

HTSFilter: Data-based filtering for replicated transcriptome sequencing experiments

Andrea Rau, Méлина Gallopin, Gilles Celeux, Florence Jaffrézic

Modified: December 10, 2013. Compiled: October 13, 2014

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*, *CountDataSet* (the S4 class associated with the *DESeq* package), the S3 classes associated with the *edgeR* package (*DGEList*, *DGEEexact*, *DGEGLM*, and *DGELRT*), and the S4 class associated with the *DESeq2* package (*DESeqDataSet*).

Contents

1	Introduction	2
2	Input data	4
3	<i>matrix</i> and <i>data.frame</i> classes	4
4	<i>DESeq</i> package pipeline: S4 class <i>CountDataSet</i>	7
5	<i>edgeR</i> package pipeline	9
	5.1 S3 class <i>DGEEexact</i>	9
	5.2 S3 class <i>DGELRT</i>	11
6	<i>DESeq2</i> package pipeline: S4 class <i>DESeqDataSet</i>	12
7	Alternative normalization using <i>EDAseq</i>	14
8	Session Info	14

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 [2]; 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 [5].

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 (author?) [7] for more details.

For more information about between-sample normalization strategies, see [3]; in particular, strategies for normalizing data with differences in library size and composition may be found in [1] and [9], and strategies for normalizing data exhibiting sample-specific biases due to GC content may be found in [8] and [6]. Within the *HTSFilter* package, the Trimmed Means of M-values (TMM) [9] and DESeq [1] normalization strategies may

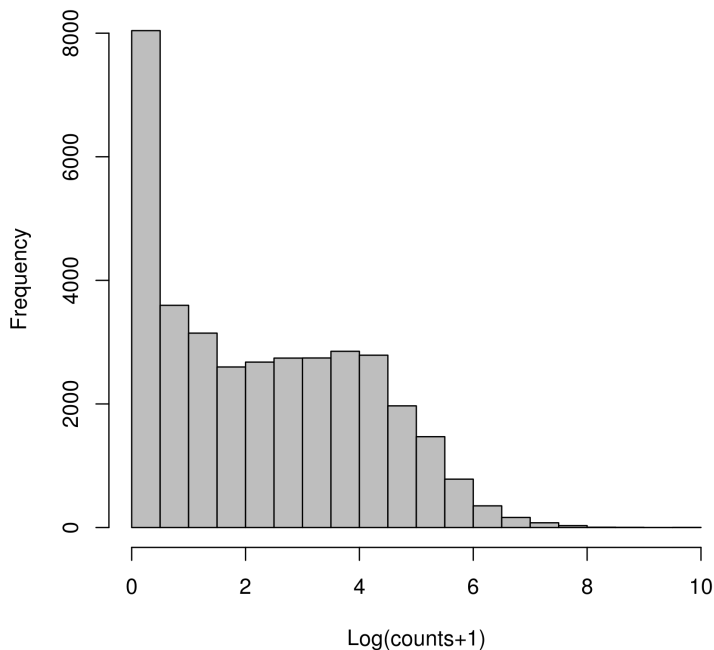


Figure 1: Histogram of log transformed counts from the Sultan et al. data [10], illustrating the large number of genes with very small counts as well as the large heterogeneity in counts observed.

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 7 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), a *CountDataSet* (the S4 class associated with the *DESeq* package), one of the S3 classes associated with the *edgeR* package (*DGEList*, *DGEEexact*, *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. [10] 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 [4].

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

```
> library(HTSFilter)
> library(DESeq)
> 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, namely the Trimmed Mean of M-values (TMM) method of Robinson and Oshlack [9]. To use the `DESeq` normalization method [1], `normalization="DESeq"` may be specified.

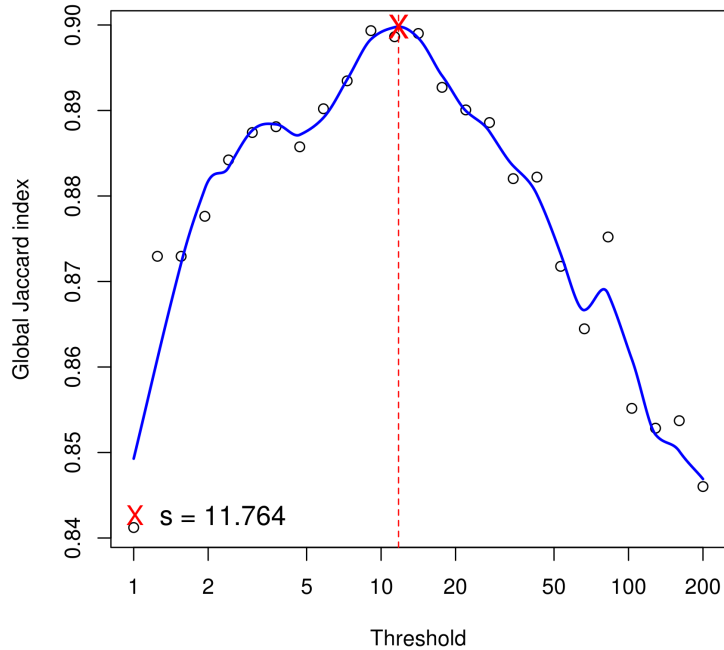


Figure 2: Global Jaccard index for the `sultan` data calculated for a variety of threshold values after TMM normalization [9], with a loess curve (blue line) superposed and data-based threshold values (red cross and red dotted line) equal to 11.764.

```
> mat <- exprs(sultan)
> conds <- pData(sultan)$cell.line
> ## Only 25 tested thresholds to reduce computation time
> filter <- HTSFilter(mat, conds, s.len=25)
> mat <- filter$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

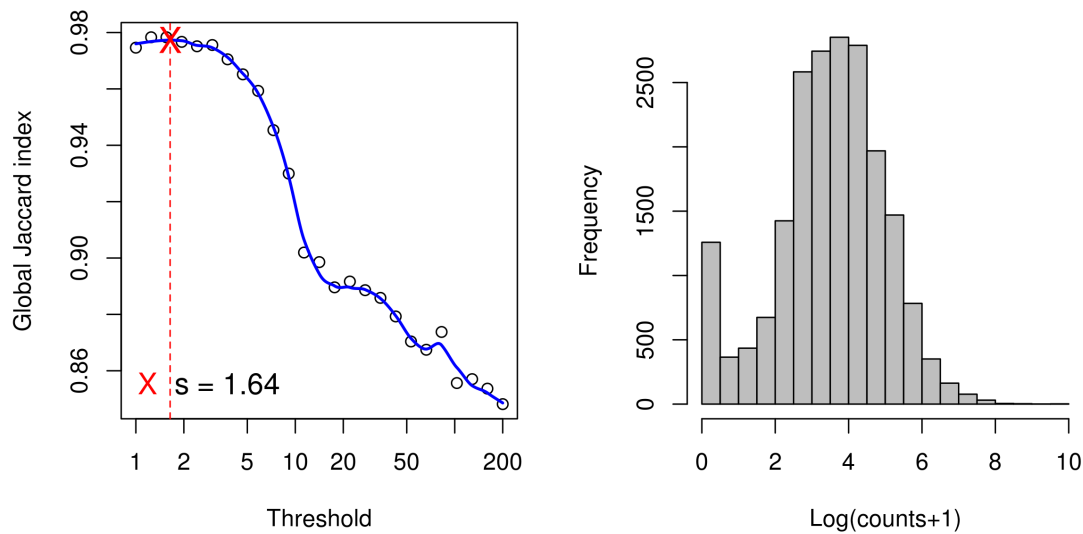


Figure 3: (left) Global Jaccard index for the `sultan` data calculated for a variety of threshold values after TMM normalization [9], with a loess curve (blue line) superposed and data-based threshold values (red cross and red dotted line) equal to 11.764. (right) Global Jaccard index for the previously filtered `sultan` data, with loess curve (blue line) superposed as before.

values, and in particular, to verify that a reasonable maximum value is 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 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 *DESeq* package pipeline: S4 class *CountDataSet*

The *HTSFilter* package allows for three potential applications of the proposed filter within the *DESeq* analysis pipeline:

1. Estimation of library sizes and dispersion parameters (the `estimateSizeEffects` and `estimateDispersions` functions in *DESeq*), followed by data filtering on normalized data (recommended);
2. Estimation of library sizes (`estimateSizeEffects`), data filtering on normalized data, and estimation of dispersion parameters (`estimateDispersions`);
3. Data filtering on normalized data, followed by re-estimation of library sizes and estimation of dispersion parameters (`estimateSizeEffects` and `estimateDispersions`).

We note that the primary difference among the three strategies would be seen in the dispersion parameters estimates for genes with low levels of expression; because fitted dispersion values are estimated based on the mean-dispersion relationship observed across the full data in the *DESeq* package, estimates obtained on filtered data will necessarily be slightly different from those obtained on unfiltered data, as genes with low levels of expression would have been removed from the former. On the other hand, we note that the filtering thresholds (and as such, the genes filtered from the analysis) are identical for the three strategies listed above. The estimated library sizes may differ slightly in the third strategy as compared to the first and second strategies; however, this difference will be minimal as only genes with weak, constant levels of expression are filtered from the analysis. A full discussion of these three strategies is beyond the scope of this vignette; however, in practice we recommend the use of the first strategy: estimation of library sizes and dispersion parameters prior to data filtering.

To filter high-throughput sequencing data in the form of a *CountDataSet* (the class used within the *DESeq* pipeline for differential analysis), we coerce `sultan` into an object of the class *CountDataSet*. 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 *CountDataSet*, the default normalization strategy is "DESeq", although alternative normalization strategies may also be used.

```
> cds <- newCountDataSet(exprs(sultan), conds)
> cds <- estimateSizeFactors(cds)
> cds <- estimateDispersions(cds)
> ## Only 25 tested thresholds to reduce computation time
> cds <- HTSFilter(cds, s.len=25)$filteredData
> res <- nbinomTest(cds, levels(conds)[1], levels(conds)[2])
> class(cds)
```

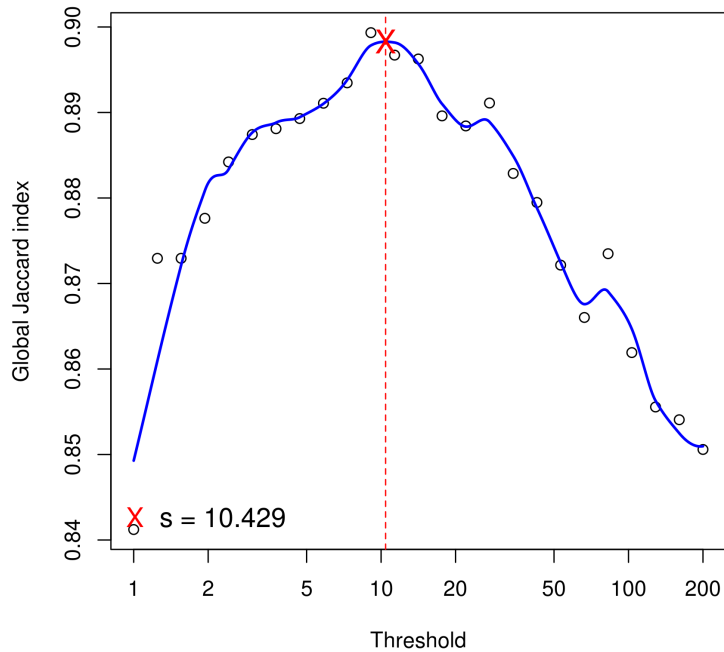


Figure 4: Global Jaccard index for the `sultan` data calculated for a variety of threshold values after DESeq normalization [1], with a loess curve (blue line) superposed and data-based threshold values (red cross and red dotted line) equal to 10.429.

```
[1] "CountDataSet"
attr("package")
[1] "DESeq"
```

```
> dim(cds)
```

```
Features Samples
      5143      4
```

As the normalization strategy used here was slightly different, the proposed filter now removes 3867 genes from further analyses, leaving 5143 genes. Again we verify the behavior of the global similarity index calculated over a range of threshold values (see Figure 4). For this example, we find a data-based threshold equal to 10.429; genes with normalized values less than this threshold in all samples are filtered from subsequent analyses.

5 *edgeR* package pipeline

We next illustrate the use of *HTSFilter* within the *edgeR* pipeline for differential analysis (S3 classes *DGEList*, *DGEEexact*, *DGEGLM*, or *DGELRT*). For the purposes of this vignette, we will consider the S3 classes *DGEEexact* 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).

5.1 S3 class *DGEEexact*

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 `estimateCommonDisp` and `estimateTagwiseDisp` 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 [9]; 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 <- estimateCommonDisp(dge)
> dge <- estimateTagwiseDisp(dge)
> et <- exactTest(dge)
> et <- HTSFilter(et, DGEList=dge, s.len=25, plot=FALSE)$filteredData
> dim(et)

[1] 4995    3

> class(et)

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

```
> topTags(et)
```

```
Comparison of groups: Ramos B cell-HEK293T
```

	logFC	logCPM	PValue
ENSG00000133124	-14.394468	11.56788	0.000000e+00
ENSG00000105369	11.925756	11.23368	0.000000e+00
ENSG00000135144	10.921903	11.05897	0.000000e+00
ENSG00000111348	13.797203	10.92307	1.299113e-315
ENSG00000177606	-9.731883	10.81414	2.902641e-299
ENSG00000012124	10.171680	10.29618	2.879471e-286
ENSG00000118308	13.494132	10.61474	4.245092e-272
ENSG00000046604	-12.906279	10.10888	1.234351e-242
ENSG00000166165	-8.648658	10.13930	5.232947e-230
ENSG00000100721	14.399178	11.53329	7.945829e-230
	FDR		
ENSG00000133124	0.000000e+00		
ENSG00000105369	0.000000e+00		
ENSG00000135144	0.000000e+00		
ENSG00000111348	1.622267e-312		
ENSG00000177606	2.899739e-296		
ENSG00000012124	2.397160e-283		
ENSG00000118308	3.029177e-269		
ENSG00000046604	7.706978e-240		
ENSG00000166165	2.904286e-227		
ENSG00000100721	3.968942e-227		

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

```
> topTags(et)
```

```
Comparison of groups: Ramos B cell-HEK293T
```

	logFC	logCPM	PValue
ENSG00000133124	-14.394468	11.56788	0.000000e+00
ENSG00000105369	11.925756	11.23368	0.000000e+00
ENSG00000135144	10.921903	11.05897	0.000000e+00
ENSG00000111348	13.797203	10.92307	1.299113e-315
ENSG00000177606	-9.731883	10.81414	2.902641e-299
ENSG00000012124	10.171680	10.29618	2.879471e-286
ENSG00000118308	13.494132	10.61474	4.245092e-272
ENSG00000046604	-12.906279	10.10888	1.234351e-242
ENSG00000166165	-8.648658	10.13930	5.232947e-230
ENSG00000100721	14.399178	11.53329	7.945829e-230

	FDR
ENSG00000133124	0.000000e+00
ENSG00000105369	0.000000e+00
ENSG00000135144	0.000000e+00
ENSG00000111348	1.622267e-312
ENSG00000177606	2.899739e-296
ENSG00000012124	2.397160e-283
ENSG00000118308	3.029177e-269
ENSG00000046604	7.706978e-240
ENSG00000166165	2.904286e-227
ENSG00000100721	3.968942e-227

5.2 S3 class *DGELRT*

We follow the same steps as the previous example, where the `estimateGLMCommonDisp`, `estimateGLMTrendedDisp`, and `estimateGLMTagwiseDisp` functions are now used to obtain per-gene dispersion parameter estimates, 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 <- estimateGLMCommonDisp(dge, design)
> dge <- estimateGLMTrendedDisp(dge, design)
> dge <- estimateGLMTagwiseDisp(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
ENSG00000133124 -14.394460 11.551268 1571.967 0.000000e+00
ENSG00000105369  11.925768 11.254350 1712.406 0.000000e+00
ENSG00000135144  10.921893 11.081746 1486.253 0.000000e+00
ENSG00000111348  13.797195 10.947753 1463.959 2.656591e-320
ENSG00000012124  10.171676 10.331336 1335.896 1.788742e-292
ENSG00000118308  13.494137 10.644652 1303.113 2.381268e-285
ENSG00000177606  -9.731883 10.788439 1302.851 2.713889e-285
ENSG00000046604 -12.906279 10.071821 1120.002 1.484717e-245
ENSG00000100721  14.399175 11.550154 1084.526 7.623798e-238
ENSG00000213402  10.364993  9.692685 1064.316 1.882215e-233
              FDR
ENSG00000133124 0.000000e+00
ENSG00000105369 0.000000e+00
ENSG00000135144 0.000000e+00
ENSG00000111348 3.317418e-317
ENSG00000012124 1.786953e-289
ENSG00000118308 1.936554e-282
ENSG00000177606 1.936554e-282
ENSG00000046604 9.270203e-243
ENSG00000100721 4.231208e-235
ENSG00000213402 9.401664e-231
```

6 *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 implements 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.

```
> 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)

[1] "DESeqDataSet"
attr("package")
[1] "DESeq2"

> dim(filter)

[1] 5143    4

> res <- results(filter, independentFiltering=FALSE)
> head(res)
```

log2 fold change (MAP): cell.line Ramos B cell vs HEK293T

Wald test p-value: cell.line Ramos B cell vs HEK293T

DataFrame with 6 rows and 6 columns

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
ENSG000000000003	15.490806	-6.2823663	1.5065344	-4.1700782	3.044951e-05	0.0001448676
ENSG000000000419	15.414261	0.9205672	0.6815883	1.3506206	1.768170e-01	0.2773314581
ENSG000000000457	18.491907	-0.8627927	0.6236037	-1.3835594	1.664934e-01	0.2645275087
ENSG000000000460	9.704518	0.2755672	0.8156650	0.3378436	7.354810e-01	0.8185702590
ENSG00000001036	7.847756	-5.4044899	1.5915665	-3.3957047	6.845218e-04	0.0022830710
ENSG00000001167	77.332720	-0.3702755	0.3163472	-1.1704717	2.418112e-01	0.3531417082

```
>
```

As may be seen above, 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.

7 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(exprs(ses.norm), conds, s.len=25, norm="none",
+   plot=FALSE)
> head(filter$on)
> table(filter$on)
```

8 Session Info

```
> sessionInfo()
```

```
R version 3.1.1 Patched (2014-09-25 r66681)
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
locale:
```

```
[1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8      LC_COLLATE=C
[5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8     LC_NAME=C
```

```
[9] LC_ADDRESS=C LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

attached base packages:

```
[1] splines stats4 parallel stats graphics
[6] grDevices utils datasets methods base
```

other attached packages:

```
[1] DESeq2_1.6.0 RcppArmadillo_0.4.450.1.0
[3] Rcpp_0.11.3 GenomicRanges_1.18.0
[5] GenomeInfoDb_1.2.0 IRanges_2.0.0
[7] S4Vectors_0.4.0 edgeR_3.8.0
[9] limma_3.22.0 DESeq_1.18.0
[11] lattice_0.20-29 locfit_1.5-9.1
[13] HTSFilter_1.6.0 Biobase_2.26.0
[15] BiocGenerics_0.12.0
```

loaded via a namespace (and not attached):

```
[1] AnnotationDbi_1.28.0 BBmisc_1.7
[3] BatchJobs_1.4 BiocParallel_1.0.0
[5] DBI_0.3.1 Formula_1.1-2
[7] Hmisc_3.14-5 MASS_7.3-35
[9] RColorBrewer_1.0-5 RSQLite_0.11.4
[11] XML_3.98-1.1 XVector_0.6.0
[13] acepack_1.3-3.3 annotate_1.44.0
[15] base64enc_0.1-2 brew_1.0-6
[17] checkmate_1.4 cluster_1.15.3
[19] codetools_0.2-9 colorspace_1.2-4
[21] digest_0.6.4 fail_1.2
[23] foreach_1.4.2 foreign_0.8-61
[25] genefilter_1.48.0 geneplotter_1.44.0
[27] ggplot2_1.0.0 grid_3.1.1
[29] gtable_0.1.2 iterators_1.0.7
[31] latticeExtra_0.6-26 munsell_0.4.2
[33] nnet_7.3-8 plyr_1.8.1
[35] proto_0.3-10 reshape2_1.4
[37] rpart_4.1-8 scales_0.2.4
[39] sendmailR_1.2-1 stringr_0.6.2
[41] survival_2.37-7 tools_3.1.1
[43] xtable_1.7-4
```

References

- [1] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(R106):1–28, 2010.
- [2] Richard Bourgon, Robert Gentleman, and Wolfgang Huber. Independent filtering increases detection power for high-throughput experiments. *PNAS*, 107(21):9546–9551, 2010.
- [3] Marie-Agnès Dillies, Andrea Rau, Julie Aubert, Christelle Hennequet-Antier, Marine Jeanmougin, Nicolas Servant, Céline Keime, Guillemette Marot, David Castel, Jordi Estelle, Gregory Guerneq, 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.
- [4] 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.
- [5] R. Gentleman, V. Carey, W. Huber, and F. Hahne. *genefilter: methods for filtering genes from microarray experiments*. R package version 1.38.0.
- [6] 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.
- [7] 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.
- [8] Davide Risso, Katja Schwartz, Gavin Sherlock, and Sandrine Dudoit. GC-content normalization for RNA-seq data. *BMC Bioinformatics*, 12(480), 2011.
- [9] Mark D. Robinson and Alicia Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, 11(R25), 2010.
- [10] 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.