

Analyzing RNA-seq data for differential exon usage with the DEXSeq package

Alejandro Reyes, Simon Anders, Wolfgang Huber

June 23, 2011

Contents

| | | |
|----------|--|-----------|
| 1 | The Pasilla dataset | 1 |
| 2 | Normalisation and dispersion estimation | 3 |
| 3 | Testing for differential exon usage | 4 |
| 4 | Batch effects | 5 |
| 5 | Visualization | 6 |
| 6 | Gene count table | 8 |
| 7 | Creating <i>ExonCountSet</i> objects | 9 |
| 7.1 | From files produced by HTSeq | 9 |
| 7.2 | From elementary R data structures | 10 |
| 8 | Session Information | 10 |

Abstract

RNA-seq is a powerful tool for transcriptome analysis. It enables the discovery of novel transcript splice sites and isoforms, and there is interest in the quantitative comparison of exon usage between different conditions. For the analysis of differential expression between conditions, appropriate modeling of the experimental and biological variability is important, and such capabilities are offered, for instance, by the packages *edgeR* [?] and *DESeq* [1]. However, there is currently no software that specifically addresses exon level expression and differential exon usage. In this package, we provide a method to systematically detect differential exon usage using RNA-seq. We use as input the number of reads mapping to each of the exons of a genome. The method is demonstrated on the data from the package *pasilla*.

1 The Pasilla dataset

We will use the `pasillaExons` dataset from the *pasilla* package. `pasillaExons` is an object of class *ExonCountSet*. Brooks et al. [2] investigated the effect of siRNA knock-down of Pasilla, whose protein is known to bind to mRNA in the spliceosome, and which is thought to be involved

in the regulation of splicing, on the transcriptome of fly S2-DRSC cells. Pasilla is the *Drosophila melanogaster* ortholog of mammalian NOVA1 and NOVA2. The dataset, which is provided by NCBI Gene Expression Omnibus (GEO) under the accession number GSE18508¹, contains 3 biological replicates of the knockdown as well as 4 biological replicates for the untreated control.

In the *pasilla* package, we provide data from a subset of genes. A subset was chosen in order to speed up the computations shown in this vignette. We start by loading the *DEXSeq* package and the example data.

```
> library("DEXSeq")
> data("pasillaExons", package = "pasilla")
```

The `pData` accessor function shows the available sample annotations.

```
> pData(pasillaExons)
```

| | sizeFactor | condition | replicate | type |
|--------------|------------|-----------|-----------|-------------|
| treated1fb | NA | treated | 1 | single-read |
| treated2fb | NA | treated | 2 | paired-end |
| treated3fb | NA | treated | 3 | paired-end |
| untreated1fb | NA | untreated | 1 | single-read |
| untreated2fb | NA | untreated | 2 | single-read |
| untreated3fb | NA | untreated | 3 | paired-end |
| untreated4fb | NA | untreated | 4 | paired-end |

We also print the first 6 lines of selected columns of the feature data annotation:

```
> head(fData(pasillaExons)[, c(1, 2, 7:10)])
```

| | geneID | exonID | chr | start | end | strand |
|-----------------|-------------|--------|-------|---------|---------|--------|
| FBgn0000256:001 | FBgn0000256 | E001 | chr2L | 3872658 | 3872947 | - |
| FBgn0000256:002 | FBgn0000256 | E002 | chr2L | 3873019 | 3873322 | - |
| FBgn0000256:003 | FBgn0000256 | E003 | chr2L | 3873385 | 3874395 | - |
| FBgn0000256:004 | FBgn0000256 | E004 | chr2L | 3874450 | 3875302 | - |
| FBgn0000256:005 | FBgn0000256 | E005 | chr2L | 3878895 | 3879067 | - |
| FBgn0000256:006 | FBgn0000256 | E006 | chr2L | 3879652 | 3880038 | - |

There are 46 genes in the dataset, of these, there is one with 36 exons, and for instance, three with 16 exons:

```
> table(table(geneIDs(pasillaExons)))
```

| | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|---|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 14424 | 2 | 3 | 3 | 2 | 4 | 2 | 3 | 3 | 3 | 2 | 3 | 2 |
| 15 | 16 | 17 | 19 | 22 | 23 | 24 | 25 | 36 | | | | |
| 1 | 3 | 2 | 1 | 1 | 3 | 1 | 1 | 1 | | | | |

In Section 7, we explain how you can create analogous data objects from your own data.

¹<http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE18508>

2 Normalisation and dispersion estimation

Different samples might be sequenced with different depths. In order to adjust for such coverage biases, we introduce size factor parameters. *DEXSeq* uses the same method as *DESeq*, which is provided in the function `estimateSizeFactors`.

```
> pasillaExons <- estimateSizeFactors(pasillaExons)

> sizeFactors(pasillaExons)

      treated1fb  treated2fb  treated3fb  untreated1fb  untreated2fb  untreated3fb
      1.084      0.970      0.932      1.092      1.447      0.827
untreated4fb
      0.827
```

To test for differential expression, we need to estimate the data's variance. This is needed to be able to distinguish between normal technical and biological variation (noise) and real effects on gene expression between the different conditions. The information on the size of the noise is drawn from the biological replicates in the dataset. However, as typical for RNA-seq experiments, the number of replicates is too small to estimate variance or dispersion parameters individually gene by gene. Instead, variance information is shared across genes, in an intensity dependent manner. Computationally, this is done through Cox-Reid likelihood estimation (our method follows that of the package *edgeR* [?]), and by fitting a regression of the dispersion values on the mean. These steps are implemented in the function `estimateDispersions`.

```
> pasillaExons <- estimateDispersions(pasillaExons)
```

The result from the Cox-Reid estimation is stored in the column `dispersion_CR_est` of the feature data. Then, the dispersion-mean relation $\alpha = \alpha_0 + \alpha_1/\mu$ is fit to these values (and the coefficients stored in slot `dispFitCoefs`). Finally, for each exon, the maximum of the CR estimate and the fitted value is taken as the exon's final dispersion value and stored in the `dispersion` slot.

```
> head(fData(pasillaExons)$dispersion_CR_est)

[1] 0.0381 0.0130 0.0252 0.0452 0.0524 0.0841

> pasillaExons@dispFitCoefs

      (Intercept) I(1/means[good])
      0.0433      1.4081

> head(fData(pasillaExons)$dispersion)

[1] 0.0539 0.0497 0.0455 0.0465 0.0547 0.6671
```

In Section 4, we will see how to incorporate further experimental or technical variables into the dispersion estimation.

3 Testing for differential exon usage

Having the dispersion estimates and the size factors, we can now test for differential exon usage. For each gene, we fit a generalized linear model with the formula

```
sample + exon + condition * I(exon == exonID)
```

and compare it to the smaller model (the null model)

```
sample + exon + condition.
```

We compare the deviances of both fits using a χ^2 -distribution. To create a data frame that encodes the model for a gene, with columns `sample`, `exon`, `condition`, `sizeFactors` and `count`, the function `modelFrameForGene` is used.

```
> head(modelFrameForGene(pasillaExons, "FBgn0010909"))
```

| | sample | exon | sizeFactor | condition | replicate | type | dispersion | count |
|---|------------|------|------------|-----------|-----------|-------------|------------|-------|
| 1 | treated1fb | E001 | 1.08 | treated | 1 | single-read | 0.2009 | 3423 |
| 2 | treated1fb | E002 | 1.08 | treated | 1 | single-read | 0.0575 | 443 |
| 3 | treated1fb | E003 | 1.08 | treated | 1 | single-read | 0.0455 | 589 |
| 4 | treated1fb | E004 | 1.08 | treated | 1 | single-read | 0.0463 | 671 |
| 5 | treated1fb | E005 | 1.08 | treated | 1 | single-read | 0.0653 | 637 |
| 6 | treated1fb | E006 | 1.08 | treated | 1 | single-read | 0.0450 | 816 |

The actual test (which already includes a call to `modelFrameForGene`) is performed by the function `testGeneForDEU`:

```
> testGeneForDEU(pasillaExons, "FBgn0010909")
```

| | deviance | df | pvalue |
|------|----------|----|----------|
| E001 | 0.0798 | 1 | 7.78e-01 |
| E002 | 0.4040 | 1 | 5.25e-01 |
| E003 | 0.0423 | 1 | 8.37e-01 |
| E004 | 1.0590 | 1 | 3.03e-01 |
| E005 | 2.2407 | 1 | 1.34e-01 |
| E006 | 0.5873 | 1 | 4.43e-01 |
| E007 | 0.2365 | 1 | 6.27e-01 |
| E008 | 0.0356 | 1 | 8.50e-01 |
| E009 | 0.0491 | 1 | 8.25e-01 |
| E010 | 62.9543 | 1 | 2.11e-15 |
| E011 | 1.4860 | 1 | 2.23e-01 |
| E012 | 0.0490 | 1 | 8.25e-01 |
| E013 | 1.4334 | 1 | 2.31e-01 |
| E014 | 0.0833 | 1 | 7.73e-01 |
| E015 | 0.0870 | 1 | 7.68e-01 |
| E016 | 0.1898 | 1 | 6.63e-01 |
| E017 | 0.2882 | 1 | 5.91e-01 |
| E018 | 1.6767 | 1 | 1.95e-01 |
| E019 | 0.0632 | 1 | 8.01e-01 |
| E020 | 0.6447 | 1 | 4.22e-01 |
| E021 | 0.0808 | 1 | 7.76e-01 |
| E022 | 0.4237 | 1 | 5.15e-01 |
| E023 | 3.5248 | 1 | 6.05e-02 |

We see that there is one exon, E010, with a very small p value, while for all other exons, the p values are unremarkable.

A convenient interface which calls `testGeneForDEU` for all genes and fills the `pvalue` and `padjust` columns of the `featureData` slots of the `ExonCountSet` object with the results is provided by the function `testForDEU`.

```
> pasillaExons <- testForDEU(pasillaExons)
```

Testing for differential exon usage

The function `DEUresultTable` provides a summary table of the results.

```
> res1 <- DEUresultTable(pasillaExons)
```

```
> table(res1$padjust < 0.1)
```

```
FALSE TRUE
  453   16
```

4 Batch effects

In the previous section we performed the analysis of differential exon usage ignoring the information regarding the library type of the samples.

```
> design(pasillaExons)
```

| | condition | replicate | type |
|--------------|-----------|-----------|-------------|
| treated1fb | treated | 1 | single-read |
| treated2fb | treated | 2 | paired-end |
| treated3fb | treated | 3 | paired-end |
| untreated1fb | untreated | 1 | single-read |
| untreated2fb | untreated | 2 | single-read |
| untreated3fb | untreated | 3 | paired-end |
| untreated4fb | untreated | 4 | paired-end |

In this section, we show how to take the factor `type` into account in the analysis. First, we need to provide the function `estimateDispersions` with a formula that makes it aware of the additional factor (besides `condition`, which it considers by default).

```
> formuladispersion <- count ~ sample + (exon + type) * condition
```

```
> pasillaExons <- estimateDispersions(pasillaExons, formula = formuladispersion)
```

Dispersion estimation. (Progress report: one dot per 100 genes)

Setting up model frames

Setting up model matrices.

Calculating initial fits.

Performing Cox-Reid dispersion estimation

Fitting mean-dispersion relation

Finished with dispersion estimation.

Second, for the testing, we will also change the two formulas to take into account the library type.

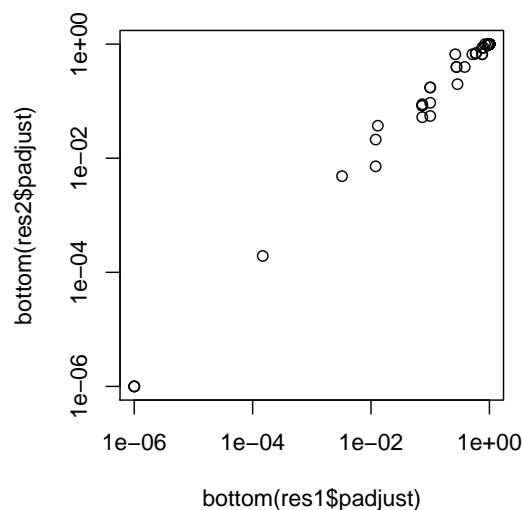


Figure 1: Comparison of differential exon usage p values from analysis with (y -axis, `res2`) and without (x -axis, `res1`) consideration of batch (library type) effects.

```
> formula0 <- count ~ sample + type * exon + condition
> formula1 <- count ~ sample + type * exon + condition * I(exon ==
+   exonID)
```

```
> pasillaExons <- testForDEU(pasillaExons, formula0 = formula0,
+   formula1 = formula1)
```

Testing for differential exon usage

```
> res2 <- DEUresultTable(pasillaExons)
```

```
> table(res2$padjust < 0.1)
```

```
FALSE TRUE
  455    14
```

Fixme: *The results look not very different from those of Section~3:*

```
> bottom = function(x) pmax(x, 1e-06)
> plot(bottom(res1$padjust), bottom(res2$padjust), log = "xy")
```

See Figure~1.

5 Visualization

DEXSeq has a function to visualize the results of `testForDEU`.

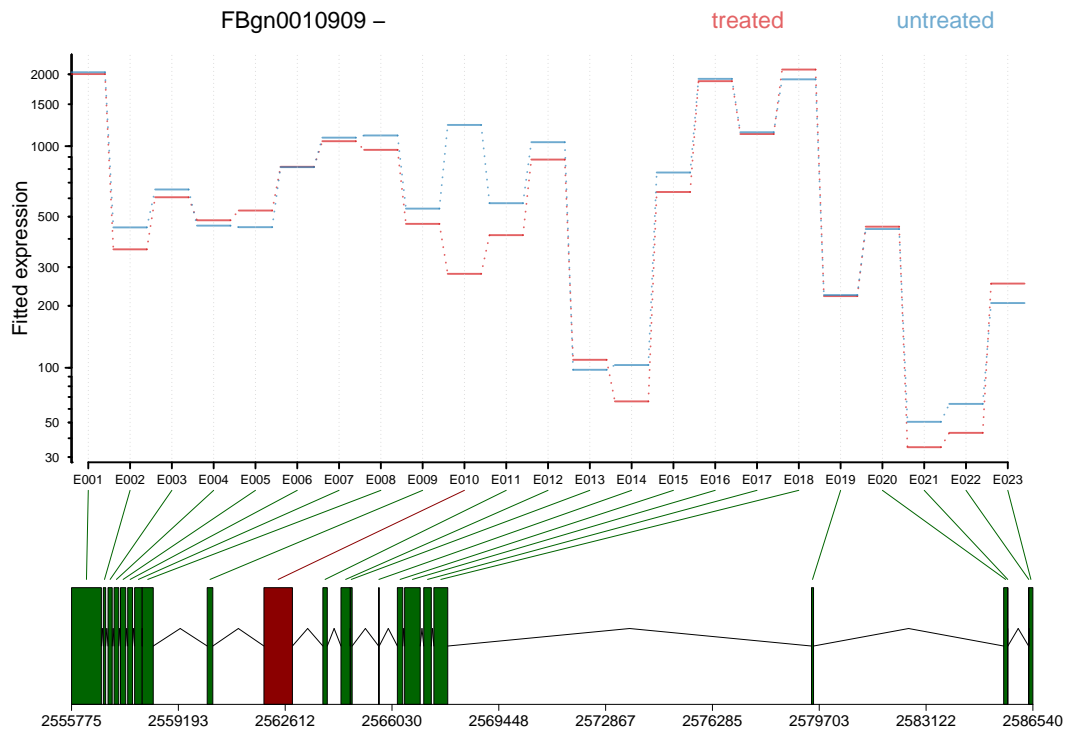


Figure 2: The plot represents the expression estimates from a call to `testForDEU`. Shown in red is the exon that showed significant differential exon usage.

```
> plotDEXSeq(pasillaExons, "FBgn0010909", cex.axis = 1.2, cex = 1.3,
+   lwd = 2, legend = TRUE)
```

The result is shown in Figure~2. Optionally, one can also visualize the transcript models (Figure~3), which might be useful for putting differential exon usage results into the context of isoform regulation.

```
> plotDEXSeq(pasillaExons, "FBgn0010909", displayTranscripts = TRUE,
+   cex.axis = 1.2, cex = 1.3, lwd = 2, legend = TRUE)
```

Another useful option is to look at the count values from the individual samples, rather than at the model effect estimates. For this display, it is useful to normalize the counts by the size factors (Figure~4).

```
> plotDEXSeq(pasillaExons, "FBgn0010909", coefficients = FALSE,
+   norCounts = TRUE, cex.axis = 1.2, cex = 1.3, lwd = 2, legend = TRUE)
```

To generate an easily browsable, detailed overview over all analysis results, the package provides an HTML report generator, implemented in the function `DEXSeqHTML`. This function uses the package `hwriter` to create a result table with links to plots for the significant results, allowing a more detailed exploration of the results. To see an example, visit <http://www.embl.de/~reyes/DEXSeqReport/testForDEU.html>. The report shown there was generated using this code.

```
> DEXSeqHTML(pasillaExons, FDR = 0.1, color = c("#FF00080", "#0000FF80"))
```

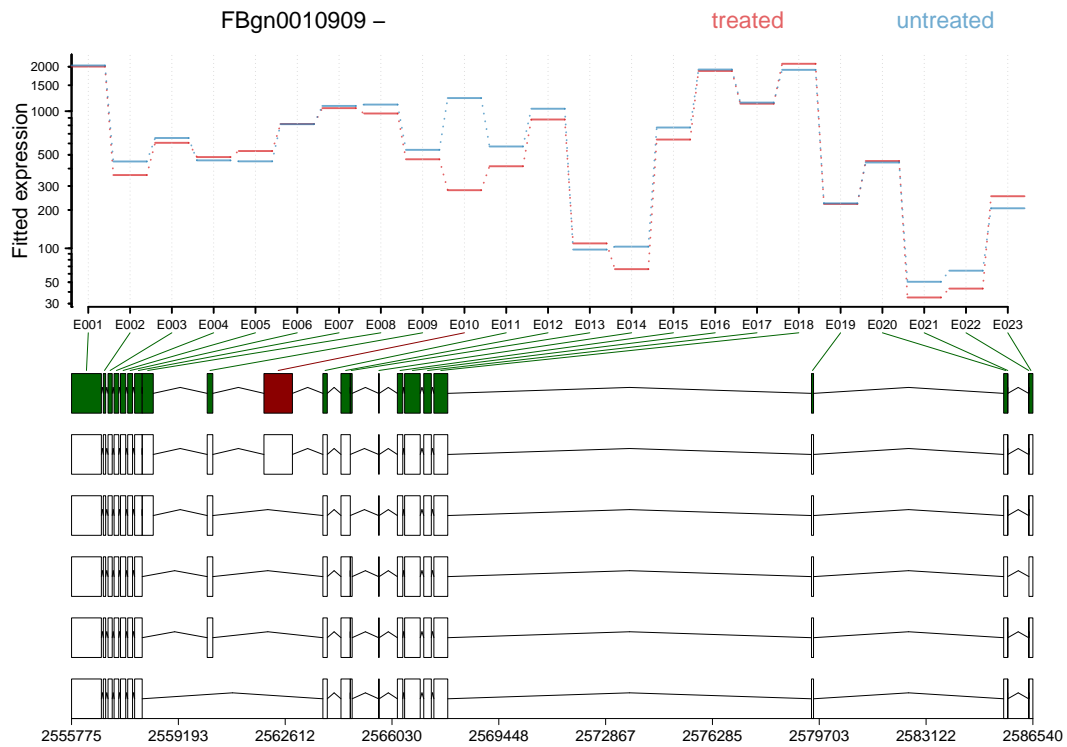


Figure 3: As in Figure 2, but including the annotated transcript models.

6 Gene count table

The function `geneCountTable` computes a table of *gene counts*, which are obtained by summing the counts from all exons with the same `geneID`. This might be useful for the detection of differential expression of genes, where the table can be used as input e. g. for the packages *DESeq* or *edgeR*.

```
> head(geneCountTable(pasillaExons))
```

| | treated1fb | treated2fb | treated3fb | untreated1fb | untreated2fb |
|-------------|--------------|--------------|------------|--------------|--------------|
| FBgn0000256 | 2285 | 2231 | 1889 | 2369 | 4675 |
| FBgn0000578 | 6748 | 5808 | 5520 | 6910 | 11338 |
| FBgn0002921 | 18397 | 19684 | 15691 | 15686 | 21674 |
| FBgn0003089 | 12 | 8 | 7 | 3 | 16 |
| FBgn0010226 | 212 | 248 | 228 | 184 | 252 |
| FBgn0010280 | 4085 | 4611 | 4225 | 4145 | 6597 |
| | untreated3fb | untreated4fb | | | |
| FBgn0000256 | 2191 | 2138 | | | |
| FBgn0000578 | 6042 | 5351 | | | |
| FBgn0002921 | 10953 | 11842 | | | |
| FBgn0003089 | 13 | 9 | | | |
| FBgn0010226 | 131 | 193 | | | |
| FBgn0010280 | 3914 | 3768 | | | |

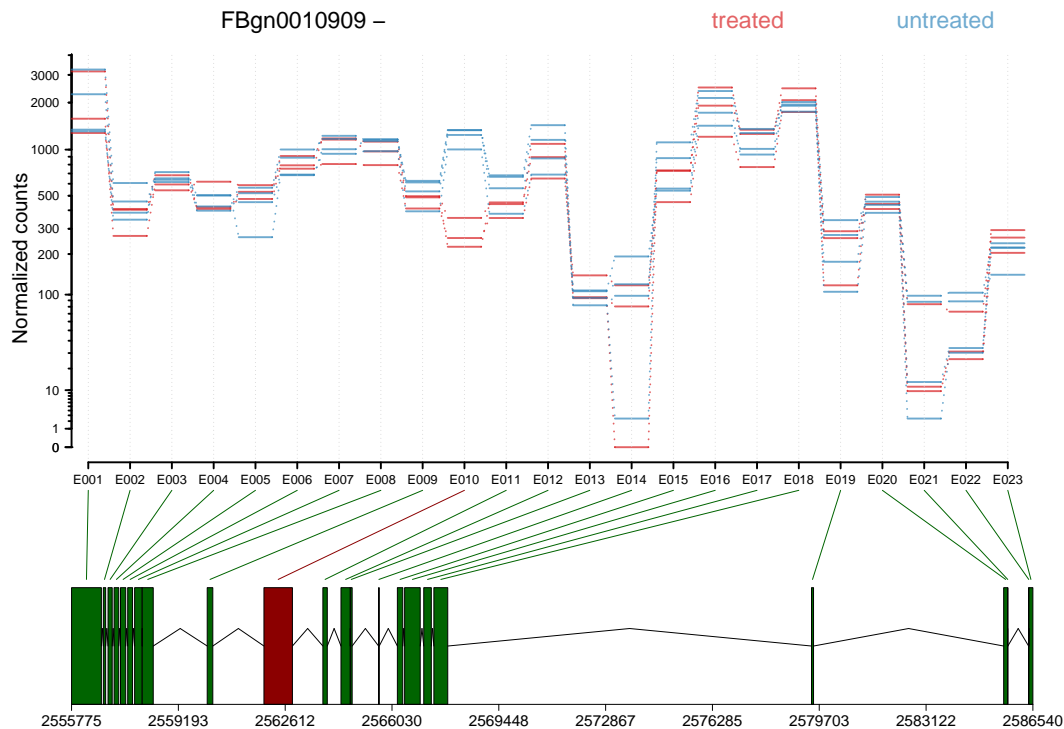


Figure 4: As in Figure 2, with normalized count values of each exon in each of the samples.

7 Creating *ExonCountSet* objects

7.1 From files produced by HTSeq

In this section, we describe how to create an *ExonCountSet* from an alignment of the RNA-seq reads to the genome, in SAM format, and a file describing gene and transcript models in GTF format.

The first steps of this workflow involve two scripts for the Python library *HTSeq*. These scripts are provided as part of the R package *DEXSeq*. The first script, `dexseq_prepare_annotation.py`, parses an annotation file in GTF format to define non-overlapping exonic regions: for instance, consider a gene whose transcripts contain either of two exons whose genomic regions overlap. In such a case, the script defines three exonic regions: two for the non-overlapping parts of each of the two exons, and a third one for the overlapping part. The script produces as output a new file in GTF format. The second script, `dexseq_count.py`, reads the GTF file produced by `dexseq_prepare_annotation.py` and an alignment in SAM format and counts the number of reads falling in each of the defined exonic regions.

The *DEXSeq* function `read.HTSeqCounts` is then able to read the output from these scripts and returns an *ExonCountSet* object with the relevant information for differential exon usage analysis and visualization. Of course, users can postprocess or replace the annotation in the object using their own means in R.

The files that were used in this way to create the `pasillaGenes` object are provided within the *pasilla* package:

```
> dir(system.file("extdata", package = "pasilla"))

[1] "Dmel.BDGP5.25.62.DEXSeq.chr.gff" "geneIDsinsubset.txt"
[3] "treated1fb.txt"                  "treated2fb.txt"
[5] "treated3fb.txt"                  "untreated1fb.txt"
[7] "untreated2fb.txt"                "untreated3fb.txt"
[9] "untreated4fb.txt"
```

The vignette² of the package *pasilla* provides a complete transcript of these steps.

7.2 From elementary R data structures

Users can also provide their own data, contained in elementary R objects, directly to the function `newExonCountSet` in order to create an *ExonCountSet* object. The minimum requirements are

1. a per-exon count matrix, with one row for every exon and one column for every sample,
2. a vector, matrix or data frame with information about the samples, and
3. two vectors of gene and exon identifiers that align with the rows of the count matrix.

With such a minimal object, it is possible to perform the analysis for differential exon usage, but the visualization functions will not be so useful. The necessary information about exons start and end positions can be given as a data frame to the `newExonCountSet` function, or can be added to the *ExonCountSet* object after its creation via the `featureData` accessor. For more information, please see the manual page of `newExonCountSet`.

```
> bare <- newExonCountSet(countData = counts(pasillaExons), design = design(pasillaExons),
+   geneIDs = geneIDs(pasillaExons), exonIDs = exonIDs(pasillaExons))
```

References

- [1] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010.
- [2] A.~N. Brooks, L.~Yang, M.~O. Duff, K.~D. Hansen, J.~W. Park, S.~Dudoit, S.~E. Brenner, and B.~R. Graveley. Conservation of an RNA regulatory map between *Drosophila* and mammals. *Genome Research*, pages 193–202, 2011.

8 Session Information

```
> sessionInfo()
```

```
R version 2.14.0 Under development (unstable) (2011-06-09 r56106)
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
locale:
```

```
[1] LC_CTYPE=en_US.utf8      LC_NUMERIC=C
[3] LC_TIME=en_US.utf8      LC_COLLATE=C
```

²Data preprocessing and creation of the data objects `pasillaGenes` and `pasillaExons`

```
[5] LC_MONETARY=en_US.utf8    LC_MESSAGES=en_US.utf8
[7] LC_PAPER=C                  LC_NAME=C
[9] LC_ADDRESS=C                LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.utf8  LC_IDENTIFICATION=C
```

attached base packages:

```
[1] stats      graphics  grDevices  utils      datasets  methods   base
```

other attached packages:

```
[1] DEXSeq_0.1.9  Biobase_2.13.6
```

loaded via a namespace (and not attached):

```
[1] MASS_7.3-13  hwriter_1.3  tools_2.14.0
```