HTSFilter: Data-based filtering for replicated transcriptome sequencing experiments

Andrea Rau, Mélina Gallopin, Gilles Celeux, Florence Jaffrézic

Modified: July 26, 2017. Compiled: April 27, 2020

Abstract

This vignette illustrates the use of the *HTSFilter* package to filter replicated data from transcriptome sequencing experiments (e.g., RNA sequencing data) for a variety of different data classes: *matrix*, *data.frame*, the S3 classes associated with the *edgeR* package (*DGEExact* and *DGELRT*), and the S4 class associated with the *DESeq2* package (*DESeqDataSet*).

Contents

1 Introduction

High-throughput sequencing (HTS) data, such as RNA-sequencing (RNA-seq) data, are increasingly used to conduct differential analyses, in which statistical tests are performed for each biological feature (e.g., a gene, transcript, exon) in order to identify those whose expression levels show systematic covariation with a particular condition, such as a treatment or phenotype of interest. For the remainder of this vignette, we will focus on gene-level differential analyses, although these methods may also be applied to differential analyses of (count-based measures of) transcript- or exon-level expression.

Because hypothesis tests are performed for gene-by-gene differential analyses, the obtained p-values must be adjusted to correct for multiple testing. However, procedures to adjust p-values to control the number of detected false positives often lead to a loss of power to detect truly differentially expressed (DE) genes due to the large number of hypothesis tests performed. To reduce the impact of such procedures, independent data filters are often used to identify and remove genes that appear to generate an uninformative signal [1]; this in turn moderates the correction needed to adjust for multiple testing. For independent filtering methods for microarray data, see for example the *genefilter* Bioconductor package [2].

The *HTSFilter* package implements a novel data-based filtering procedure based on the calculation of a similarity index among biological replicates for read counts arising from replicated transcriptome sequencing (RNA-seq) data. This technique provides an intuitive data-driven way to filter high-throughput transcriptome sequencing data and to effectively remove genes with low, constant expression levels without incorrectly removing those that would otherwise have been identified as DE. The two fundamental assumptions of the filter implemented in the *HTSFilter* package are as follows:

1. Biological replicates are present for each experimental condition, and
2. Data can be appropriately normalized (scaled) to correct for systematic inter-sample biases.

Assuming these conditions hold, *HTSFilter* implements a method to identify a filtering threshold that maximizes the *filtering similarity* among replicates, that is, one where most genes tend to either have normalized counts less than or equal to the cutoff in all samples (i.e., filtered genes) or greater than the cutoff in all samples (i.e., non-filtered genes). This filtering similarity is defined using the global Jaccard index, that is, the average Jaccard index calculated between pairs of replicates within each experimental condition; see Rau *et al.* (2013) [3] for more details.

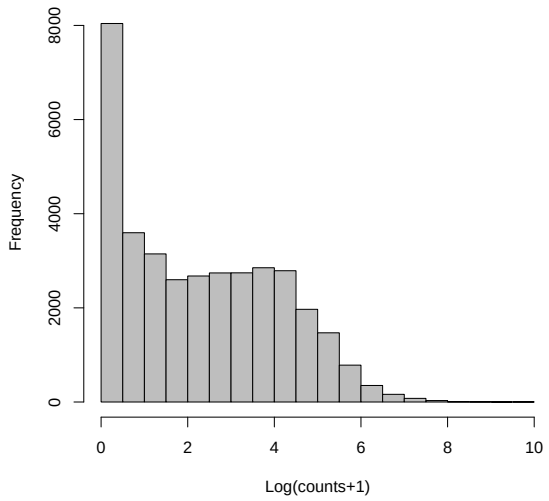

Figure 1: Histogram of log transformed counts from the Sultan *et al.* data [9], illustrating the large number of genes with very small counts as well as the large heterogeneity in counts observed

For more information about between-sample normalization strategies, see [4]; in particular, strategies for normalizing data with differences in library size and composition may be found in [5] and [6], and strategies for normalizing data exhibiting sample-specific biases due to GC content may be found in [7] and [8]. Within the *HTSFilter* package, the Trimmed Means of M-values (TMM) [6] and DESeq [5] normalization strategies may be used prior to calculating an appropriate data-based filter. If an alternative normalization strategy is needed or desired, the normalization may be applied prior to filtering the data with `normalization="none"` in the `HTSFilter` function; see Section 6 for an example.

The *HTSFilter* package is able to accommodate unnormalized or normalized replicated count data in the form of a *matrix* or *data.frame* (in which each row corresponds to a biological feature and each column to a biological sample), one of the S3 classes associated with the *edgeR* package (*DGEList*, *DGEExact*, *DGEGLM*, and *DGELRT*), or *DESeqDataSet* (the S4 class associated with the *DESeq2* package), as illustrated in the following sections.

Finally, we note that the filtering method implemented in the *HTSFilter* package is designed to filter transcriptome sequencing, and not microarray, data; in particular, the proposed filter is effective for data with features that take on values over a large order of magnitude and with a subset of features exhibiting small levels of expression across samples (see, for example, Figure 1). In this vignette, we illustrate its use on count-based measures of gene expression, although its use is not strictly limited to discrete data.

2 Input data

For the purposes of this vignette, we make use of data from a study of sex-specific expression of liver cells in human and the *DESeq* and *edgeR* packages for differential analysis. Sultan *et al.* [9] obtained a high-throughput sequencing data (using a 1G Illumina Genome Analyzer sequencing machine) from a human embryonic kidney and a B cell line, with two biological replicates each. The raw read counts and phenotype tables were obtained from the ReCount online resource [10].

To begin, we load the *HTSFilter* package, and attach the gene-level count data contained in `sultan`:

```
> library(HTSFilter)
> library(edgeR)
> library(DESeq2)
> data("sultan")
> hist(log(exprs(sultan)+1), col="grey", breaks=25, main="",
+   xlab="Log(counts+1)")
> pData(sultan)

          sample.id num.tech.reps    cell.line
SRX008333 SRX008333             1 Ramos B cell
SRX008334 SRX008334             1 Ramos B cell
SRX008331 SRX008331             1      HEK293T
SRX008332 SRX008332             1      HEK293T

> dim(sultan)

Features  Samples
    9010        4
```

The unfiltered data contain 9010 genes in four samples (two replicates per condition).

3 *matrix* and *data.frame* classes

To filter high-throughput sequencing data in the form of a *matrix* or *data.frame*, we first access the expression data, contained in `exprs(sultan)`, and create a vector identifying the condition labels for each of the samples via the `pData` Biobase function. We then filter the data using the `HTSFilter` function, specifying that the number of tested thresholds be only 25 (`s.len=25`) rather than the default value of 100 to reduce computation time for this example. Note that as it is unspecified, the default normalization method is used for filtering the data,

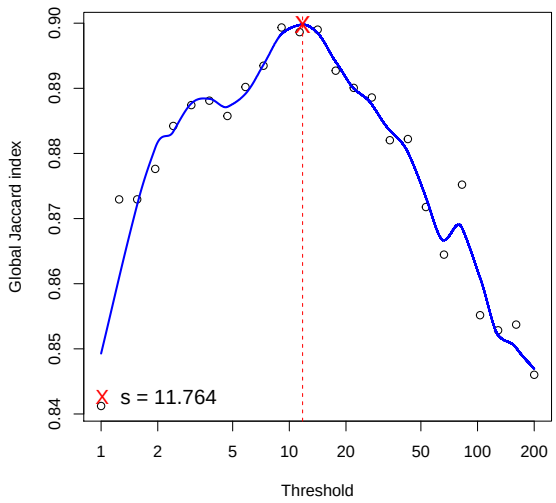

Figure 2: Global Jaccard index for the `sultan` data calculated for a variety of threshold values after TMM normalization [6], with a loess curve (blue line) superposed and data-based threshold values (red cross and red dotted line) equal to 11.764

namely the Trimmed Mean of M-values (TMM) method of Robinson and Oshlack [6]. To use the DESeq normalization method [5], `normalization="DESeq"` may be specified.

```
> mat <- exprs(sultan)
> conds <- as.character(pData(sultan)$cell.line)
> ## Only 25 tested thresholds to reduce computation time
> filter <- HTSFilter(mat, conds, s.min=1, s.max=200, s.len=25)
> mat <- filter$filteredData
> dim(mat)

[1] 4995    4

> dim(filter$removedData)

[1] 4015    4
```

For this example, we find a data-based threshold equal to 11.764; genes with normalized values less than this threshold in all samples are filtered from subsequent analyses. The proposed filter thus removes 4015 genes from further analyses, leaving 4995 genes.

We note that an important part of the filter proposed in the *HTSFilter* package is a check of the behavior of the global similarity index calculated over a range of threshold values, and in particular, to verify that a reasonable maximum value is

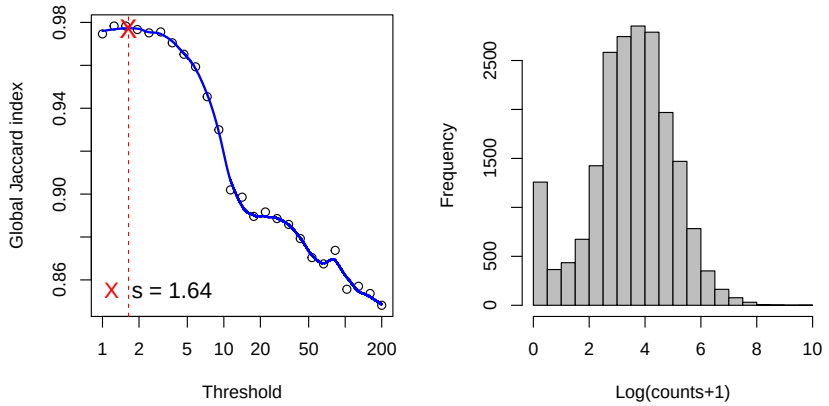

Figure 3: (left) Global Jaccard index for the `sultan` data calculated for a variety of threshold values after TMM normalization [6], with a loess curve (blue line) superposed and data-based threshold values (red cross and red dotted line) equal to 11.764
(right) Global Jaccard index for the previously filtered `sultan` data, with loess curve (blue line) superposed as before.

reached for the global similarity index over the range of tested threshold values (see Figure 2); the maximum possible value for the global Jaccard index is nearly 1. To illustrate the importance of this check, we attempt to re-apply the proposed filter to the previously filtered data (in practice, of course, this would be nonsensical):

```
> par(mfrow = c(1,2), mar = c(4,4,2,2))
> filter.2 <- HTSFilter(mat, conds, s.len=25)
> dim(filter.2$removedData)

[1] 0 4

> hist(log(filter.2$filteredData+1), col="grey", breaks=25, main="",
+   xlab="Log(counts+1)")
```

In the lefthand panel of Figure 3, we note a plateau of large global Jaccard index values for thresholds less than 2, with a decrease thereafter; this corresponds to filtering no genes, unsurprising given that genes with low, constant levels of expression have already been filtered from the analysis (see the righthand panel of Figure 3).

4 *edgeR* package pipeline

We next illustrate the use of *HTSFilter* within the *edgeR* pipeline for differential analysis (S3 classes *DGEList*, *DGEExact*, *DGEGLM*, or *DGELRT*). For the purposes of this vignette, we will consider the S3 classes *DGEExact* and *DGELRT*. The former is the class containing the results of the differential expression analysis between two groups of count libraries (resulting from a call to the function `exactTest` in *edgeR*); the latter is the class containing the results of a generalized linear model (GLM)-based differential analysis (resulting from a call to the function `glmLRT` in *edgeR*). Although the filter may be applied earlier in the *edgeR* pipeline (i.e., to objects of class *DGEList* or *DGEGLM*), we do not recommend doing so, as parameter estimation makes use of counts adjusted using a a quantile-to-quantile method (pseudo-counts).

4.1 S3 class *DGEExact*

We first coerce the data into the appropriate class with the function `DGEList`, where the `group` variable is set to contain a vector of condition labels for each of the samples. Next, after calculating normalizing factors to scale library sizes (`calcNormFactors`), we estimate common and tagwise dispersion parameters using `estimateDisp` (using the quantile conditional likelihood for each gene) and obtain differential analysis results using `exactTest`. Finally, we apply the filter using the `HTSFilter` function, again specifying that the number of tested thresholds be only 25 (`s.len=25`) rather than the default value of 100. Note that as it is unspecified, the default normalization method is used for filtering the data, namely the Trimmed Mean of M-values (TMM) method [6]; alternative normalization, including `"pseudo.counts"` for the quantile-to-quantile adjusted counts used for parameter estimation, may also be specified. We suppress the plot of the global Jaccard index using `plot = FALSE`, as it is identical to that shown in Figure 2.

```
> dge <- DGEList(counts=exprs(sultan), group=conds)
> dge <- calcNormFactors(dge)
> dge <- estimateDisp(dge)

Design matrix not provided. Switch to the classic mode.

> et <- exactTest(dge)
> et <- HTSFilter(et, DGEList=dge, s.len=25, plot=FALSE)$filteredData
> dim(et)

[1] 4995    3

> class(et)
```

```
[1] "DGEExact"
attr(,"package")
[1] "edgeR"
```

```
> topTags(et)
```

```
Comparison of groups:  Ramos B cell-HEK293T
                     logFC   logCPM        PValue
ENSG00000100721  14.399287 11.53328  0.000000e+00
ENSG00000133124 -14.394458 11.56789  0.000000e+00
ENSG00000105369  11.925779 11.23367  0.000000e+00
ENSG00000135144  10.921892 11.05896 2.600818e-309
ENSG00000110777  13.850554 10.97744 2.014098e-302
ENSG00000111348  13.797192 10.92306 6.278069e-298
ENSG00000118308  13.494132 10.61473 1.294723e-272
ENSG00000177606  -9.731885 10.81415 1.456204e-263
ENSG00000012124  10.171666 10.29617 3.995433e-247
ENSG00000149418  10.899611 10.19056 6.189743e-239
                         FDR
ENSG00000100721  0.000000e+00
ENSG00000133124  0.000000e+00
ENSG00000105369  0.000000e+00
ENSG00000135144 3.247771e-306
ENSG00000110777 2.012083e-299
ENSG00000111348 5.226492e-295
ENSG00000118308 9.238774e-270
ENSG00000177606 9.092173e-261
ENSG00000012124 2.217465e-244
ENSG00000149418 3.091777e-236
```

Note that the filtered data are of the class *DGEExact*, allowing for a call to the `topTags` function.

```
> topTags(et)
```

```
Comparison of groups:  Ramos B cell-HEK293T
                     logFC   logCPM        PValue
ENSG00000100721  14.399287 11.53328  0.000000e+00
ENSG00000133124 -14.394458 11.56789  0.000000e+00
ENSG00000105369  11.925779 11.23367  0.000000e+00
ENSG00000135144  10.921892 11.05896 2.600818e-309
ENSG00000110777  13.850554 10.97744 2.014098e-302
ENSG00000111348  13.797192 10.92306 6.278069e-298
ENSG00000118308  13.494132 10.61473 1.294723e-272
```

```
ENSG00000177606  -9.731885 10.81415 1.456204e-263
ENSG00000012124  10.171666 10.29617 3.995433e-247
ENSG00000149418  10.899611 10.19056 6.189743e-239
                          FDR
ENSG00000100721  0.000000e+00
ENSG00000133124  0.000000e+00
ENSG00000105369  0.000000e+00
ENSG00000135144 3.247771e-306
ENSG00000110777 2.012083e-299
ENSG00000111348 5.226492e-295
ENSG00000118308 9.238774e-270
ENSG00000177606 9.092173e-261
ENSG00000012124 2.217465e-244
ENSG00000149418 3.091777e-236
```

4.2 S3 class *DGELRT*

We follow the same steps as the previous example, where the `estimateDisp` function is now used to obtain per-gene dispersion parameter estimates using the adjusted profile loglikelihood, the `glmFit` function is used to fit a negative binomial generalized log-linear model to the read counts for each gene, and the `glmLRT` function is used to conduct likelihood ratio tests for one or more coefficients in the GLM. The output of `glmLRT` is an S3 object of class *DGELRT* and contains the GLM differential analysis results. As before, we apply the filter using the `HTSFilter` function, again suppressing the plot of the global Jaccard index using `plot = FALSE`, as it is identical to that shown in Figure 2.

```
> design <- model.matrix(~conds)
> dge <- DGEList(counts=exprs(sultan), group=conds)
> dge <- calcNormFactors(dge)
> dge <- estimateDisp(dge, design)
> fit <- glmFit(dge,design)
> lrt <- glmLRT(fit,coef=2)
> lrt <- HTSFilter(lrt, DGEGLM=fit, s.len=25, plot=FALSE)$filteredData
> dim(lrt)

[1] 4995    4

> class(lrt)

[1] "DGELRT"
attr(,"package")
[1] "edgeR"
```

Note that the filtered data are of the class *DGEList*, allowing for a call to the `topTags` function.

```
> topTags(lrt)

Coefficient:  condsRamos B cell
                     logFC   logCPM       LR        PValue
ENSG00000100721  14.399291 11.53265 1711.214  0.000000e+00
ENSG00000133124 -14.394459 11.56849 1554.769  0.000000e+00
ENSG00000110777  13.850562 10.97659 1487.284  0.000000e+00
ENSG00000105369  11.925777 11.23291 1576.317  0.000000e+00
ENSG00000135144  10.921894 11.05814 1495.598  0.000000e+00
ENSG00000111348  13.797195 10.92217 1465.781 1.067182e-320
ENSG00000118308  13.494128 10.61369 1345.242 1.666173e-294
ENSG00000177606  -9.731884 10.81502 1222.977 6.192499e-268
ENSG00000012124  10.171667 10.29498 1199.883 6.468477e-263
ENSG00000149418  10.899617 10.18933 1171.186 1.115284e-256
                          FDR
ENSG00000100721  0.000000e+00
ENSG00000133124  0.000000e+00
ENSG00000110777  0.000000e+00
ENSG00000105369  0.000000e+00
ENSG00000135144  0.000000e+00
ENSG00000111348 8.884288e-318
ENSG00000118308 1.188933e-291
ENSG00000177606 3.866441e-265
ENSG00000012124 3.590005e-260
ENSG00000149418 5.570841e-254
```

5 *DESeq2* package pipeline: S4 class *DESeqDataSet*

The *HTSFilter* package allows for a straightforward integration within the *DESeq2* analysis pipeline, most notably allowing for p-values to be adjusted only for those genes passing the filter. Note that *DESeq2* now impelements an independent filtering procedure by default in the `results` function; this filter is a potential alternative filtering technique and does not need to be used in addition to the one included in *HTSFilter*. In fact, each filter is targeting the same weakly expressed genes to be filtered from the analysis. As such, if the user wishes to make use of *HTSFilter* within the *DESeq2* pipeline, the argument `independentFiltering=FALSE` should be used when calling the `results` function in *DESeq2*.

To illustrate the application of a filter for high-throughput sequencing data in the form of a *DESeqDataSet* (the class used within the *DESeq2* pipeline for differential analysis), we coerce `sultan` into an object of the class *DESeqDataSet* using the function `DESeqDataSetFromMatrix`. Once again, we specify that the number of tested thresholds be only 25 (`s.len=25`) rather than the default value of 100 to reduce computation time. For objects in the form of a *DESeqDataSet*, the default normalization strategy is `"DESeq"`, although alternative normalization strategies may also be used. Note that below we replace the spaces in condition names with "." prior to creating a `"DESeqDataSet"` object.

```
> conds <- gsub(" ", ".", conds)
> dds <- DESeqDataSetFromMatrix(countData = exprs(sultan),
+                               colData = data.frame(cell.line = conds),
+                               design = ~ cell.line)
> dds <- DESeq(dds)
> filter <- HTSFilter(dds, s.len=25, plot=FALSE)$filteredData
> class(filter)
> dim(filter)
> res <- results(filter, independentFiltering=FALSE)
> head(res)
>
```

The filtered data remain an object of class *DESeqDataSet*, and subsequent functions from *DESeq2* (such as the results summary function `results`) may be called directly upon it.

6 Alternative normalization using *EDAseq*

As a final example, we illustrate the use of the *HTSFilter* package with an alternative normalization strategy, namely the full quantile normalization method in the *EDASeq* package; such a step may be useful when the TMM or DESeq normalization methods are not appropriate for a given dataset. Once again, we create a new object of the appropriate class with the function `newSeqExpressionSet` and normalize data using the `betweenLaneNormalization` function (with `which="full"`) in *EDASeq*.

```
> library(EDASeq)
> ses <- newSeqExpressionSet(exprs(sultan),
+         phenoData=pData(sultan))
> ses.norm <- betweenLaneNormalization(ses, which="full")
```

Subsequently, `HTSFilter` is applied to the normalized data (again using `s.len=25`), and the normalization method is set to `norm="none"`. We may then make use of the `on` vector in the results, which identifies filtered and unfiltered genes (respectively) with 0 and 1, to identify rows in the original data matrix to be retained.

```
> filter <- HTSFilter(counts(ses.norm), conds, s.len=25, norm="none",
+           plot=FALSE)
> head(filter$on)
> table(filter$on)
```

7 Session Info

```
> sessionInfo()

R version 4.0.0 (2020-04-24)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2012 R2 x64 (build 9600)

Matrix products: default

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices
[6] utils     datasets  methods   base

other attached packages:
 [1] DESeq2_1.28.0               SummarizedExperiment_1.18.0
 [3] DelayedArray_0.14.0         matrixStats_0.56.0
 [5] Biobase_2.48.0              GenomicRanges_1.40.0
 [7] GenomeInfoDb_1.24.0         IRanges_2.22.0
 [9] S4Vectors_0.26.0            BiocGenerics_0.34.0
[11] edgeR_3.30.0                limma_3.44.0
[13] HTSFilter_1.28.0
```

```
loaded via a namespace (and not attached):
 [1] Rcpp_1.0.4.6           locfit_1.5-9.4
 [3] lattice_0.20-41        assertthat_0.2.1
 [5] digest_0.6.25          R6_2.4.1
 [7] RSQLite_2.2.0          DESeq_1.40.0
 [9] evaluate_0.14          ggplot2_3.3.0
[11] pillar_1.4.3           zlibbioc_1.34.0
[13] rlang_0.4.5            annotate_1.66.0
[15] blob_1.2.1             Matrix_1.2-18
[17] rmarkdown_2.1          splines_4.0.0
[19] BiocParallel_1.22.0    geneplotter_1.66.0
[21] RCurl_1.98-1.2         bit_1.1-15.2
[23] munsell_0.5.0          compiler_4.0.0
[25] xfun_0.13              pkgconfig_2.0.3
[27] htmltools_0.4.0        tidyselect_1.0.0
[29] tibble_3.0.1           GenomeInfoDbData_1.2.3
[31] XML_3.99-0.3           crayon_1.3.4
[33] dplyr_0.8.5            bitops_1.0-6
[35] grid_4.0.0             xtable_1.8-4
[37] gtable_0.3.0           lifecycle_0.2.0
[39] DBI_1.1.0              magrittr_1.5
[41] scales_1.1.0           XVector_0.28.0
[43] genefilter_1.70.0      ellipsis_0.3.0
[45] vctrs_0.2.4            BiocStyle_2.16.0
[47] RColorBrewer_1.1-2     tools_4.0.0
[49] bit64_0.9-7            glue_1.4.0
[51] purrr_0.3.4            survival_3.1-12
[53] yaml_2.2.1             AnnotationDbi_1.50.0
[55] colorspace_1.4-1       BiocManager_1.30.10
[57] memoise_1.1.0          knitr_1.28
```

References

[1] Richard Bourgon, Robert Gentleman, and Wolfgang Huber. Independent filtering increases detection power for high-throughput experiments. *PNAS*, 107(21):9546–9551, 2010.

[2] R. Gentleman, V. Carey, W. Huber, and F. Hahne. *genefilter: methods for filtering genes from microarray experiments*. R package version 1.38.0.

[3] A. Rau, M. Gallopin, G. Celeux, and F. Jaffrézic. Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics*, doi: 10.1093/bioinformatics/btt350, 2013.

[4] Marie-Agnès Dillies, Andrea Rau, Julie Aubert, Christelle Hennequet-Antier, Marine Jeanmougin, Nicolas Servant, Céline Keime, Guillemette Marot, David Castel, Jordi Estelle, Gregory Guernec, Bernd Jagla, Luc Jouneau, Denis Laloë, Caroline Le Gall, Brigitte Schaëffer, Stéphane Le Crom, and Florence Jaffrézic. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Briefings in Bioinformatics*, (in press), 2012.

[5] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(R106):1–28, 2010.

[6] Mark D. Robinson and Alicia Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, 11(R25), 2010.

[7] Davide Risso, Katja Schwartz, Gavin Sherlock, and Sandrine Dudoit. GC-content normalization for RNA-seq data. *BMC Bioinformatics*, 12(480), 2011.

[8] Kasper D. Hansen, Rafael A. Irizarry, and Zhijin Wu. Removing technical variability in RNA-seq data using conditional quantile normalization. *Biostatistics*, (in press)(227), 2012.

[9] M. Sultan, M. H. Schulz, H. Richard, A. Magen, A. Klingenhoff, M. Scherf, M. Seifert, T. Borodina, A. Soldatov, D. Parkhomchuk, D. Schmidt, S. O'Keeffe, S. Haas, M. Vingron, H. Lehrach, and M. L. Yaspo. A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome. *Science*, 15(5891):956–60, 2008.

[10] A. C. Frazee, B. Langmead, and J. T. Leek. ReCount: a multi-experiment resource of analysis-ready RNA-seq gene count datasets. *BMC Bioinformatics*, 12(449), 2011.