HTSFilter: Independent data-based filtering for replicated high-throughput sequencing experiments

Andrea Rau, Mélina Gallopin, Gilles Celeux, Florence Jaffrézic

Modified: November 7, 2012. Compiled: November 7, 2012

Abstract

This vignette illustrates the use of the *HTSFilter* package to filter replicated data from high-throughput sequencing experiments (e.g., RNA sequencing data) for three different data classes: matrix, *CountDataSet* (the S4 class associated with the *DESeq* package), and *DGEList* (the S3 class associated with the *edgeR* package).

Contents

1 Introduction

High-throughput sequencing (HTS) data, such as RNA-sequencing (RNA-seq) data, are increasingly used to conduct differential analyses, in which gene-by-gene statistical tests are performed in order to identify genes whose expression levels show systematic covariation with a particular condition, such as a treatment or phenotype of interest. 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 independent filtering procedure based on the calculation of a similarity index among biological replicates for read counts arising from replicated high-throughput sequencing data; see [7] for additional details. This technique provides an intuitive data-driven way to filter high-throughput 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 three fundamental assumptions of the filter implemented in the *HTSFilter* package are as follows:

1. Biological replicates are present for each experimental condition,

2. Data can be appropriately normalized (scaled) to correct for systematic inter-sample biases, and

3. Within a given condition, after controlling for inter-sample biases, each gene exhibits similar expression levels among replicates.

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 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 (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), and a *DGEList* (the S3 class associated with the *edgeR* package), as illustrated in the following sections.

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. 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* and *Biobase* packages, and attach the gene-level count data contained in `sultan`:

```
> library(HTSFilter)
> library(Biobase)
> data("sultan")
> 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.

3 Matrix and data.frame classs

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. To use the DESeq normalization method [1], `normalization="DESeq"` may be specified.

```
> mat <- exprs(sultan)
> conds <- pData(sultan)$cell.line
> filter <- HTSFilter(mat, conds, s.len=25)
> mat <- filter$filteredData
> dim(mat)

[1] 4995    4
```
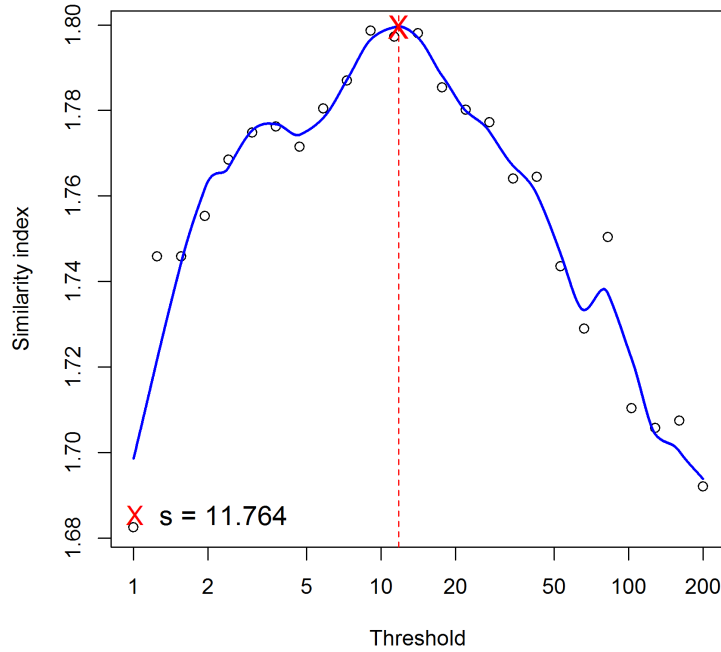
Figure 1: Global Jaccard index for the `sultan` data calculated for a variety of threshold values after TMM normalization [9], with a loess curve (blue line) superposed and data-based threshold values (red cross and red dotted line) equal to 11.764.

```
> dim(filter$removedData)
```

```
[1] 4015     4
```

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 maximum is reached for the global similarity index over the range of tested threshold values (see Figure 1). 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.

4 S4 class *CountDataSet* (*DESeq* package)

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
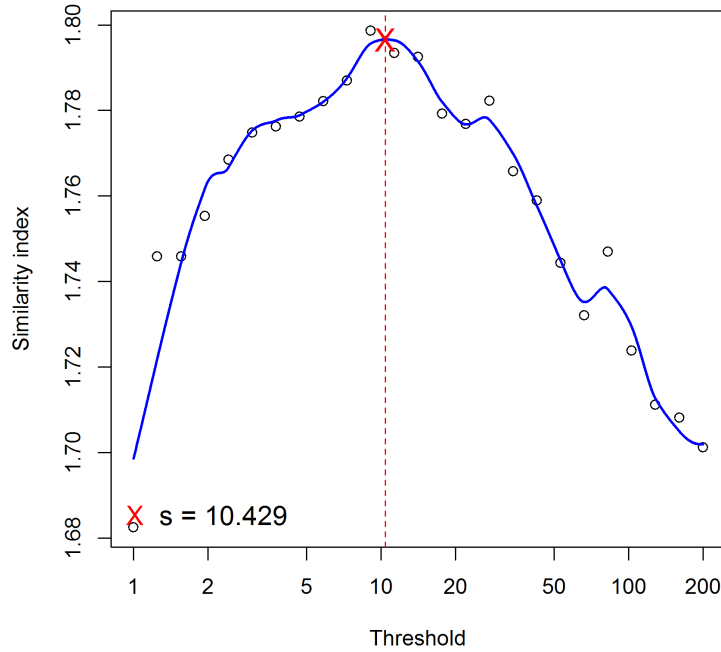
4

Figure 2: 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.

be only 25 (`s.len=25`) rather than the default value of 100 to reduce computation time for this example. In addition, we specify that the normalization strategy to be used is `normalization="DESeq"`.

```
> library(DESeq)
> cds <- newCountDataSet(exprs(sultan), conds)
> filter <- HTSFilter(counts(cds), conds, s.len=25, normalization = "DESeq")
> cds <- newCountDataSet(filter$filteredData, conds)
> dim(cds)
> dim(filter$removedData)
> class(cds)
```

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 2). 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.

Typically, the next step would be to perform a differential analysis using the *DESeq* pipeline, namely by estimating the size factors of the filtered data (`estimateSizeFactors`), estimating the per-gene and fitted dispersions (`estimateDispersions`), and performing an exact test (`nbinomTest`) to identify differentially expressed genes.

```
> ## Not run:
> ## cds <- estimateSizeFactors(cds)
> ## cds <- estimateDispersions(cds)
> ## res <- nbinomTest(cds, "treated", "untreated")
```

5 S3 class *DGEList* (*edgeR* package)

To filter high-throughput sequencing data in the form of a *DGEList* (the class used within the *edgeR* pipeline for differential analysis), 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. We then 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]. We suppress the plot of the global Jaccard index using `plot = FALSE`, as it is identical to that shown in Figure 1.

```
> dge <- DGEList(counts=exprs(sultan), group=conds)
> filter <- HTSFilter(dge, s.len=25, plot=FALSE)
> dge <- filter$filteredData
> dim(dge)

[1] 4995    4

> dim(filter$removedData)

[1] 4015    4

> class(dge)

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

Note that the filtered data are of the same class as the original data, *DGEList*, and that library sizes have been re-estimated after filtering the data using the TMM method. Typically, the next step would be to perform a differential analysis using the *edgeR* pipeline, namely by estimating common (`estimateCommonDisp`) and per-gene dispersions (`estimateTagwiseDisp`), and performing an exact test (`exactTest`) to identify differentially expressed genes.

```
> ## Not run:
> ## dge <- estimateCommonDisp(dge)
> ## dge <- estimateTagwiseDisp(dge)
> ## res <- exactTest(dge)
```

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 `normalization="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, normalization="none",
+            plot=FALSE)
> head(filter$on)
> ses <- newSeqExpressionSet(exprs(sultan)[which(filter$on == 1),],
+        phenoData=phenoData(sultan))
```

7 Session Info

```
> sessionInfo()

R version 2.15.1 (2012-06-22)
Platform: x86_64-pc-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=English_United States.1252
[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] stats      graphics  grDevices utils     datasets
[6] methods    base

other attached packages:
[1] Biobase_2.16.0          BiocGenerics_0.2.0
[3] HTSFilter_0.1.1         edgeR_2.6.12
[5] limma_3.12.3            SweaveListingUtils_0.5.5
[7] startupmsg_0.7.2

loaded via a namespace (and not attached):
[1] tools_2.15.1
```

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 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.

[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. Independent data-based filtering for replicated high-throughput sequencing experiments. *(submitted)*, 2012.

[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.