

Package ‘HTSFilter’

May 7, 2025

Type Package

Title Filter replicated high-throughput transcriptome sequencing data

Version 1.48.0

Date 2020-12-04

Depends R (>= 4.0.0)

Imports edgeR, DESeq2, BiocParallel, Biobase, utils, stats, grDevices,
graphics, methods

Suggests EDASeq, testthat, knitr, rmarkdown, BiocStyle

Description This package implements a filtering procedure for
replicated transcriptome sequencing data based on a global
Jaccard similarity index in order to identify genes with low,
constant levels of expression across one or more experimental
conditions.

License Artistic-2.0

LazyLoad yes

biocViews Sequencing, RNASeq, Preprocessing, DifferentialExpression,
GeneExpression, Normalization, ImmunoOncology

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

git_url <https://git.bioconductor.org/packages/HTSFilter>

git_branch RELEASE_3_21

git_last_commit 9bcd700

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-05-07

Author Andrea Rau [cre, aut] (ORCID: <<https://orcid.org/0000-0001-6469-488X>>),
Melina Gallopin [ctb],
Gilles Celeux [ctb],
Florence Jaffrézic [ctb]

Maintainer Andrea Rau <andrea.rau@inrae.fr>

Contents

HTSFilter-package	2
HTSBasicFilter	3
HTSFilter	7
normalizeData	12
sultan	13
Index	15

HTSFilter-package	<i>Filter replicated high-throughput transcriptome sequencing data</i>
-------------------	--

Description

This package implements a filtering procedure for replicated transcriptome sequencing data based on a global Jaccard similarity index in order to identify genes with low, constant levels of expression across one or more experimental conditions.

Details

Package:	HTSFilter
Type:	Package
Version:	1.31.1
Date:	2020-11-26
License:	Artistic-2.0
LazyLoad:	yes

Author(s)

Andrea Rau, Melina Gallopin, Gilles Celeux, and Florence Jaffrezic
Maintainer: Andrea Rau <andrea.rau@inrae.fr>

References

R. Bourgon, R. Gentleman, and W. Huber. (2010) Independent filtering increases detection power for high- throughput experiments. *PNAS* **107**(21):9546-9551.

P. Jaccard (1901). Etude comparative de la distribution orale dans une portion des Alpes et des Jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, **37**:547-549.

A. Rau, M. Gallopin, G. Celeux, F. Jaffrezic (2013). Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics*, doi: 10.1093/bioinformatics/btt350.

Examples

```

library(Biobase)
data("sultan")
conds <- pData(sultan)$cell.line

#####
## Matrix or data.frame
#####

filter <- HTSFilter(exprs(sultan), conds, s.len=25, plot=FALSE)

#####
## DGEEExact
#####

library(edgeR)
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
## topTags(et)

#####
## DESeq2
#####

library(DESeq2)
conds <- gsub(" ", ".", conds)
dds <- DESeqDataSetFromMatrix(countData = exprs(sultan),
                             colData = data.frame(cell.line = conds),
                             design = ~ cell.line)

## Not run:
##
## dds <- DESeq(dds)
## filter <- HTSFilter(dds, s.len=25, plot=FALSE)$filteredData
## class(filter)
## res <- results(filter, independentFiltering=FALSE)

```

Description

Implement a variety of basic filters for transcriptome sequencing data.

Usage

```
HTSBasicFilter(x, ...)

## S4 method for signature 'matrix'
HTSBasicFilter(
  x,
  method,
  cutoff.type = "value",
  cutoff = 10,
  length = NA,
  normalization = c("TMM", "DESeq", "none")
)

## S4 method for signature 'data.frame'
HTSBasicFilter(
  x,
  method,
  cutoff.type = "value",
  cutoff = 10,
  length = NA,
  normalization = c("TMM", "DESeq", "none")
)

## S4 method for signature 'DGEList'
HTSBasicFilter(
  x,
  method,
  cutoff.type = "value",
  cutoff = 10,
  length = NA,
  normalization = c("TMM", "DESeq", "pseudo.counts", "none")
)

## S4 method for signature 'DGEEExact'
HTSBasicFilter(
  x,
  method,
  cutoff.type = "value",
  cutoff = 10,
  length = NA,
  normalization = c("TMM", "DESeq", "pseudo.counts", "none")
)

## S4 method for signature 'DGEGLM'
HTSBasicFilter(
  x,
  method,
  cutoff.type = "value",
```

```

    cutoff = 10,
    length = NA,
    normalization = c("TMM", "DESeq", "none")
)

## S4 method for signature 'DGELRT'
HTSBasicFilter(
  x,
  method,
  cutoff.type = "value",
  cutoff = 10,
  length = NA,
  normalization = c("TMM", "DESeq", "none")
)

## S4 method for signature 'DESeqDataSet'
HTSBasicFilter(
  x,
  method,
  cutoff.type = "value",
  cutoff = 10,
  length = NA,
  normalization = c("DESeq", "TMM", "none"),
  pAdjustMethod = "BH"
)

```

Arguments

<code>x</code>	A numeric matrix or data.frame representing the counts of dimension ($g \times n$), for g genes in n samples, a <code>DGEList</code> object, a <code>DGEEExact</code> object, a <code>DGEGLM</code> object, a <code>DGELRT</code> object, or a <code>DESeqDataSet</code> object.
<code>...</code>	Additional optional arguments
<code>method</code>	Basic filtering method to be used: "mean", "sum", "rpkm", "variance", "cpm", "max", "cpm.mean", "cpm.sum", "cpm.variance", "cpm.max", "rpkm.mean", "rpkm.sum", "rpkm.variance", or "rpkm.max"
<code>cutoff.type</code>	Type of cutoff to be used: a numeric value indicating the number of samples to be used for filtering (when <code>method = "cpm"</code> or <code>"rpkm"</code>), or one of "value", "number", or "quantile"
<code>cutoff</code>	Cutoff to be used for chosen filter
<code>length</code>	Optional vector of length n containing the lengths of each gene in x ; optional except in the case of <code>method = "rpkm"</code>
<code>normalization</code>	Normalization method to be used to correct for differences in library sizes, with choices "TMM" (Trimmed Mean of M-values), "DESeq" (normalization method proposed in the DESeq package), "pseudo.counts" (pseudo-counts obtained via quantile-quantile normalization in the edgeR package, only available for objects of class <code>DGEList</code> and <code>DGEEExact</code>), and "none" (to be used only if user is certain no normalization is required, or if data have already been pre-normalized by an alternative method)

`pAdjustMethod` The method used to adjust p-values, see `?p.adjust`

Details

This function implements a basic filter for high-throughput sequencing data for a variety of filter types: mean, sum, RPKM, variance, CPM, maximum, mean CPM values, the sum of CPM values, the variance of CPM values, maximum CPM value, mean RPKM values, the sum of RPKM values, the variance of RPKM values, or the maximum RPKM value. The filtering criteria used may be for a given cutoff value, a number of genes, or a given quantile value.

Value

- `filteredData` An object of the same class as `x` containing the data that passed the filter
- `on` A binary vector of length `g`, where 1 indicates a gene with normalized expression greater than the optimal filtering threshold `s.optimal` in at least one sample (irrespective of condition labels), and 0 indicates a gene with normalized expression less than or equal to the optimal filtering threshold in all samples
- `normFactor` A vector of length `n` giving the estimated library sizes estimated by the normalization method specified in `normalization`
- `removedData` A matrix containing the filtered data
- `filterCrit` A vector or matrix containing the criteria used to perform filtering

Author(s)

Andrea Rau, Melina Gallopin, Gilles Celeux, and Florence Jaffrezic

References

- R. Bourgon, R. Gentleman, and W. Huber. (2010) Independent filtering increases detection power for high-throughput experiments. *PNAS* **107**(21):9546-9551.
- A. Rau, M. Gallopin, G. Celeux, F. Jaffrezic (2013). Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics*, doi: 10.1093/bioinformatics/btt350.

Examples

```
library(Biobase)
data("sultan")
conds <- pData(sultan)$cell.line

#####
## Matrix or data.frame
#####

## Filter genes with total (sum) normalized gene counts < 10
filter <- HTSBasicFilter(exprs(sultan), method="sum", cutoff.type="value",
                        cutoff = 10)

#####
```

```
## DGEXact
#####

library(edgeR)
## Filter genes with CPM values less than 100 in more than 2 samples
dge <- DGEList(counts=exprs(sultan), group=conds)
dge <- calcNormFactors(dge)
filter <- HTSBasicFilter(dge, method="cpm", cutoff.type=2, cutoff=100)

#####
## DESeq2
#####

library(DESeq2)
conds <- gsub(" ", ".", conds)
dds <- DESeqDataSetFromMatrix(countData = exprs(sultan),
                              colData = data.frame(cell.line = conds),
                              design = ~ cell.line)

## Not run: Filter genes with mean normalized gene counts < 40% quantile
## dds <- DESeq(dds)
## filter <- HTSBasicFilter(dds, method="mean", cutoff.type="quantile",
## cutoff = 0.4)
## res <- results(filter, independentFiltering=FALSE)
```

HTSFilter

Calculate data-based filtering threshold for replicated transcriptome sequencing data.

Description

Calculate a data-based filtering threshold for replicated transcriptome sequencing data through the pairwise Jaccard similarity index between pairs of replicates within each experimental condition.

Usage

```
HTSFilter(x, ...)

## S4 method for signature 'matrix'
HTSFilter(
  x,
  conds,
  s.min = 1,
  s.max = 200,
  s.len = 100,
  loess.span = 0.3,
  normalization = c("TMM", "DESeq", "none"),
  plot = TRUE,
```

```
    plot.name = NA,
    parallel = FALSE,
    BPPARAM = bpparam()
)

## S4 method for signature 'data.frame'
HTSFilter(
  x,
  conds,
  s.min = 1,
  s.max = 200,
  s.len = 100,
  loess.span = 0.3,
  normalization = c("TMM", "DESeq", "none"),
  plot = TRUE,
  plot.name = NA,
  parallel = FALSE,
  BPPARAM = bpparam()
)

## S4 method for signature 'DGEList'
HTSFilter(
  x,
  s.min = 1,
  s.max = 200,
  s.len = 100,
  loess.span = 0.3,
  normalization = c("TMM", "DESeq", "pseudo.counts", "none"),
  plot = TRUE,
  plot.name = NA,
  parallel = FALSE,
  BPPARAM = bpparam(),
  conds
)

## S4 method for signature 'DGEEExact'
HTSFilter(
  x,
  DGEList,
  s.min = 1,
  s.max = 200,
  s.len = 100,
  loess.span = 0.3,
  normalization = c("TMM", "DESeq", "pseudo.counts", "none"),
  plot = TRUE,
  plot.name = NA,
  parallel = FALSE,
  BPPARAM = bpparam(),
```

```
    conds
  )

## S4 method for signature 'DGEGLM'
HTSFilter(
  x,
  s.min = 1,
  s.max = 200,
  s.len = 100,
  loess.span = 0.3,
  normalization = c("TMM", "DESeq", "none"),
  plot = TRUE,
  plot.name = NA,
  parallel = FALSE,
  BPPARAM = bpparam(),
  conds
)

## S4 method for signature 'DGEGLRT'
HTSFilter(
  x,
  DGEGLM,
  s.min = 1,
  s.max = 200,
  s.len = 100,
  loess.span = 0.3,
  normalization = c("TMM", "DESeq", "none"),
  plot = TRUE,
  plot.name = NA,
  parallel = FALSE,
  BPPARAM = bpparam(),
  conds
)

## S4 method for signature 'DESeqDataSet'
HTSFilter(
  x,
  s.min = 1,
  s.max = 200,
  s.len = 100,
  loess.span = 0.3,
  normalization = c("DESeq", "TMM", "none"),
  plot = TRUE,
  plot.name = NA,
  pAdjustMethod = "BH",
  parallel = FALSE,
  BPPARAM = bpparam(),
  conds
)
```

)

Arguments

<code>x</code>	A numeric matrix or data.frame representing the counts of dimension ($g \times n$), for g genes in n samples, a <code>DGEList</code> object, a <code>DGEEExact</code> object, a <code>DGEGLM</code> object, a <code>DGELRT</code> object, or a <code>DESeqDataSet</code> object.
<code>...</code>	Additional optional arguments
<code>conds</code>	Vector of length n identifying the experimental condition of each of the n samples; required when <code>sQuote(x)</code> is a numeric matrix. In the case of objects of class <code>DGEList</code> , <code>DGEEExact</code> , <code>DGEGLM</code> , <code>DGELRT</code> , or <code>DESeqDataSet</code> , the design matrix is automatically
<code>s.min</code>	Minimum value of filtering threshold to be considered, with default value equal to 1
<code>s.max</code>	Maximum value of filtering threshold to be considered, with default value equal to 200
<code>s.len</code>	Length of sequence of filtering thresholds to be considered (from <code>s.min</code> to <code>s.max</code>) for the calculation of the global similarity index
<code>loess.span</code>	Span of the loess curve to be fitted to the filtering thresholds and corresponding global similarity indices, with default value equal to 0.3
<code>normalization</code>	Normalization method to be used to correct for differences in library sizes, with choices “TMM” (Trimmed Mean of M-values), “DESeq” (normalization method proposed in the DESeq package), “pseudo.counts” (pseudo-counts obtained via quantile-quantile normalization in the edgeR package, only available for objects of class <code>DGEList</code> and <code>DGEEExact</code>), and “none” (to be used only if user is certain no normalization is required, or if data have already been pre-normalized by an alternative method)
<code>plot</code>	If “TRUE”, produce a plot of the calculated global similarity indices against the filtering threshold with superimposed loess curve
<code>plot.name</code>	If <code>plot = “TRUE”</code> , the name of the PDF file to be saved to the current working directory. If <code>plot.name = NA</code> , the plot is drawn in the current window.
<code>parallel</code>	If FALSE, no parallelization. If TRUE, parallel execution using <code>BiocParallel</code> (see next argument <code>BPPARAM</code>). A note on running in parallel using <code>BiocParallel</code> : it may be advantageous to remove large, unneeded objects from the current R environment before calling the function, as it is possible that R’s internal garbage collection will copy these files while running on worker nodes.
<code>BPPARAM</code>	Optional parameter object passed internally to <code>bplapply</code> when <code>parallel=TRUE</code> . If not specified, the parameters last registered with <code>register</code> will be used.
<code>DGEList</code>	Object of class <code>DGEList</code> , to be used when filtering objects of class <code>DGEEExact</code>
<code>DGEGLM</code>	Object of class <code>DGEGLM</code> , to be used when filtering objects of class <code>DGELRT</code>
<code>pAdjustMethod</code>	The method used to adjust p-values, see <code>?p.adjust</code>

Details

The Jaccard similarity index, which measures the overlap of two sets, is calculated as follows. Given two binary vectors, each of length n , we define the following values:

- a = the number of attributes with a value of 1 in both vectors
- b = the number of attributes with a value of 1 in the first vector and 0 in the second
- c = the number of attributes with a value of 0 in the first vector and 1 in the second
- d = the number of attributes with a value of 0 in both vectors

We note that all attributes fall into one of these four quantities, so $a + b + c + d = n$. Given these quantities, we may calculate the Jaccard similarity index between the two vectors as follows:

$$J = \frac{a}{a + b + c}.$$

Value

- `filteredData` An object of the same class as `x` containing the data that passed the filter
- `on` A binary vector of length g , where 1 indicates a gene with normalized expression greater than the optimal filtering threshold `s.optimal` in at least one sample (irrespective of condition labels), and 0 indicates a gene with normalized expression less than or equal to the optimal filtering threshold in all samples
- `s` The optimal filtering threshold as identified by the global similarity index
- `indexValues` A matrix of dimension $(s.len \times 2)$ giving the tested filtering thresholds and the corresponding global similarity indices. Note that the threshold values are equally spaced on the *log* scale, and thus unequally spaced on the count scale (i.e., we test more threshold values at very low levels of expression, and fewer at very high levels of expression).
- `normFactor` A vector of length n giving the estimated library sizes estimated by the normalization method specified in `normalization`
- `removedData` A matrix containing the filtered data

Author(s)

Andrea Rau, Melina Gallopin, Gilles Celeux, and Florence Jaffrezic

References

- R. Bourgon, R. Gentleman, and W. Huber. (2010) Independent filtering increases detection power for high-throughput experiments. *PNAS* **107**(21):9546-9551.
- P. Jaccard (1901). Etude comparative de la distribution orale dans une portion des Alpes et des Jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, **37**:547-549.
- A. Rau, M. Gallopin, G. Celeux, F. Jaffrezic (2013). Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics*, doi: 10.1093/bioinformatics/btt350.

Examples

```

library(Biobase)
data("sultan")
conds <- pData(sultan)$cell.line

#####
## Matrix or data.frame
#####

filter <- HTSFilter(exprs(sultan), conds, s.len=25, plot=FALSE)

#####
## DGEEExact
#####

library(edgeR)
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
## topTags(et)

#####
## DESeq2
#####

library(DESeq2)
conds <- gsub(" ", ".", conds)
dds <- DESeqDataSetFromMatrix(countData = exprs(sultan),
                             colData = data.frame(cell.line = conds),
                             design = ~ cell.line)

## Not run:
##
## dds <- DESeq(dds)
## filter <- HTSFilter(dds, s.len=25, plot=FALSE)$filteredData
## class(filter)
## res <- results(filter, independentFiltering=FALSE)

```

normalizeData

Normalize transcriptome sequencing data.

Description

Normalize count-based measures of transcriptome sequencing data using the Trimmed Means of M-values (TMM) or DESeq approach.

Usage

```
normalizeData(data, normalization)
```

Arguments

data	numeric matrix representing the counts of dimension $(g \times n)$, for g genes in n samples.
normalization	Normalization method to be used to correct for differences in library sizes, with choices “TMM” (Trimmed Mean of M-values), “DESeq” (normalization method proposed in the DESeq package), and “none”

Value

- data.norm A numeric matrix representing the normalized counts of dimension $(g \times n)$, for g genes in n samples.
- norm.factor A vector of length n giving the estimated library sizes estimated by the normalization method specified in normalization

Author(s)

Andrea Rau, Melina Gallopin, Gilles Celeux, and Florence Jaffrezic

References

S. Anders and W. Huber (2010). Differential expression analysis for sequence count data. *Genome Biology*, 11(R106):1-28.

A. Rau, M. Gallopin, G. Celeux, F. Jaffrezic (2013). Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics*, doi: 10.1093/bioinformatics/btt350.

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

Examples

```
library(Biobase)
data("sultan")
normData <- normalizeData(exprs(sultan), norm="DESeq")
```

sultan

RNA-seq data from humans in Sultan et al. (2008)

Description

This dataset represents RNA-seq data from humans in two conditions (Ramos B cell line and HEK293T), with two biological replicates per condition. The ExpressionSet was downloaded from the ReCount online resource.

Usage

```
data(sultan)
```

Format

An ExpressionSet named `sultan.eset` containing the phenotype data and expression data for the Sultan et al. (2008) experiment. Phenotype data may be accessed using the `pData` function, and expression data may be accessed using the `exprs` function.

Value

Object of class 'ExpressionSet'. Matrix of counts can be accessed after loading the 'Biobase' package and calling `exprs(sultan)`.

Source

ReCount online resource (<http://bowtie-bio.sourceforge.net/recount>).

References

data_blah.com

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.

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'Keefe, 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.

Index

- * **datasets**
 - sultan, [13](#)
- * **methods**
 - HTSBasicFilter, [3](#)
 - HTSFilter, [7](#)
 - normalizeData, [12](#)
- * **package**
 - HTSFilter-package, [2](#)

HTSBasicFilter, [3](#)

HTSBasicFilter, data.frame-method
(HTSBasicFilter), [3](#)

HTSBasicFilter, DESeqDataSet-method
(HTSBasicFilter), [3](#)

HTSBasicFilter, DGEXact-method
(HTSBasicFilter), [3](#)

HTSBasicFilter, DGEGLM-method
(HTSBasicFilter), [3](#)

HTSBasicFilter, DGEList-method
(HTSBasicFilter), [3](#)

HTSBasicFilter, DGELRT-method
(HTSBasicFilter), [3](#)

HTSBasicFilter, matrix-method
(HTSBasicFilter), [3](#)

HTSFilter, [7](#)

HTSFilter, data.frame-method
(HTSFilter), [7](#)

HTSFilter, DESeqDataSet-method
(HTSFilter), [7](#)

HTSFilter, DGEXact-method (HTSFilter), [7](#)

HTSFilter, DGEGLM-method (HTSFilter), [7](#)

HTSFilter, DGEList-method (HTSFilter), [7](#)

HTSFilter, DGELRT-method (HTSFilter), [7](#)

HTSFilter, matrix-method (HTSFilter), [7](#)

HTSFilter-methods (HTSFilter), [7](#)

HTSFilter-package, [2](#)

normalizeData, [12](#)

sultan, [13](#)