DESeq2 R Package Supplementary Instructions Manual

Ana Margarida Gonçalves

89338, amsg99@ua.pt

Introduction

The vignette *DESeq2::DESeq2* in the R documentation, visible in [1], explains the use of the *DESeq2* package and demonstrates typical workflows, with an R example. This file was written by the creators of the package *DESeq*, reported in [2]. As it is well detailed, this supplementary instructions manual will only have small details that might be helpful to understand better the package. Note that some excerpts of the vignette are presented in this manual to allow a better understanding.

Additionally, the titles of this document are equivalent to the ones in [1] in order to help to locate in the vignette.

Two examples can be seen in: Github MargaridaGoncalves / *Deseq2_RNA_seq*

1 Which data might be used in this package

A basic task in the analysis of count data from RNA sequencing (RNA-seq) is the detection of differentially expressed genes. The count data are presented as a table which reports, for each sample, the number of sequence fragments that have been assigned to each gene, as shown in section 3.1. Analogous data also arise for other assay types, including comparative ChIP-Seq (Chromatin immunoprecipitation followed by sequencing, a method used to analyze protein interactions with DNA), Hi-C, shRNA (short hairpin RNA) screening, and mass spectrometry.

Taking as an example, differential expression analysis is used to identify differences in the transcriptome across a cohort of samples.It is often used to define the differences between multiple biological conditions (e.g. drug treated vs untreated samples) [3].

The package DESeq2 provides methods to test for differential expression by use of negative binomial generalized linear models; the estimates of dispersion and logarithmic fold changes incorporate data-driven prior distributions [1].

2 DESeq2 Package Information

RNA sequencing (RNA-Seq) data must be normalized by the sequencing depth before any comparison of the counts between experiments can be made.

In DESeq, reported in [2], the sequencing depth is estimated by the count of the gene with the median count ratio across all genes.

Differential expression analysis is used to identify differences in the transcriptome (gene expression) across a cohort of samples. Often, it will be used to define the differences between multiple biological conditions (e.g. drug treated vs. untreated samples).

DESeq2 doesn't actually use normalized counts, rather it uses the raw counts and models the normalization inside the Generalized Linear Model (GLM). The identification of differentially expressed genes from RNA-Seq data through the *Deseq* function. It uses a negative binomial distribution and uses local regression to estimate the relationship between the variance and the mean.

Choice of distribution While for large counts, normal distributions might provide a good approximation of between-replicate variability, this is not the case for lower count values, whose discreteness and skewness mean that probability estimates computed from a normal approximation would be inadequate.

For small numbers of replicates as often encountered in practice, it is not possible to obtain simultaneously reliable estimates of the variance and mean parameters of the NB distribution. EdgeR addresses this problem by estimating a single common dispersion parameter. In the DESeq method, the researchers made use of the possibility to estimate a more flexible, mean-dependent local regression. The amount of data available in typical experiments is large enough to allow for sufficiently precise local estimation of the dispersion.

In [4] is reported that if fewer than 12 replicates are used, a superior combination of true positive and false positive performances makes edgeR and DESeq2 the leading tools. Furthermore, for higher replicate numbers, minimizing false positives is more important and the DESeq marginally outperforms the other tools.

3 Initial Data

Here it is exemplified the types of Matrix that are used in the analysis.

Note 1: As previously stated, a basic task in the analysis of count data from RNA sequencing is the detection of differentially expressed genes. Here, in DESeq2 it is needed the an expression matrix that has the counts and a table of the sample(s) information(s).

Note 2: It is important to notice that the column names of the expression matrix must be equal to the row names of the sample info matrix, in order to use the function *DESeqDataSetFromMatrix*. Additionally, it is absolutely critical that the columns of the count matrix and the rows of the column data (information about samples) are in the same order.

Note 3: The conditions used in the design in the function *DESeqDataSetFromMatrix* must be factors. Specify the reference level in all interest factors.

3.1 Expression Matrix

As input, the DESeq2 package expects count data as obtained, e.g., from RNA-seq or another high-throughput sequencing experiment, in the formof a matrix of integer values. The value in the nth row and the mth column of the matrix tells how many reads can be assigned to gene N in sample M.

Note 4: The values in the matrix should be un-normalized counts or estimated counts of sequencing reads (for single-end RNA-seq) or fragments (forpaired-end RNA-seq). It is important to provide count matrices as input for DESeq2's statistical model hold, as only the count values allow assessing the measurement precision correctly. The DESeq2 model internally corrects for library size, so transformed or normalized values such as counts scaled by library size should not be used as input.

	Sample 1	Sample 2		Sample M
	GSM6160812	GSM6160813		GSM6160833
Gene 1	88	10	...	102
Gene 2	55	33	...	34
...	...	...	...	...
Gene N	32	0	...	22

	Sample 1		Sample 2			Sample M	
	Time 1 CD4T	Time 2 CD4T	Time 1 CD4T	Time 2 CD4T		Time 1 CD4T	Time 2 CD4T
Gene 1	38	10	106	11	...	11	22
Gene 2	55	33	333	77	...	22	87
...	...	...	...	...	...	...	...
Gene N	32	0	100	30	...	22	33

3.2 Sample info Matrix

Here we have the details of the conditions of the experiment.
It can be simple as the study design was made only with consideration of one factor such as:

	Condition 1
	Treatment
GSM6160812	Placebo
GSM6160813	Insulin
...	...
GSM6160833	Placebo

Or it can be more detailed as:

	Condition 1	Condition 2	Condition 3	Condition 4
	Treatment	Time Point	Sex	Batch
GSM6160812	Placebo	Pre-clinical	Female	1
GSM6160813	Insulin	At-diagnosis	Female	2
...	...	...	...	...
GSM6160833	Placebo	Pre-clinical	Male	2

4 Extract results from a DESeq analysis

First it is necessary to create a DESeq object. For that we use the function *DESeqDataSetFromMatrix*. As the name of the function describes, the DESeq dataset is created from a matrix. In the section *Quick start* of [1] it is detailed other structures where the data set can be imported from. There they show 4 ways of constructing a DESeqDataSet, depending on what pipeline was used upstream of DESeq2 to generated counts or estimated counts, but this supplementary document will only detail the coding for a count matrix.

```
dds <- DESeqDataSetFromMatrix(countData = expression.matrix,
                              colData = data_info,
                              design= ~ Batch + Treatment )
```

This code chunk assumes that you have a count matrix called *expression.matrix* and a table of sample information called *data_info*. The *design* indicates how to model the samples, here, that we want to measure the effect of the *treatment*, controlling for *batch* differences. As detailed in **note 2**, the two factor variables *batch* and *treatment* should be columns of data_info.

How to check for Note 2:

```
all(rownames(data_info)==colnames(expression.matrix))
```

Note 5: It is recommended to put the variable of interest at the end of the formula and make sure the control level is the first level.

A DESeqDataSet object must have an associated design formula. The design formula expresses the variables which will be used in modeling. The argument *design* can take form of different models:

1. design(dds) $\leftarrow \sim$ Treatment
2. design(dds) $\leftarrow \sim$ Batch + Time Point + Treatment
3. design(dds) $\leftarrow \sim$ Batch + Time Point + Treatment + Time Point:Treatment

The examples 2 and 3 present a multi-factor design.

5 Multi-factor designs

Experiments with more than one factor influencing the counts can be analyzed using design formulas that include the additional variables.

If there is unwanted variation present in the data (e.g. batch effects) it is always recommend to correct for this, which can be accommodated in DESeq2 by including in the design any known batch variables.

DESeq2 can analyze any possible experimental design that can be expressed with fixed effects terms (multiple factors, designs with interactions, designs with continuous variables, splines, etc).

By adding variables to the design, one can control for additional variation in the counts. For example, if the condition samples are balanced across experimental batches, by including the *batch factor* to the design, one can increase the sensitivity for finding differences due to condition.

6 Pre-filtering

While it is not necessary to pre-filter low count genes before running the DESeq2 functions, there are two reasons which make pre-filtering useful: by removing rows (genes) in which there are very few reads, we reduce the memory size of the dds data object, and we increase the speed of count modeling within DESeq2. It can also improve visualizations, as features with no information for differential expression are not plotted in dispersion plots or MA-plots. If we have counts that range from 0 to 500 there is no motive to analyse a gene that is presented with low counts through all the samples.

In the example presented in the [1], they perform pre-filtering to keep only rows that have a count of at least 10 for a minimal number of samples. The count of 10 is a reasonable choice for bulk RNA sequencing given that at least 5 million reads per sample are expected. Furthermore, at least 12 replicates per condition for experiments where identifying the majority of all differentially expressed (DE) genes is important. (Bulk RNA-Seq experiments provide a view of gene expression of an entire sample. However they do not differentiate among cell types within the sample, rather they give a view of gene expression within a whole organ or tissue type.)

In RNA-seq data analysis we often see that many genes (up to 50%) have little or no expression. It is common to pre-filter (remove) these genes prior to analysis. In general genomics filtering might be beneficial to the analysis.

A recommendation for the minimal number of samples is to specify the smallest group size, e.g. here there are 3 treated samples. If there are not discrete groups, one can use the minimal number of samples where non-zero counts would be considered interesting.One can also omit this step entirely and just rely on the independent filtering procedures available in results().

```
smallestGroupSize <-3
keep <-rowSums(counts(dds)>=10)>=smallestGroupSize
dds <-dds[keep,]
```

7 Differential expression analysis

The standard differential expression analysis steps are wrapped into a single function, DESeq. Results tables are generated using the function results, which extracts a results table with log2 fold changes, p values and adjusted p values. With no additional arguments to results, the log2 fold change and Wald test p value will be for the last variable in the design formula, and if this is a factor, the comparison will be the last level of this variable over the reference level.

```
dds <- DESeq(dds)
resultsNames(dds) # lists the coefficients
```

Then we obtain the DESeq results through the function results():

```
res <- results(dds)
```

```
[1] "Intercept"
[2] "Batch_Batch_1_vs_Batch_2"
[3] "treatment_Placebo_vs_Insulin"
```

Note that we could have specified the coefficient or contrast we want to build a results table for, using either of the following equivalent commands:

```
res <- results(dds, name = "Treatment_Placebo_vs_Insulin")
res <- results(dds, contrast = c("Treatment","Placebo","Insulin"))
```

The results function automatically performs independent filtering based on the mean of normalized counts for each gene, optimizing the number of genes which will have an adjusted p-value below a given FDR cutoff, *alpha*. By default the argument *alpha* is set to 0.1.

7.1 log2 fold change interpretation

For a particular gene, a log2 fold change of -1 for condition 'Placebo vs Insulin' means that the Placebo induces a multiplicative change in observed gene expression level of $2^{-1} = 0.5$ compared to the Insulin condition. If the variable of interest is continuous-valued, then the reported log2 fold change is per unit of change of that variable.

7.2 p-values set to NA

Some values in the results table can be set to NA for one of the following reasons:

1. If within a row, all samples have zero counts, the baseMean column will be zero, and the log2 fold change estimates, p-value and adjusted p-value will all be set to NA;

2. If a row contains a sample with an extreme count outlier then the p-value and adjusted p-value will be set to NA.

7.3 Filtering the low count genes

```
keep = rowSums(counts(dds) >= 10) >= min(table(data_info$X2))
dds<- dds[keep,]
```

8 Log fold change shrinkage for visualization and ranking

Shrinkage of effect size (LFC estimates) is useful for visualization and ranking of genes. To shrink the LFC, we pass the dds object to the function *lfcShrink*.

We provide the dds object and the name or number of the coefficient we want to shrink, where the number refers to the order of the coefficient as it appears in *resultsNames(dds)*.

```
resultsNames(dds)
## [1] "Intercept" "Batch_Batch_1_vs_Batch_2" "Treatment_Placebo_vs_Insulin"
```

Note that *apeglm* is not the only shrinkage estimator available. From DESeq2::lfcShrink we have:

```
  lfcShrink(
 dds,
 coef,
 contrast,
 res,
 type = c("apeglm", "ashr", "normal"),
 lfcThreshold = 0,
 svalue = FALSE,
 returnList = FALSE,
 format = c("DataFrame", "GRanges", "GRangesList"),
 saveCols = NULL,
 apeAdapt = TRUE,
 apeMethod = "nbinomCR",
 parallel = FALSE,
 BPPARAM = bpparam(),
 quiet = FALSE,
 ...
)
```

- *apeglm* is the adaptive Student's t prior shrinkage estimator from the 'apeglm' package [5];

- *ashr* is the adaptive shrinkage estimator from the 'ashr' package, using a fitted mixture of normals prior [6];
- *normal* is the 2014 DESeq2 shrinkage estimator using a Normal prior [7].

The 'apeglm' publication demonstrates that *apeglm* and *ashr* outperform the original *normal* shrinkage estimator.

Note 6: *apeglm* requires the use of *coef.*

Note 7: Only *coef* or contrast can be specified, not both, simultaneously. Relatively to the *coef*, the name or number of the coefficient (LFC) to shrink can be consulted throught the code line "resultsNames(dds)" after running DESeq(dds). Coef = 1 refers to the intercept
Because we are interested in Placebo vs Insulin, we set 'coef=3'

```
resLFC1 <- lfcShrink(dds, coef=3, type="apeglm")
resLFC1
```

Which is equivalent to writing:

```
resLFC <- lfcShrink(dds, coef= "Treatment_Placebo_vs_Insulin", type = "apeglm")
```

9 p-values and adjusted p-values

As previously stated in the *results* function, *alpha* is set to 0.1. To change it is needed to code:

```
res05 <- results(dds, alpha=0.05)
summary(res05)

sum(res05$padj < 0.05, na.rm=TRUE)
```

10 Plot Counts

Here we specify the gene which had the smallest p-value from the results table created above:

```
plotCounts(dds, gene=which.min(res$padj), intgroup="treatment")
```

This can be helpful if we want to see a specific gene of the dataset:

```
plotCounts(dds, gene="Gene 1", intgroup="treatment")
```

11 Count data transformations

In order to test for differential expression, we operate on raw counts and use discrete distributions as described in the previous section on differential expression. However for other downstream analyses – e.g. for visualization or clustering – it might be useful to work with transformed versions of the count data.
One makes use of the concept of variance stabilizing transformations (VST) (Tibshirani 1988; Huber et al. 2003; Anders and Huber2010), and the other is the regularized logarithm (rlog), which incorporates a prior on the sample differences (Love, Huber, and Anders 2014).

12 Recommendations for single-cell analysis

The DESeq2 developers and collaborating groups have published recommendations for the best use of DESeq2 for single-cell datasets, which have been described first in Van den Berge et al. (2018).
Important: Default values for DESeq2 were designed for bulk data and will not be appropriate for single-cell datasets.
In this section the document [1] presents the necessary changes.

References

[1] "Analyzing RNA-seq data with DESeq2, [DESeq2 package version: 1.41.12]." https://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html. Accessed: 12/10/2023.

[2] S. Anders and W. Huber, "Differential expression analysis for sequence count data," Genome Biology, vol. 11, 10 2010.

[3] J. Li, D. M. Witten, I. M. Johnstone, and R. Tibshirani, "Normalization, testing, and false discovery rate estimation for rna-sequencing data," Biostatistics, vol. 13, pp. 523–538, 7 2012.

[4] N. J. Schurch, P. Schofield, M. Gierliński, C. Cole, A. Sherstnev, V. Singh, N. Wrobel, K. Gharbi, G. G. Simpson, T. Owen-Hughes, M. Blaxter, and G. J. Barton, "How many biological replicates are needed in an rna-seq experiment and which differential expression tool should you use?," RNA, vol. 22, pp. 839–851, 6 2016.

[5] A. Zhu, J. G. Ibrahim, and M. I. Love, "Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences," Bioinformatics, vol. 35, pp. 2084–2092, 11 2018.

[6] M. Stephens, "False discovery rates: a new deal," Biostatistics, vol. 18, pp. 275–294, 10 2016.

[7] M. I. Love, W. Huber, and S. Anders, "Moderated estimation of fold change and dispersion for rna-seq data with deseq2," Genome Biology, vol. 15, 12 2014.