

# Package ‘EDASeq’

December 15, 2024

**Version** 2.41.0

**Title** Exploratory Data Analysis and Normalization for RNA-Seq

**Description** Numerical and graphical summaries of RNA-Seq read data.

Within-lane normalization procedures to adjust for GC-content effect (or other gene-level effects) on read counts: loess robust local regression, global-scaling, and full-quantile normalization (Risso et al., 2011). Between-lane normalization procedures to adjust for distributional differences between lanes (e.g., sequencing depth): global-scaling and full-quantile normalization (Bullard et al., 2010).

**Author** Davide Risso [aut, cre, cph], Sandrine Dudoit [aut], Ludwig Geistlinger [ctb]

**Maintainer** Davide Risso <risso.davide@gmail.com>

**Date** 08-30-2011

**Depends** Biobase (>= 2.15.1), ShortRead (>= 1.11.42)

**Imports** methods, graphics, BiocGenerics, IRanges (>= 1.13.9), aroma.light, Rsamtools (>= 1.5.75), biomaRt, Biostrings, AnnotationDbi, GenomicFeatures, GenomicRanges, BiocManager

**Suggests** BiocStyle, knitr, yeastRNASeq, leeBamViews, edgeR, KernSmooth, testthat, DESeq2, rmarkdown

**VignetteBuilder** knitr

**License** Artistic-2.0

**LazyLoad** yes

**biocViews** ImmunoOncology, Sequencing, RNASeq, Preprocessing, QualityControl, DifferentialExpression

**URL** <https://github.com/drisso/EDASeq>

**BugReports** <https://github.com/drisso/EDASeq/issues>

**RoxygenNote** 7.1.0

**git\_url** <https://git.bioconductor.org/packages/EDASeq>

**git\_branch** devel

**git\_last\_commit** 68d2eba

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-15

## Contents

EDASeq-package . . . . .	2
barplot-methods . . . . .	3
betweenLaneNormalization-methods . . . . .	3
biasBoxplot-methods . . . . .	5
biasPlot-methods . . . . .	6
boxplot-methods . . . . .	7
getGeneLengthAndGCCContent . . . . .	7
MDPlot-methods . . . . .	8
meanVarPlot-methods . . . . .	9
newSeqExpressionSet . . . . .	9
plot-methods . . . . .	10
plotNtFrequency-methods . . . . .	11
plotPCA-methods . . . . .	11
plotQuality-methods . . . . .	12
plotRLE-methods . . . . .	13
SeqExpressionSet-class . . . . .	14
withinLaneNormalization-methods . . . . .	16
yeastGC . . . . .	17
yeastLength . . . . .	18

## Index 19

---

EDASeq-package	<i>Exploratory Data Analysis and Normalization for RNA-Seq data</i>
----------------	---

---

## Description

Numerical summaries and graphical representations of some key features of the data along with implementations of both within-lane normalization methods for GC content bias and between-lane normalization methods to adjust for sequencing depth and possibly other differences in distribution.

## Details

The `SeqExpressionSet` class is used to store gene-level counts along with sample information. It extends the virtual class `eSet`. See the help page of the class for details.

"Read-level" information is managed via the `FastqFileList` and `BamFileList` classes of `Rsamtools`.

Most used graphic tools for the `FastqFileList` and `BamFileList` objects are: `'barplot'`, `'plotQuality'`, `'plotNtFrequency'`. For `SeqExpressionSet` objects are: `'biasPlot'`, `'meanVarPlot'`, `'MDPlot'`.

To perform gene-level normalization use the functions `'withinLaneNormalization'` and `'betweenLaneNormalization'`.

See the package vignette for a typical Exploratory Data Analysis example.

**Author(s)**

Davide Risso and Sandrine Dudoit. Maintainer: Davide Risso <risso.davide@gmail.com>

**References**

J. H. Bullard, E. A. Purdom, K. D. Hansen and S. Dudoit (2010). Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. BMC Bioinformatics Vol. 11, Article 94.

D. Risso, K. Schwartz, G. Sherlock and S. Dudoit (2011). GC-Content Normalization for RNA-Seq Data. Technical Report No. 291, Division of Biostatistics, University of California, Berkeley, Berkeley, CA.

---

barplot-methods                      *Methods for Function* barplot *in Package* **EDASeq**

---

**Description**

High-level functions to produce barplots of some complex objects.

**Methods**

signature(height = "BamFile") Usage: barplot(height,strata=c("rname","strand")) It produces a barplot of the total number of reads in each chromosome (if "rname") or strand.

signature(height = "BamFileList") It produces a barplot of the total number of reads in each object in height. If unique=TRUE is specified, it stratified the total by uniquely/non-uniquely mapped reads.

signature(height = "FastqFileList") It produces a barplot of the total number of reads in each object in height.

---

betweenLaneNormalization-methods  
*Methods for Function* betweenLaneNormalization *in Package*  
**EDASeq**

---

**Description**

Between-lane normalization for sequencing depth and possibly other distributional differences between lanes.

**Usage**

betweenLaneNormalization(x, which=c("median","upper","full"), offset=FALSE, round=TRUE)

**Arguments**

x	A numeric matrix representing the counts or a <a href="#">SeqExpressionSet</a> object.
which	Method used to normalized. See the details section and the reference below for details.
offset	Should the normalized value be returned as an offset leaving the original counts unchanged?
round	If TRUE the normalization returns rounded values (pseudo-counts). Ignored if offset=TRUE.

**Details**

This method implements three normalizations described in Bullard et al. (2010). The methods are:

median: a scaling normalization that forces the median of each lane to be the same.

upper: the same but with the upper quartile.

full: a non linear full quantile normalization, in the spirit of the one used in microarrays.

**Methods**

`signature(x = "matrix")` It returns a matrix with the normalized counts if `offset=FALSE` or with the offset if `offset=TRUE`.

`signature(x = "SeqExpressionSet")` It returns a `linkS4class{SeqExpressionSet}` with the normalized counts in the `normalizedCounts` slot and with the offset in the `offset` slot (if `offset=TRUE`).

**Author(s)**

Davide Risso.

**References**

J. H. Bullard, E. A. Purdom, K. D. Hansen and S. Dudoit (2010). Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics* Vol. 11, Article 94.

D. Risso, K. Schwartz, G. Sherlock and S. Dudoit (2011). GC-Content Normalization for RNA-Seq Data. Manuscript in Preparation.

**Examples**

```
library(yeastRNASeq)
data(geneLevelData)
data(yeastGC)

sub <- intersect(rownames(geneLevelData), names(yeastGC))

mat <- as.matrix(geneLevelData[sub, ])

data <- newSeqExpressionSet(mat,
```

```

phenoData=AnnotatedDataFrame(
  data.frame(conditions=factor(c("mut", "mut", "wt", "wt")),
    row.names=colnames(geneLevelData)),
  featureData=AnnotatedDataFrame(data.frame(gc=yeastGC[sub])))

norm <- betweenLaneNormalization(data, which="full", offset=FALSE)

```

---

biasBoxplot-methods      *Methods for Function biasBoxplot in Package EDASeq*

---

### Description

biasBoxplot produces a boxplot representing the distribution of a quantity of interest (e.g. gene counts, log-fold-changes, ...) stratified by a covariate (e.g. gene length, GC-content, ...).

### Usage

```
biasBoxplot(x,y,num.bins,...)
```

### Arguments

x	A numeric vector with the quantity of interest (e.g. gene counts, log-fold-changes, ...)
y	A numeric vector with the covariate of interest (e.g. gene length, GC-content, ...)
num.bins	A numeric value specifying the number of bins in which to stratify y. Default to 10.
...	See <a href="#">par</a>

### Methods

signature(x = "numeric", y = "numeric", num.bins = "numeric") It plots a line representing the regression of every column of the matrix x on the numeric covariate y. One can pass the usual graphical parameters as additional arguments (see [par](#)).

### Examples

```

library(yeastRNASeq)
data(geneLevelData)
data(yeastGC)

sub <- intersect(rownames(geneLevelData), names(yeastGC))

mat <- as.matrix(geneLevelData[sub,])

data <- newSeqExpressionSet(mat,
  phenoData=AnnotatedDataFrame(
    data.frame(conditions=factor(c("mut", "mut", "wt", "wt")),

```

```

                                row.names=colnames(geneLevelData))),
                                featureData=AnnotatedDataFrame(data.frame(gc=yeastGC[sub])))

lfc <- log(geneLevelData[sub, 3] + 1) - log(geneLevelData[sub, 1] + 1)

biasBoxplot(lfc, yeastGC[sub], las=2, cex.axis=.7)

```

---

biasPlot-methods

*Methods for Function biasPlot in Package EDASeq*


---

## Description

biasPlot produces a plot of the [lowess](#) regression of the counts on a covariate of interest, typically the GC-content or the length of the genes.

## Methods

signature(x = "matrix", y = "numeric") It plots a line representing the regression of every column of the matrix x on the numeric covariate y. One can pass the usual graphical parameters as additional arguments (see [par](#)).

signature(x = "SeqExpressionSet", y = "character") It plots a line representing the regression of every lane in x on the covariate specified by y. y must be one of the column of the featureData slot of the x object. One can pass the usual graphical parameters as additional arguments (see [par](#)). The parameter color\_code (optional) must be a number specifying the column of phenoData to be used for color-coding. By default it is color-coded according to the first column of phenoData. If legend=TRUE and col is not specified a legend with the information stored in phenoData is added.

## Examples

```

library(yeastRNASeq)
data(geneLevelData)
data(yeastGC)

sub <- intersect(rownames(geneLevelData), names(yeastGC))

mat <- as.matrix(geneLevelData[sub,])

data <- newSeqExpressionSet(mat,
                           phenoData=AnnotatedDataFrame(
                               data.frame(conditions=factor(c("mut", "mut", "wt", "wt")),
                                             row.names=colnames(geneLevelData))),
                           featureData=AnnotatedDataFrame(data.frame(gc=yeastGC[sub])))

biasPlot(data, "gc", ylim=c(0,5), log=TRUE)

```

---

boxplot-methods	<i>Methods for Function boxplot in Package EDASeq</i>
-----------------	---

---

### Description

High-level functions to produce boxplots of some complex objects.

### Methods

`signature(x = "FastqQuality")` It plots the distribution of the quality per read position.

`signature(x = "SeqExpressionSet")` It plots the distribution of the log counts in each lane of `x`.

---

<code>getGeneLengthAndGCCContent</code>	<i>Get gene length and GC-content</i>
---	---------------------------------------

---

### Description

Automatically retrieves gene length and GC-content information from Biomart or org.db packages.

### Usage

```
getGeneLengthAndGCCContent(id, org, mode=c("biomart", "org.db"))
```

### Arguments

<code>id</code>	Character vector of one or more ENSEMBL or ENTREZ gene IDs.
<code>org</code>	Organism three letter code, e.g. 'hsa' for 'Homo sapiens'. See also: <a href="http://www.genome.jp/kegg/catalog/">http://www.genome.jp/kegg/catalog/</a> In org.db mode, this can be also a specific genome assembly, e.g. 'hg38' or 'sac-Cer3'.
<code>mode</code>	Mode to retrieve the information. Defaults to 'biomart'. See Details.

### Details

The 'biomart' mode is based on functionality from the biomaRt package and retrieves the required information from the BioMart database. This is available for all ENSEMBL organisms and is typically most current, but can be time-consuming when querying several thousand genes at a time.

The 'org.db' mode uses organism-based annotation packages from Bioconductor. This is much faster than the 'biomart' mode, but is only available for selected model organism currently supported by BioC annotation functionality.

Results for the same gene ID(s) can differ between both modes as they are based on different sources for the underlying genome assembly. While the 'biomart' mode uses the latest ENSEMBL version, the 'org.db' mode uses BioC annotation packages typically built from UCSC.

**Value**

A numeric matrix with two columns: gene length and GC-content.

**Author(s)**

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

**See Also**

[getSequence](#) to retrieve a genomic sequence from BioMart, [genes](#) to extract genomic coordinates from a TxDb object, [getSeq](#) to extract genomic sequences from a BSgenome object, [alphabetFrequency](#) to calculate nucleotide frequencies.

**Examples**

```
getGeneLengthAndGCContent("ENSG0000012048", "hsa")
```

---

MDPlot-methods

*Methods for Function MDPlot in Package EDASeq*

---

**Description**

MDPlot produces a mean-difference smooth scatterplot of two lanes in an experiment.

**Usage**

```
MDPlot(x,y,...)
```

**Arguments**

x	Either a numeric matrix or a <a href="#">SeqExpressionSet</a> object containing the gene expression.
y	A numeric vector specifying the lanes to be compared.
...	See <a href="#">par</a>

**Details**

The mean-difference (MD) plot is a useful plot to visualize difference in two lanes of an experiment. From a MDPlot one can see if normalization is needed and if a linear scaling is sufficient or nonlinear normalization is more effective.

The MDPlot also plots a lowess fit (in red) underlying a possible trend in the bias related to the mean expression.

**Methods**

```
signature(x = "matrix", y = "numeric")  
signature(x = "SeqExpressionSet", y = "numeric")
```



**Examples**

```

library(yeastRNASeq)
data(geneLevelData)
data(yeastGC)

sub <- intersect(rownames(geneLevelData), names(yeastGC))

mat <- as.matrix(geneLevelData[sub,])

data <- newSeqExpressionSet(mat,
  phenoData=AnnotatedDataFrame(
    data.frame(conditions=factor(c("mut", "mut", "wt", "wt")),
      row.names=colnames(geneLevelData))),
  featureData=AnnotatedDataFrame(data.frame(gc=yeastGC[sub])))

MDPlot(data,c(1,3))

```

---

meanVarPlot-methods      *Methods for Function meanVarPlot in Package EDASeq*

---

**Description**

meanVarPlot produces a smoothScatter plot of the mean variance relation.

**Methods**

signature(x = "SeqExpressionSet") It takes as additional argument log, which if true consider the logarithm of the counts before computing mean and variance. To avoid missing values, we consider the maximum between 0 and the log of the counts. Along with the scatter plot the function plots a line representing the [lowess](#) fit.

---

newSeqExpressionSet      *Function to create a new SeqExpressionSet object.*

---

**Description**

User-level function to create new objects of the class [SeqExpressionSet](#).

**Usage**

```

newSeqExpressionSet(counts,
  normalizedCounts = matrix(data=NA, nrow=nrow(counts), ncol=ncol(counts), dimnames=dimnames(counts)),
  offset = matrix(data=0, nrow=nrow(counts), ncol=ncol(counts), dimnames=dimnames(counts)),
  phenoData = annotatedDataFrameFrom(counts, FALSE),
  featureData = annotatedDataFrameFrom(counts, TRUE),
  ...)

```

**Arguments**

counts	A matrix containing the counts for an RNA-Seq experiment. One column for each lane and one row for each gene.
normalizedCounts	A matrix with the same dimensions of counts with the normalized counts.
offset	A matrix with the same dimensions of counts defining the offset (usually useful for normalization purposes). See the package vignette for a discussion on the offset.
phenoData	A data.frame or <a href="#">AnnotatedDataFrame</a> with sample information, such as biological condition, library preparation protocol, flow-cell,...
featureData	A data.frame or <a href="#">AnnotatedDataFrame</a> with feature information, such as gene length, GC-content, ...
...	Other arguments will be passed to the constructor inherited from <a href="#">eSet</a> .

**Value**

An object of class [SeqExpressionSet](#).

**Author(s)**

Davide Risso

**See Also**

[SeqExpressionSet](#)

**Examples**

```
counts <- matrix(data=0, nrow=100, ncol=4)
for(i in 1:4) {
  counts[, i] <- rpois(100, lambda=50)
}
cond <- c(rep("A", 2), rep("B", 2))

counts <- newSeqExpressionSet(counts, phenoData=data.frame(conditions=cond))
```

**Description**

High-level function to produce plots given one `BamFileList` object and one `FastqFileList` object.

**Methods**

signature(x = "BamFileList", y = "FastqFileList") It produce a barplot of the percentage of mapped reads. If strata=TRUE it stratifies the bars according to the unique/non-unique mapped reads. To be meaningful, x should be a set of aligned reads and y a set of raw reads on the same samples.

---

plotNtFrequency-methods

*Methods for Function plotNtFrequency in Package EDASeq*

---

**Description**

Plots the nucleotide frequencies per position.

**Methods**

signature(x = "ShortRead")

signature(x = "BamFile")

It plots the nucleotide frequencies per position, averaging all the reads in x.

---

plotPCA-methods

*Methods for Function plotPCA in Package EDASeq*

---

**Description**

plotPCA produces a Principal Component Analysis (PCA) plot of the counts in object

**Usage**

```
## S4 method for signature 'matrix'
plotPCA(object, k=2, labels=TRUE, isLog=FALSE, ...)
## S4 method for signature 'SeqExpressionSet'
plotPCA(object, k=2, labels=TRUE, ...)
```

**Arguments**

object	Either a numeric matrix or a <a href="#">SeqExpressionSet</a> object containing the gene expression.
k	The number of principal components to be plotted.
labels	Logical. If TRUE, and k=2, it plots the colnames of object as point labels.
isLog	Logical. Set to TRUE if the data are already on the log scale.
...	See <a href="#">par</a>

## Details

The Principal Component Analysis (PCA) plot is a useful diagnostic plot to highlight differences in the distribution of replicate samples, by projecting the samples into a lower dimensional space.

If there is strong differential expression between two classes, one expects the samples to cluster by class in the first few Principal Components (PCs) (usually 2 or 3 components are enough). This plot also highlights possible batch effects and/or outlying samples.

## Methods

```
signature(x = "matrix")  
signature(x = "SeqExpressionSet")
```

## Examples

```
library(yeastRNASeq)  
data(geneLevelData)  
  
mat <- as.matrix(geneLevelData)  
  
data <- newSeqExpressionSet(mat,  
                           phenoData=AnnotatedDataFrame(  
                             data.frame(conditions=factor(c("mut", "mut", "wt", "wt")),  
                                           row.names=colnames(geneLevelData))))  
  
plotPCA(data, col=rep(1:2, each=2))
```

---

plotQuality-methods    *Methods for Function plotQuality in Package EDASeq*

---

## Description

plotQuality produces a plot of the quality of the reads.

## Methods

```
signature(x = "BamFileList")
```

 It produces a plot that summarizes the per-base mean quality of the reads of each BAM file in x.  

```
signature(x = "BamFile")
```

 It produces a boxplot of the per-base distribution of the quality scores of the reads in x.  

```
signature(x = "FastqFileList")
```

 It produces a plot that summarizes the per-base mean quality of the reads of each FASTQ file in x.

## Details

Since FASTQ files can be very long, it can be very expensive to process a whole file. One way to avoid this, is to consider a subset of the file and then plot the quality of the subset. As long as one assumes that the subset is random, this is a good approximation. The function [FastqSampler](#) of [ShortRead](#) can be used for this. See its help page for an example.

---

plotRLE-methods

*Methods for Function plotRLE in Package EDASeq*

---

## Description

plotRLE produces a Relative Log Expression (RLE) plot of the counts in x

## Usage

```
plotRLE(x, ...)
```

## Arguments

x	Either a numeric matrix or a <a href="#">SeqExpressionSet</a> object containing the gene expression.
...	See <a href="#">par</a>

## Details

The Relative Log Expression (RLE) plot is a useful diagnostic plot to visualize the differences between the distributions of read counts across samples.

It shows the boxplots of the log-ratios of the gene-level read counts of each sample to those of a reference sample (defined as the median across the samples). Ideally, the distributions should be centered around the zero line and as tight as possible. Clear deviations indicate the need for normalization and/or the presence of outlying samples.

## Methods

```
signature(x = "matrix")  
signature(x = "SeqExpressionSet")
```

## Examples

```
library(yeastRNASeq)  
data(geneLevelData)  
  
mat <- as.matrix(geneLevelData)  
  
data <- newSeqExpressionSet(mat,  
                             phenoData=AnnotatedDataFrame(  
                               data.frame(conditions=factor(c("mut", "mut", "wt", "wt")),
```

```
row.names=colnames(geneLevelData))))
```

```
plotRLE(data, col=rep(2:3, each=2))
```

---

SeqExpressionSet-class

*"SeqExpressionSet" class for collections of short reads*

---

## Description

This class represents a collection of digital expression data (usually counts from RNA-Seq technology) along with sample information.

## Objects from the Class

Objects of this class can be created from a call to the [newSeqExpressionSet](#) constructor.

## Extends

Class eSet, directly. Class VersionedBiobase, by class eSet, distance 2. Class Versioned, by class eSet, distance 3.

## Slots

Inherited from eSet:

**assayData** Contains matrices with equal dimensions, and with column number equal to `nrow(phenoData)`. `assayData` must contain a matrix counts with rows representing features (e.g., genes) and columns representing samples. The optional matrices `normalizedCounts` and `offset` can be added to represent a normalization in terms of pseudo-counts or offset, respectively, to be used for subsequent analyses. See the vignette for details. Class: [AssayData-class](#).

**phenoData** Sample information. For compatibility with DESeq, there should be at least the column `conditions`. See [eSet](#) for details.

**featureData** Feature information. It is recommended to include at least `length` and `GC-content` information. This slot is used for [withinLaneNormalization](#). See [eSet](#) for details.

**experimentData** See [eSet](#)

**annotation** See [eSet](#)

**protocolData** See `link{eSet}`

**Methods**

See [eSet](#) for inherited methods. Additional methods:

**counts** signature(object="SeqExpressionSet"): returns the counts matrix.

**counts<-** signature(object = "SeqExpressionSet"): method to replace the counts matrix.

**normCounts** signature(object="SeqExpressionSet"): returns the normalizedCounts matrix.

**normCounts<-** signature(object = "SeqExpressionSet"): method to replace the normalizedCounts matrix.

**offst** signature(object = "SeqExpressionSet"): returns the offset matrix.

**offst<-** signature(object = "SeqExpressionSet"): method to replace the offset slot.

**boxplot** signature(x = "SeqExpressionSet"): produces a boxplot of the log counts.

**meanVarPlot** signature(x = "SeqExpressionSet"): produces a [smoothScatter](#) plot of the mean variance relation. See [meanVarPlot](#) for details.

**biasPlot** signature(x = "SeqExpressionSet", y = "character"): produces a plot of the [lowess](#) regression of the counts on some covariate of interest (usually GC-content or length). See [biasPlot](#) for details.

**withinLaneNormalization** signature(x = "SeqExpressionSet", y = "missing"): within lane normalization for GC-content (or other lane specific) bias. See [withinLaneNormalization](#) for details.

**betweenLaneNormalization** signature(x = "SeqExpressionSet"): between lane normalization for sequencing depth and possibly other distributional differences between lanes. See [betweenLaneNormalization](#) for details.

**Author(s)**

Davide Risso <risso.davide@gmail.com>

**See Also**

[eSet](#), [newSeqExpressionSet](#), [biasPlot](#), [withinLaneNormalization](#), [betweenLaneNormalization](#)

**Examples**

```
showMethods(class="SeqExpressionSet", where=getNamespace("EDASeq"))

counts <- matrix(data=0, nrow=100, ncol=4)
for(i in 1:4) {
  counts[,i] <- rpois(100,lambda=50)
}
cond <- c(rep("A", 2), rep("B", 2))

data <- newSeqExpressionSet(counts, phenoData=AnnotatedDataFrame(data.frame(conditions=cond)))

head(counts(data))
boxplot(data, col=as.numeric(pData(data)[,1])+1)
```

---

withinLaneNormalization-methods

*Methods for Function withinLaneNormalization in Package  
EDASeq*

---

## Description

Within-lane normalization for GC-content (or other lane-specific) bias.

## Usage

```
withinLaneNormalization(x, y, which=c("loess", "median", "upper", "full"), offset=FALSE, num.bins=10, round=TRUE)
```

## Arguments

x	A numeric matrix representing the counts or a <a href="#">SeqExpressionSet</a> object.
y	A numeric vector representing the covariate to normalize for (if x is a matrix) or a character vector with the name of the covariate (if x is a <a href="#">SeqExpressionSet</a> object). Usually it is the GC-content.
which	Method used to normalized. See the details section and the reference below for details.
offset	Should the normalized value be returned as an offset leaving the original counts unchanged?
num.bins	The number of bins used to stratify the covariate for median, upper and full methods. Ignored if loess. See the reference for a discussion on the number of bins.
round	If TRUE the normalization returns rounded values (pseudo-counts). Ignored if offset=TRUE.

## Details

This method implements four normalizations described in Risso et al. (2011).

The loess normalization transforms the data by regressing the counts on y and subtracting the loess fit from the counts to remove the dependence.

The median, upper and full normalizations are based on the stratification of the genes based on y. Once the genes are stratified in num.bins strata, the methods work as follows.

**median:** scales the data to have the same median in each bin.

**upper:** the same but with the upper quartile.

**full:** forces the distribution of each stratum to be the same using a non linear full quantile normalization, in the spirit of the one used in microarrays.



**Methods**

signature(x = "matrix", y = "numeric") It returns a matrix with the normalized counts if offset=FALSE or with the offset if offset=TRUE.

signature(x = "SeqExpressionSet", y = "character") It returns a [SeqExpressionSet](#) with the normalized counts in the normalizedCounts slot and with the offset in the offset slot (if offset=TRUE).

**Author(s)**

Davide Risso.

**References**

D. Risso, K. Schwartz, G. Sherlock and S. Dudoit (2011). GC-Content Normalization for RNA-Seq Data. Manuscript in Preparation.

**Examples**

```
library(yeastRNASeq)
data(geneLevelData)
data(yeastGC)

sub <- intersect(rownames(geneLevelData), names(yeastGC))

mat <- as.matrix(geneLevelData[sub, ])

data <- newSeqExpressionSet(mat,
                           phenoData=AnnotatedDataFrame(
                               data.frame(conditions=factor(c("mut", "mut", "wt", "wt")),
                                             row.names=colnames(geneLevelData))),
                           featureData=AnnotatedDataFrame(data.frame(gc=yeastGC[sub])))

norm <- withinLaneNormalization(data, "gc", which="full", offset=FALSE)
```

---

yeastGC

*GC-content of S. Cerevisiae genes*

---

**Description**

This data set gives the GC-content (proportion of G and C) of the genes of *S. Cerevisiae*, from SGD release 64 annotation.

**Format**

A vector containing 6717 observations.

**Source**

SGD release 64: <http://www.yeastgenome.org>

---

yeastLength

*Length of S. Cerevisiae genes*

---

**Description**

This data set gives the length (in base pairs) of the genes of *S. Cerevisiae*, from SGD release 64 annotation.

**Format**

A vector containing 6717 observations.

**Source**

SGD release 64: <http://www.yeastgenome.org>

# Index

- \* **classes**
  - SeqExpressionSet-class, [14](#)
- \* **datasets**
  - yeastGC, [17](#)
  - yeastLength, [18](#)
- \* **methods**
  - barplot-methods, [3](#)
  - betweenLaneNormalization-methods, [3](#)
  - biasBoxplot-methods, [5](#)
  - biasPlot-methods, [6](#)
  - boxplot-methods, [7](#)
  - MDPlot-methods, [8](#)
  - meanVarPlot-methods, [9](#)
  - plot-methods, [10](#)
  - plotNtFrequency-methods, [11](#)
  - plotPCA-methods, [11](#)
  - plotQuality-methods, [12](#)
  - plotRLE-methods, [13](#)
  - withinLaneNormalization-methods, [16](#)
- alphabetFrequency, [8](#)
- AnnotatedDataFrame, [10](#)
- BamFileList, [2](#)
- barplot, BamFile-method
  - (barplot-methods), [3](#)
- barplot, BamFileList-method
  - (barplot-methods), [3](#)
- barplot, FastqFileList-method
  - (barplot-methods), [3](#)
- barplot-methods, [3](#)
- betweenLaneNormalization, [15](#)
- betweenLaneNormalization
  - (betweenLaneNormalization-methods), [3](#)
- betweenLaneNormalization, matrix-method
  - (betweenLaneNormalization-methods), [3](#)
- betweenLaneNormalization, SeqExpressionSet-method
  - (betweenLaneNormalization-methods), [3](#)
- betweenLaneNormalization-methods, [3](#)
- biasBoxplot (biasBoxplot-methods), [5](#)
- biasBoxplot, numeric, numeric, numeric-method
  - (biasBoxplot-methods), [5](#)
- biasBoxplot, numeric, numeric-method
  - (biasBoxplot-methods), [5](#)
- biasBoxplot-methods, [5](#)
- biasPlot, [15](#)
- biasPlot (biasPlot-methods), [6](#)
- biasPlot, matrix, numeric-method
  - (biasPlot-methods), [6](#)
- biasPlot, SeqExpressionSet, character-method
  - (biasPlot-methods), [6](#)
- biasPlot-methods, [6](#)
- boxplot, FastqQuality-method
  - (boxplot-methods), [7](#)
- boxplot, SeqExpressionSet-method
  - (boxplot-methods), [7](#)
- boxplot-methods, [7](#)
- counts, SeqExpressionSet-method
  - (SeqExpressionSet-class), [14](#)
- counts<-, SeqExpressionSet-method
  - (SeqExpressionSet-class), [14](#)
- EDASeq (EDASeq-package), [2](#)
- EDASeq-package, [2](#)
- eSet, [2](#), [10](#), [14](#), [15](#)
- exprs, SeqExpressionSet-method
  - (SeqExpressionSet-class), [14](#)
- exprs<-, SeqExpressionSet, ANY-method
  - (SeqExpressionSet-class), [14](#)
- FastqFileList, [2](#)
- FastqSampler, [13](#)
- genes, [8](#)

- getGeneLengthAndGCCContent, 7
- getSeq, 8
- getSequence, 8
- initialize, SeqExpressionSet-method  
(SeqExpressionSet-class), 14
- lowess, 6, 9, 15
- MDPlot (MDPlot-methods), 8
- MDPlot, matrix, numeric-method  
(MDPlot-methods), 8
- MDPlot, SeqExpressionSet, numeric-method  
(MDPlot-methods), 8
- MDPlot-methods, 8
- meanVarPlot, 15
- meanVarPlot (meanVarPlot-methods), 9
- meanVarPlot, SeqExpressionSet-method  
(meanVarPlot-methods), 9
- meanVarPlot-methods, 9
- newSeqExpressionSet, 9, 14, 15
- normCounts (SeqExpressionSet-class), 14
- normCounts, SeqExpressionSet-method  
(SeqExpressionSet-class), 14
- normCounts<- (SeqExpressionSet-class),  
14
- normCounts<-, SeqExpressionSet, ANY-method  
(SeqExpressionSet-class), 14
- normCounts<-, SeqExpressionSet-method  
(SeqExpressionSet-class), 14
- offst (SeqExpressionSet-class), 14
- offst, SeqExpressionSet-method  
(SeqExpressionSet-class), 14
- offst<- (SeqExpressionSet-class), 14
- offst<- , SeqExpressionSet, ANY-method  
(SeqExpressionSet-class), 14
- offst<- , SeqExpressionSet-method  
(SeqExpressionSet-class), 14
- par, 5, 6, 8, 11, 13
- plot, BamFileList, FastqFileList-method  
(plot-methods), 10
- plot-methods, 10
- plotNtFrequency  
(plotNtFrequency-methods), 11
- plotNtFrequency, BamFile-method  
(plotNtFrequency-methods), 11
- plotNtFrequency, ShortRead-method  
(plotNtFrequency-methods), 11
- plotNtFrequency-methods, 11
- plotPCA (plotPCA-methods), 11
- plotPCA, matrix-method  
(plotPCA-methods), 11
- plotPCA, SeqExpressionSet-method  
(plotPCA-methods), 11
- plotPCA-methods, 11
- plotQuality (plotQuality-methods), 12
- plotQuality, BamFile-method  
(plotQuality-methods), 12
- plotQuality, BamFileList-method  
(plotQuality-methods), 12
- plotQuality, FastqFileList-method  
(plotQuality-methods), 12
- plotQuality-methods, 12
- plotRLE (plotRLE-methods), 13
- plotRLE, matrix-method  
(plotRLE-methods), 13
- plotRLE, SeqExpressionSet-method  
(plotRLE-methods), 13
- plotRLE-methods, 13
- Rsamtools, 2
- SeqExpressionSet, 2, 4, 8–11, 13, 16, 17
- SeqExpressionSet-class, 14
- smoothScatter, 15
- withinLaneNormalization, 14, 15
- withinLaneNormalization  
(withinLaneNormalization-methods),  
16
- withinLaneNormalization, matrix, numeric-method  
(withinLaneNormalization-methods),  
16
- withinLaneNormalization, SeqExpressionSet, character-method  
(withinLaneNormalization-methods),  
16
- withinLaneNormalization-methods, 16
- yeastGC, 17
- yeastLength, 18