Chapter 15

RNA-Seq Data

15.1 Introduction

The limma approach to RNA-seq explained in the articles by Law et al [15] and Liu et al [20]. See also the article by Law et al [14], which gives a complete workflow case study. In the limma approach to RNA-seq, read counts are converted to log2-counts-per-million (logCPM) and the mean-variance relationship is modelled either with precision weights or with an empirical Bayes prior trend. The precision weights approach is called "voom" and the prior trend approach is called "limma-trend" [15]. In either case, the RNA-seq data can be analyzed as if it was microarray data. This means that any of the linear modelling or gene set testing methods in the limma package can be applied to RNA-seq data.

15.2 Making a count matrix

RNA-seq data usually arrives in the form of FastQ or BAM files of unaligned reads. The reads need to be mapped to a reference genome or transcriptome, then summarized at the exon or gene level to produce a matrix of counts. We find the Rsubread package [17] to be convenient, fast and effective for this purpose. Other popular methods include RSEM [16] and HTseq. A runnable example with complete code showing how to use Rsubread with limma is provided at `http://bioinf.wehi.edu.au/RNAseqCaseStudy`. Another complete code example is provided by Chen et al [6].

15.3 Normalization and filtering

Once a matrix of read counts `counts` has been created, with rows for genes and columns for samples, it is convenient to create a `DGEList` object using the edgeR package:

```
> dge <- DGEList(counts=counts)
```

The next step is to remove rows that consistently have zero or very low counts. One can for example use

```
> keep <- filterByExpr(dge, design)
> dge <- dge[keep,,keep.lib.sizes=FALSE]
```

where `filterByExpr` is a function in the edgeR package. Here we will assume that filtering has been done.

It is usual to apply scale normalization to RNA-seq read counts, and the TMM normalization method [33] in particular has been found to perform well in comparative studies. This can be applied to the `DGEList` object:

```
> dge <- calcNormFactors(dge)
```

15.4 Differential expression: limma-trend

If the sequencing depth is reasonably consistent across the RNA samples, then the simplest and most robust approach to differential exis to use limma-trend. This approach will usually work well if the ratio of the largest library size to the smallest is not more than about 3-fold.

In the limma-trend approach, the counts are converted to logCPM values using edgeR's `cpm` function:

```
> logCPM <- cpm(dge, log=TRUE, prior.count=3)
```

The prior count is used here to damp down the variances of logarithms of low counts.

The logCPM values can then be used in any standard limma pipeline, using the `trend=TRUE` argument when running `eBayes` or `treat`. For example:

```
> fit <- lmFit(logCPM, design)
> fit <- eBayes(fit, trend=TRUE)
> topTable(fit, coef=ncol(design))
```

Or, to give more weight to fold-changes in the gene ranking, one might use:

```
> fit <- lmFit(logCPM, design)
> fit <- treat(fit, lfc=log2(1.2), trend=TRUE)
> topTreat(fit, coef=ncol(design))
```

15.5 Differential expression: voom

When the library sizes are quite variable between samples, then the voom approach is theoretically more powerful than limma-trend. In this approach, the voom transformation is applied to the normalized and filtered `DGEList` object:

```
v <- voom(dge, design, plot=TRUE)
```

The voom transformation uses the experiment design matrix, and produces an `EList` object.

It is also possible to give a matrix of counts directly to voom without TMM normalization, by

```
> v <- voom(counts, design, plot=TRUE)
```

If the data are very noisy, one can apply the same between-array normalization methods as would be used for microarrays, for example:

```
> v <- voom(counts, design, plot=TRUE, normalize="quantile")
```

After this, the usual limma pipelines for differential expression can be applied, for example:

```
> fit <- lmFit(v, design)
> fit <- eBayes(fit)
> topTable(fit, coef=ncol(design))
```

Or, to give more weight to fold-changes in the ranking, one could use say:

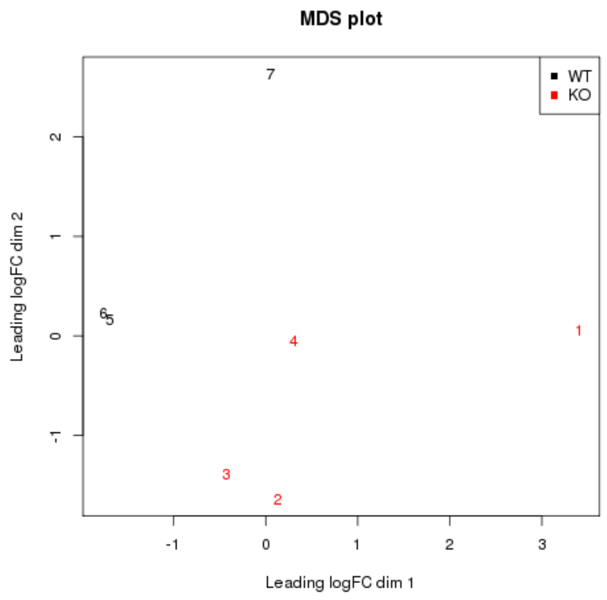
```
> fit <- treat(fit, lfc=log2(1.2))
> topTreat(fit, coef=ncol(design))
```

15.6 Voom with sample quality weights

When a multi-dimensional scaling plot from a designed RNA-seq experiment indicates the presence of outlier samples, it is possible to combine the observation-level weighting strategy used in voom with sample-specific quality weights (as described in the section above on Array Quality Weights) to down-weight outlier samples. This capability is implemented in the `voomWithQualityWeights` function.

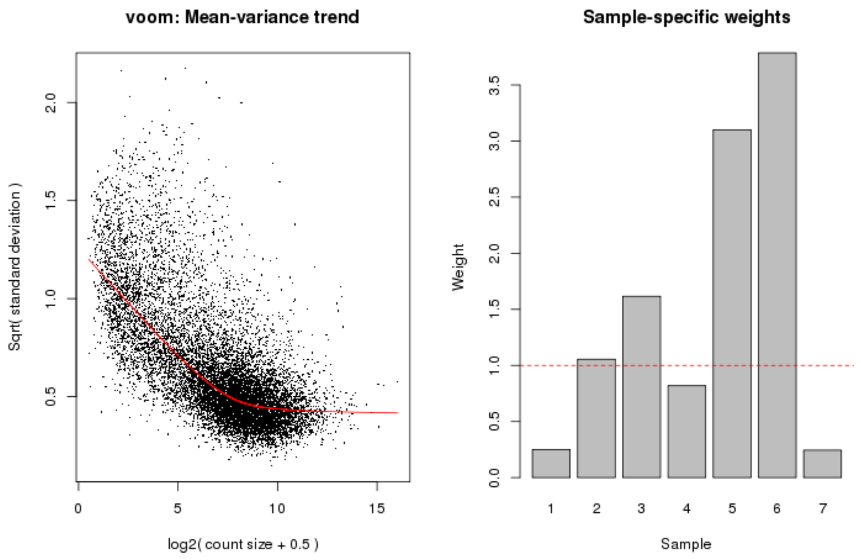
The example below shows its use on an RNA-seq data set where the epigenetic regulator *Smchd1* has been knocked-out in lymphona cell-lines (GEO series GSE64099) [20]. Overall we obtain more differential expression by applying this combined weighting strategy and the raw *p*-value and false discovery rate for the *Smchd1* gene, which has been knocked out, is smaller.

```
> plotMDS(x, labels=1:7, col=as.numeric(genotype), main="MDS plot")
> legend("topright", legend=c("WT", "KO"), col=1:2, pch=15)
```



```
> # Analysis with voom only
>  des[1:7,]
  (Intercept) Smchd1nullvsWt
1           1              1
2           1              1
3           1              1
4           1              1
5           1              0
6           1              0
7           1              0
> v <- voom(x, design=des)
> plotMDS(v, labels=1:7, col=as.numeric(genotype))
> vfit <- lmFit(v)
> vfit <- eBayes(vfit)
> options(digits=3)
> topTable(vfit,coef=2,sort.by="P")
       GeneID    Symbols logFC AveExpr      t  P.Value adj.P.Val    B
74355   74355     Smchd1 -3.12   6.067 -23.35 2.16e-08  0.000266 9.97
```

```
18028   18028       Nfib  8.98   1.714  12.60 2.17e-06  0.013355 3.15
75605   75605      Kdm5b -3.55   3.618 -11.75 3.62e-06  0.014857 5.06
667435 667435 Igkv17-121 -5.35  -1.435 -10.22 9.95e-06  0.025513 2.57
381126 381126      Garem  6.17   0.113  10.08 1.10e-05  0.025513 2.35
381413 381413     Gpr176 -4.02   1.328  -9.90 1.25e-05  0.025513 3.39
75033   75033       Mei4  6.44   0.259   9.69 1.45e-05  0.025513 2.23
69136   69136      Tusc1  5.67  -0.184   8.90 2.67e-05  0.040995 1.87
233552 233552      Gdpd5 -2.82   1.948  -8.56 3.49e-05  0.042754 2.81
80890   80890      Trim2 -1.43   4.491  -8.40 4.00e-05  0.042754 2.72
> top <- topTable(vfit,coef=2,number=Inf,sort.by="P")
> sum(top$adj.P.Val<0.05)
[1] 12
> # Analysis with combined voom and sample quality weights
> vwts <- voomWithQualityWeights(x, design=des, normalization="none", plot=TRUE)
> vfit2 <- lmFit(vwts)
> vfit2 <- eBayes(vfit2)
> topTable(vfit2,coef=2,sort.by="P")
       GeneID Symbols logFC AveExpr     t  P.Value adj.P.Val     B
74355   74355  Smchd1 -3.17   6.067 -28.5 1.61e-09  1.98e-05 12.57
18028   18028    Nfib  9.23   1.714  19.0 4.44e-08  2.73e-04  6.91
381126 381126   Garem  6.45   0.113  15.9 1.85e-07  7.58e-04  6.02
75033   75033    Mei4  6.56   0.259  15.0 2.84e-07  8.73e-04  5.83
69136   69136   Tusc1  5.88  -0.184  13.6 6.16e-07  1.11e-03  5.31
54354   54354  Rassf5  5.74   4.554  13.6 6.26e-07  1.11e-03  6.63
75605   75605   Kdm5b -3.80   3.618 -13.5 6.53e-07  1.11e-03  6.67
58998   58998   Pvrl3  7.69   0.961  13.1 8.46e-07  1.11e-03  5.33
320398 320398   Lrig3  7.39   1.584  13.1 8.49e-07  1.11e-03  5.32
17069   17069    Ly6e  2.63   7.605  13.0 9.01e-07  1.11e-03  6.26
> top2 <- topTable(vfit2,coef=2,number=Inf,sort.by="P")
> sum(top2$adj.P.Val<0.05)
[1] 1478
```



15.7 Differential splicing

limma can also detect genes that how evidence of differential splicing between conditions. One can test for differential splicing associated with any contrast for a linear model.

In this case, the matrix of counts should be at the exon level, with a row for each exon. For example,

```
> dge <- DGEList(counts=counts)
> dge$genes$GeneID <- GeneID
```

where `counts` is a matrix of exon-level counts, and GeneID identifies which gene each exon belongs to. Then filter and normalize:

```
> A <- rowSums(dge$counts)
> dge <- dge[A>10,, keep.lib.sizes=FALSE]
> dge <- calcNormFactors(dge)
```

Then apply the voom transformation and fit a linear model:

```
> v <- voom(dge, design, plot=TRUE)
> fit <- lmFit(v, design)
```

Now we can test for differential splicing associated with any coefficient in the linear model. First run the diffSplice function:

```
> ex <- diffSplice(fit, geneid="GeneID")
```

Then

```
> topSplice(ex, coef=2, test="simes")
```

will find genes that show evidence of differential splicing associated with the second coefficient in the linear model. The output is similar that from the limma topTable function. More detail can be obtained by

```
> topSplice(ex, coef=2, test="t")
```

which will show individual exons that are enriched or depleted relative to other exons in the same gene. To display the pattern of exons in the top genes:

```
> plotSplice(ex)
```