasis on potential pitfalls. Commands that you can enter in your R session are highlighted *with this colour and font.*

To install this package, start R and enter:

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

## try http:// if https:// URLs are not supported

If you already have limma installed, load the package into your R session via:

```
library(limma)
```

The usersguide.pdf can be downloaded from the same link (current version dated 18 September 2017) for further details and examples of using limma - in particular Chapters 8, 9, 13, 15 and relevant case-studies.

All files needed for this session can be found at:

http://www.well.ox.ac.uk/bioinformatics/training/RNASeq_Nov2018

**Two group comparison (Section 9.2 Users guide)**

The simplest possible single channel experiment is to compare two groups. Suppose that we wish to compare two wild type (Wt) mice with three mutant (Mu) mice:

| FileName | Target |
|---|---|
| File1 | WT |
| File2 | WT |
| File3 | Mu |
| File4 | Mu |
| File5 | Mu |

Some simulated expression data for these 5 samples is in the file "Sample_WT_Mu_expression_data.txt", which can be downloaded from the course link (located in the subfolder *Files_and_scripts*) into a suitable working directory on your computer. Make sure to set this to the working directory in your R session (use setwd or the menu bar to change the current working directory). There is data for 100 genes; the first 10 are upregulated in Mu, the next 10 are down-regulated in Mu and the remainder are not changing. The data can be read into R with:

```
eset <- read.table("Sample_WT_Mu_expression_data.txt", sep="\t", header=T, row.names=1)
```

Next, we need to create a design matrix to describe the experimental samples and conditions. Each sample has a row in the design matrix, and the columns indicate which samples belong to each experimental group. When the explanatory variable is categorical, we are actually using a linear model approach to perform ANOVA; dummy variables are created to represent the group membership (1 if sample belongs to that group, 0 otherwise). It is ***really important*** to ensure that order of samples in the design matrix matches the order of samples in the expression data matrix (otherwise your analysis will be on mixed up groups and meaningless!)

When we are comparing two groups, there are different ways we can form the design matrix. We can either:

1. Create a design matrix which includes a coefficient for the mutant vs wild type difference, or

2. Create a design matrix which includes separate coefficients for wild type and mutant mice and then extract the difference as a contrast.

These represent two different ways of *parameterising* the model and the choice determines what the parameters estimated by the model represent. This affects how we subsequently use limma commands to find the comparison of interest (difference in mean expression between groups) and extract the results. They are known as the 'treatment-contrasts' (1) and 'group-means' (2) parameterizations for reasons that will soon become apparent.

**Group-means parameterization**

We'll start with the second approach, because it's slightly more intuitive. The design matrix could be created as follows:

```
design1 <- cbind(WT=c(1,1,0,0,0),MU=c(0,0,1,1,1))
```

## inspect the object created

```
design1
```

Alternatively, it could also be defined using a factor describing the samples, and the function model.matrix:

```
group <- factor(c("WT", "WT", "Mu", "Mu", "Mu"))
```

```
design2 <- model.matrix(~0+group)
```

Inspect the design matrix named design2 and compare to design1 - what is different?

```
design2
```

R orders the levels of a factor alphabetically by default, so groupMu is in the first column and groupWT in the second column; when we created it manually for design1 we did the reverse. Although still equivalent, we'll create the two design matrices the same way for consistency:

```
group <- factor(c("WT", "WT", "Mu", "Mu", "Mu"), levels=c("WT", "Mu"))
```

*design2 <- model.matrix(~0+group)*

*design2*

We'll also rename the columns to match design1 (having checked they are correct)

*colnames(design2) <- c("WT", "MU")*

With this design matrix, a linear model with 2 coefficients is defined and the coefficients estimate the mean expression in the WT and Mu groups (hence the approach is known as the group-means parameterization). We are interested in the difference in expression between the two groups, and this can be extracted as a contrast. A few lines of R code will fit the linear model to each gene and compute the statistics for the contrast of interest:

*fit <- lmFit(eset, design1)*

*cont.matrix <- makeContrasts(MUvsWT=MU-WT, levels=design1)*

*cont.matrix*

*fit2 <- contrasts.fit(fit, cont.matrix)*

*fit2 <- eBayes(fit2)*

*topTable(fit2, adjust="BH")*

Inspect the output – what conclusions do you draw?

Let's see what this code is doing by looking at each line in turn. lmFit is the step of fitting the linear model, and gives the estimates for the model parameters (coefficients). Then a contrast matrix is defined using the 'makeContrasts' function – the groups are referred to in the same way they are defined in the column names of the design matrix (and note the use of 'MU' in the design matrix and 'Mu' in the sample factor) – the following gives an error:

*cont.matrix <- makeContrasts(MuvsWT=Mu-WT, levels=design1)*

Error in eval(ej, envir = levelsenv) : object 'Mu' not found

The contrast of the model coefficients is performed with contrasts.fit, which finds the difference between the specified coefficients. Note also that the way the contrast is defined is important: MU-WT is different to WT-MU and will determine how the resulting output is interpreted – for example, if expression is higher in MU samples compared to WT, the MU-WT contrast will give a positive value, and indicate higher expression in the first group (MU) relative to the second group (WT) i.e. a gene that is up-regulated in mutant compared to wildtype. The reverse will be true for WT-MU, resulting in fold changes of opposite sign (though the magnitude and all the statistics will be the same). Care must be taken when defining the contrast and communicating how the comparison has been performed; it is usually most intuitive to use a WT or control as the second or 'reference' group. Sometimes you will need to over-ride R's default behaviour handling factors (ordering the levels alphabetically) to achieve this.

The eBayes function moderates the empirical variance estimates to improve robustness and reduce the chance of identifying false positives - particularly likely with small sample sizes - and the moderated t-statistics and associated p-values are then computed. Finally the top differentially expressed genes can be inspected using the 'topTable' function:

*topTable(fit2, adjust="BH")*

Later, we'll look in more detail at the results output and consider the issue of multiple testing (the reason that we need to adjust the p-values).

**Treatment-contrasts Parameterization**

The alternative approach, or treatment-contrasts parametrization, directly estimates the difference between mutant and wildtype in the model itself, avoiding the need to use contrasts. The design matrix should be as follows:

| | WT | MUvsWT |
|---|---|---|
| Array1 | 1 | 0 |
| Array2 | 1 | 0 |
| Array3 | 1 | 1 |
| Array4 | 1 | 1 |
| Array5 | 1 | 1 |

Here, the first coefficient estimates the mean log-expression for wildtype mice (same as the previous model), but here plays the role of an intercept and all other coefficients in the model are relative to this baseline. In this case, the second coefficient therefore estimates the *difference* between mutant and wildtype, so the comparison of interest is defined automatically by this model.

Let's try setting up this design matrix using the 'group' factor we defined earlier:

*design3 <- model.matrix(~group)*

*design3*

Because we reversed the default ordering of the group factor levels, the second coefficient is named groupMu and has a 1 for Mu samples – this means the coefficient will estimate Mu vs WT, which is desirable so the fold-changes can be interpreted as the change relative to the baseline level in WT mice.

*colnames(design3) <- c("WT", "MUvsWT")*

Note that the columns have been named according to what each coefficient represents – however we have chosen and assigned these names ourselves and so need to be sure that we are naming them correctly. This is why it's so important to know what the coefficients defined by a given design matrix represent (see below – Understanding what the model coefficients represent). If we wrongly interpret the coefficients, it is unlikely that the intended comparison will be performed correctly.

Differentially expressed genes can now be found by:

*fit <- lmFit(eset, design3)*

*fit <- eBayes(fit)*

*topTable(fit, coef="MUvsWT", adjust="BH")*

which makes the analysis code a mere 3 lines and is done almost instantaneously! (You can check that the output is identical to the first approach).

The coefficient of interest can also be referred to by number and will give the same output:

```
topTable(fit, coef=2, adjust="BH")
```

See what happens if we specify the first coefficient in topTable instead:

```
topTable(fit, coef=1, adjust="BH")
```

Can you work out what we are seeing in the results? (Recall what the first column of the design matrix defined). Answer given at end of document.

As a final check that our code is doing what we think it is doing, we can also calculate the logFCs manually for the top-ranking genes:

```
eset <- as.matrix(eset)
```

e.g. the top result, Gene_3, has a logFC of 3.127986 in the limma output and we can reproduce this from our original expression data matrix:

```
mean(eset[3, 3:5])-mean(eset[3, 1:2]) # 3.127986
```

**Summary**

Defining the comparison of interest directly is preferred for simple designs but for more complex designs, the group-means parameterization is normally the best option; setting up the design so that each coefficient estimates the mean expression of an experimental group is the most intuitive way to know what the coefficients are estimating and to construct the contrast matrix accurately to perform the comparisons of interest. Section 9.5.1 is particularly useful in illustrating how contrasts defining the comparisons of interest correspond to the biological interpretation of the resulting gene lists – defining specific questions of interest in advance (rather than comparing everything possible) is important and helps design and analyse experiments appropriately. These sections (and later in chapter 9) outline how limma can accommodate even very complex experiments. The original limma framework has now been extended to work with count data from RNA-Seq experiments (chapter 15) and so it remains a powerful and versatile toolset for gene expression analysis.

**Understanding what the model coefficients represent**

To convince ourselves what each coefficient is estimating in the two examples above, consider the following:

The design matrix multiplied by the coefficient matrix defines our linear model: in approach 1, the group-means parametrization, a WT sample is defined by:

$(1 \times \beta_1) + (0 \times \beta_2) = \beta_1$

While a Mu sample is defined by:

$(0 \times \beta_1) + (1 \times \beta_2) = \beta_2$

Thus the comparison Mu vs WT must be extracted as the difference between the two coefficients, namely $\beta_2 - \beta_1$

In the second approach, a WT sample is defined as:

$(1 \times \beta_1) + (0 \times \beta_2) = \beta_1$

The first coefficient still estimates the mean expression of WT samples (just as in the first approach). But because of the column of 1s in the first column of the design matrix, a Mu sample is now described as:

$(1 \times \beta_1) + (1 \times \beta_2) = \beta_1 + \beta_2$

Thus Mu vs WT is equivalent to $(\beta_1 + \beta_2) - \beta_1 = \beta_2$ i.e. the second coefficient in the model. We can think of it as follows: expression in the Mu group is calculated as expression in WT plus some value (determined by the estimate of $\beta_2$). If expression is higher in Mu than WT, this value will be positive, and negative if expression is lower in Mu than WT. If they are similarly expressed, $\beta_2$ will be close to zero and this is actually what is being tested – whether the coefficient (or contrast) representing the difference between the two groups is significantly different from zero.

We can use this approach of matrix multiplication to calculate what the coefficients represent and to work out how the comparisons of interest are represented to be sure we have set up our model correctly.

**Understanding limma output**

The topTable function is used to extract the results of performing differential expression analysis with limma. The output object is a data frame usually containing the following columns (see help(topTable) for details:

1. Probe or gene ID

2. logFC – the fold change on log2 scale. Positive values indicate higher expression in the first group relative to the second group (upregulated genes), and vice versa for negative values (downregulated genes). A logFC=1 indicates a 2-fold upregulation, logFC=2 is a 4-fold upregulation, logFC=(-3) is an 8-fold down-regulation etc.

3. AveExpr – this column shows the average normalised expression level across all samples in the design matrix. Note that this is only useful for getting a rough idea of how highly expressed a gene is, particularly for the top-ranking genes from differential expression analysis, where the average value across all samples reflects neither group. Still, values down in the range 4-6 are quite low expressed genes (for microarray data), and 8-10 or higher are high expressed. Low expressed genes can be unreliable and lead to false positives so if many top-ranked genes have low average expression, this may be a warning sign of little real signal in the data.

4. t-statistic – the t-statistic calculated for each gene (moderated by the empirical Bayes method)

5. P.Value – the raw p-value, or the significance of each individual gene

6. adj.P.Val – the adjusted p-value, where the raw p-values have been corrected for multiple testing

**Multiple Testing**

Classical hypothesis testing results in a p-value indicating the probability that a particular result occurred by chance under the null hypothesis (in this case, the null hypothesis is that there is no difference in mean expression levels between WT and Mu mice). When the p-value is sufficiently low (<0.05), the null hypothesis is considered unlikely and rejected.

A p-value of 0.05 means a 5% chance of a false positive result (by chance, our small sample shows a difference, even when none really exists). And every time we conduct a hypothesis test, this 5% 'risk' of seeing a significant result by chance exists. Adjusting raw p-values for multiple testing is a way to avoid inflating the type I error rate (false positives) when testing thousands of genes simultaneously.

False discovery rate (FDR) – also known as Benjamini-Hochberg's (BH) method to control the false discovery rate is a common method used in gene expression analysis. Here, the raw p-values are adjusted in a step-wise fashion:

Raw p-value * number of tests/ rank position

Where rank position refers to the raw p-values sorted in ascending order. The smallest raw p-value is penalised most heavily (p*N/1), the second smallest p-value is adjusted to (p*N/2), the third smallest to (p*N/3) and so on. It is widely used in gene expression analysis, since genes are unlikely to be independent of each other (many are correlated), and the Bonferroni correction (setting alpha =0.05/number of tests) assumes independent tests are performed and is too stringent.

In our final inspection of the limma output, we hope to find some genes passing the FDR correction – say there were 300 significant genes with adjusted p-values <0.05. We interpret this as controlling the false discovery rate at 5%: among the set of 300 genes, only 5% of them (15 genes) are expected to be false positives.

It might be there are 1000s of significant genes, or none at all, depending on the power of the experiment (sufficient replicates) and the size of the effect (fold change) and heterogeneity of the samples (human clinical samples will be more variable than mouse models or cell line experiments). Even if no genes pass the multiple testing correction, it is rare for the results to indicate that there are really no changes of interest in the dataset - it is more likely to be an under-powered experiment and therefore the ranked list of genes are usually worth exploring by looking at whether genes of interest come up or pathway analysis for further insight and informing future follow-up work.

**Exercise – multiple groups**

Think about how you would set up a design matrix for an experiment of 4 groups (say control, treatment1, treatment2, treatment3). Create a factor to represent the samples and a suitable design matrix. What comparisons might be of interest and how would you specify these in limma?

Once you've noted down some ideas, read through section 9.3 of limma usersguide on 'Several Groups'. Continue reading section 9.4 and 9.5 (stop at the end of Section 9.5.2, describing the first (simplest) approach for the 2x2 factorial design – the other 2 approaches may be of interest as an exercise but for practical work, the first approach is least likely to lead to errors in setting up the model and contrasts).

**Getting help**

More information about the limma functions for each step can be found by typing

help('functionName')

or

?('functionName')

Note functions that are part of the 'limma' package are only recognised by R after the package has been loaded via:

library(limma)

If limma has not been loaded and we try to find help for a function defined in that package, an error message will appear:

help(eBayes)

No documentation for 'eBayes' in specified packages and libraries:

you could try '??eBayes'

**Answer to coefficient question**

The first model coefficient was estimating the mean expression of WT samples and if we extract this with topTable, the results are the genes ranked according to expression level in WT (from high to low). We are testing whether the coefficient estimate is significantly different from zero, and in this case the higher the expression level in WT, the more different from zero it will be. All the results will be highly significant, though obviously not very meaningful.