Package 'NOISeq'

April 25, 2025

Type Package

Title Exploratory analysis and differential expression for RNA-seq data

Version 2.53.0

Date 2014-02-24

Author Sonia Tarazona, Pedro Furio-Tari, Maria Jose Nueda, Alberto Ferrer and Ana Conesa

Maintainer Sonia Tarazona <sotacam@eio.upv.es>

Depends R (>= 2.13.0), methods, Biobase (>= 2.13.11), splines (>= 3.0.1), Matrix (>= 1.2)

Description Analysis of RNA-seq expression data or other similar kind of data. Exploratory plots to evualuate saturation, count distribution, expression per chromosome, type of detected features, features length, etc. Differential expression between two experimental conditions with no parametric assumptions.

License Artistic-2.0

LazyLoad yes

biocViews ImmunoOncology, RNASeq, DifferentialExpression, Visualization, Sequencing

git_url https://git.bioconductor.org/packages/NOISeq

git_branch devel

git_last_commit 15be0cd

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-04-24

Contents

ARSyNseq 2
Biodetection 4
CD 5
CountsBio 6
Data2Save 7
Data_Exploration 8
degenes 9

Differential expression plots . . . 10
example . . . 11
Exploratory_Plots . . . 12
FilterLowCounts . . . 13
GCbias . . . 14
lengthbias . . . 15
Marioni . . . 16
myCounts . . . 16
noiseq . . . 17
noiseqbio . . . 18
Normalization . . . 20
Output . . . 22
PCA . . . 23
PCA.GENES . . . 24
QCreport . . . 24
readData . . . 25
Saturation . . . 27

Index **29**

`ARSyNseq` *ASCA Removal of Systematic Noise on Seq data*

Description

`ARSyNseq` filters the noise associated to identified or not identified batch effects considering the experimental design and applying Principal Component Analysis (PCA) to the ANOVA parameters and residuals.

Usage

```
ARSyNseq(data, factor = NULL, batch = FALSE, norm = "rpkm", logtransf = FALSE, Variability = 0.75, be
```

Arguments

`data`	A Biobase's eSet object created with the `readData` function.
`factor`	Name of the factor (as it was given to the *readData* function) to be used in the ARSyN model (e.g. the factor containing the batch information). When it is NULL, all the factors are considered.
`batch`	TRUE to indicate that the *factor* argument indicates the batch information. In this case, the *factor* argument must be used to specify the names of the only factor containing the information of the batch.
`norm`	Type of normalization to be used. One of "rpkm" (default), "uqua", "tmm" or "n" (if data are already normalized). If length was provided through the *readData* function, it will be considered for the normalization (except for "n"). Please note that if a normalization method if used, the arguments *lc* and *k* are set to 1 and 0 respectively.
`logtransf`	If FALSE, a log-transformation will be applied on the data before computing ARSyN model to improve the results of PCA on count data.

`Variability` Parameter for Principal Componentents (PCs) selection of the ANOVA models effects. This is the desired proportion of variability explained for the PC of the main effects (time and experimental group). Variability=0.75 by default.

`beta` Parameter for PCs selection of the residual model. Components selected will be those that explain more than beta times the average component variability computed as the total data variability divided by the rank of the matrix associated to the factor. Default beta=2.

Details

When batch is identified with one of the factors described in the argument `factor` of the `data` object, `ARSyNseq` estimates this effect and removes it by estimating the main PCs of the ANOVA effects associated. Selected PCs will be those that explain more than the variability proportion specified in `Variability`.

When batch is not identified, the model estimates the effects associated to each factor of interest and analyses if there exists systematic noise in the residuals. If there is batch effect, it will be identified with the main PCs of these residuals. Selected PCs will be those that explain more than `beta` times the average component variability.

Value

The Biobase's eSet object created with the `readData` function that was given as input but replacing the expression data with the filtered expression data matrix.

Author(s)

Maria Jose Nueda, <mj.nueda@ua.es>

References

Nueda, M.J.; Ferrer, A. and Conesa, A. (2012) ARSyN: a method for the identification and removal of systematic noise in multifactorial time-course microarray experiments. *Biostatistics* 13(3), 553-566.

Examples

```
# Generating an artificial batch effect from Marioni's data
data(Marioni)
set.seed(123)
mycounts2 = mycounts
mycounts2[,1:4] = mycounts2[,1:4] + runif(nrow(mycounts2)*4, 3, 5)
myfactors = data.frame(myfactors, "batch" = c(rep(1,4), rep(2,6)))
mydata2 = readData(mycounts2, factors = myfactors)

# Exploring batch effect with PCA
myPCA = dat(mydata2, type = "PCA")
par(mfrow = c(1,2))
explo.plot(myPCA, factor = "Tissue")
explo.plot(myPCA, factor = "batch")

# Removing batch effect when the batch is identified for each sample and exploring results with PCA
mydata2corr1 = ARSyNseq(mydata2, factor = "batch", batch = TRUE, norm = "rpkm", logtransf = FALSE)
myPCA = dat(mydata2corr1, type = "PCA")
par(mfrow = c(1,2))
explo.plot(myPCA, factor = "Tissue")
```

```
explo.plot(myPCA, factor = "batch")

# If we consider that exist a batch but it is not identified (we do not know the batch information):
mydata2corr2 = ARSyNseq(mydata2, factor = "Tissue", batch = FALSE, norm = "rpkm", logtransf = FALSE)
myPCA = dat(mydata2corr2, type = "PCA")
par(mfrow = c(1,2))
explo.plot(myPCA, factor = "Tissue")
explo.plot(myPCA, factor = "batch")
```

`Biodetection` *Biodetection class*

Description

Biodetection class generated from dat() function with type="biodetection". This object contains the percentage of each biological class (e.g. biotype) in the genome (i.e. in the whole set of features provided), the corresponding percentage detected by the sample and the percentage of the biotype within the sample.

Usage

```
## S4 method for signature 'Biodetection'
explo.plot(object, samples = c(1, 2), plottype = c("persample", "comparison"), toplot = "protein_coo
## S4 method for signature 'Biodetection'
dat2save(object)
```

Arguments

`object`	Object generated from `dat()` function.
`samples`	Samples or conditions to be plotted. If NULL, the two first samples are plotted because the plot for this object only admit a maximum of two samples.
`plottype`	If plottype="persample", each sample is plotted in a separate plot displaying abundance of byotype in genome, percentage of biotype detected by sample and abundance of biotype in sample. If plottype="comparison", two samples can be compared in the same plot. Two plots are generated, one for the percentage of biotype detected by each of the compared samples, and the other for the abundance of the biotypes within the compared samples.
`toplot`	If plottype="comparison" and a biotype is specified in this argument (by default toplot="protein_coding"), a proportion test is performed to test if the abundance of that biotype is significantly different for the two samples being compared.
`...`	Any argument from `par`.

Slots/List Components

An object of this class contains an element (dat) which is a list with the following components:

`genome`: Vector containing the percentage of features per biotype in the genome.

`biotables`: List with as many elements as samples or conditions. Each element of the list contains the percentage of features in the genome per biotype detected in that sample or condition features per biotype and the percentage of detected features in the sample or condition per biotype.

Methods

This class has an specific show method in order to work and print a summary of the elements which are contained and a dat2save method to save the relevant information in an object cleanly. It also has an explo.plot method to plot the data contained in the object.

Author(s)

Sonia Tarazona

CD *CD class*

Description

CD class generated from dat() function with type="cd". This object contains the distributions of log-fold changes (M values) between each of the samples and a reference sample as well as confidence intervals for the median of these distributions that are used to detect a potential RNA composition bias in the data.

Usage

```
## S4 method for signature 'CD'
explo.plot(object, samples = NULL, ...)
## S4 method for signature 'CD'
dat2save(object)
```

Arguments

object	Object generated from dat() function.
samples	Samples or conditions to be plotted. If NULL, the twelve first samples are plotted because the plot for this object only admit a maximum of twelve samples.
...	Any argument from par.

Slots/List Components

Objects of this class contain (at least) the following list components:

dat: List containing the following elements:

data2plot: Data frame where each column contains the M values obtained as the log2-ratio of each sample against the reference sample. refColumn: Column number in input data that is taken as the reference sample. DiagnosticTest: Data frame that contains the lower and upper limits of the confidence intervals for the median of M values per each sample. The last column indicates if the diagnostic test for that sample has been passed or failed (so normalization has to be applied).

Methods

This class has an specific show method in order to show the confidence intervals for the M median and a dat2save method to save the relevant information in the object in a user-friendly way. It also has an explo.plot method to plot the data contained in the object.

Author(s)

Sonia Tarazona

CountsBio *CountsBio class*

Description

CountsBio class generated from dat() function with type="countsbio". This object contains the count distribution for each biological group and also the percentage of features with counts per million higher than 0, 1, 2, 5 or 10, per each sample independently and in at least one of the samples (total).

Usage

```
## S4 method for signature 'CountsBio'
explo.plot(object, samples = c(1,2), toplot = "global", plottype = c("barplot", "boxplot"),...)
## S4 method for signature 'CountsBio'
dat2save(object)
```

Arguments

object	Object generated with dat() function.
toplot	This parameter indicates which biological group is to be plotted. It may be a number or a text with the name of the biological group. If toplot=1 (or "global"), a global plot with all the biological groups will be generated.
samples	Samples or conditions to be plotted. If NULL, the two first samples are plotted because the plot for this object only admit a maximum of two samples.
plottype	Type of plot to be generated for "countsbio" data. If "barplot", the plot indicates the percentage of features with counts per millior higher than 0, 1, 2, 5 or 10 counts or less. Above each bar, the sequencing depth (million reads) is shown. If "boxplot", a boxplot is drawn per sample or condition showing the count distribution for features with more than 0 counts. Both types of plot can be obtained for all features ("global") or for a specified biotype (when biotypes are available).
...	Any argument from par.

Slots/List Components

Objects of this class contain a list (dat) with the following components:

result: Matrix containing the expression data for all the detected features and all samples or conditions.

bionum: Vector containing the number of detected features per biological group (global indicates the total).

biotypes: Vector containing the biological group (biotype) for each detected feature.

summary: List with as many elements as number of biotypes and an additional element with the global information (for all features). Each element is a data frame containing for each sample or condition the number of features with 0 counts, 1 count or less, 2 counts or less, 5 counts or less and 10 counts or less, more than 10 counts, the total number of features and the sequencing depth.

Methods

This class has an specific show method in order to work and print a summary of the elements which are contained and a dat2save method to save the relevant information in an object cleanly. It also has an explo.plot method to plot the data contained in the object.

Author(s)

Sonia Tarazona

Data2Save *Saving data generated for exploratory plots.*

Description

This function is to save the data generated to draw the exploratory plots in a user-friendly format.

Value

The dat2save() function takes the object generated by dat() and creates a new one with the most relevant information.

Author(s)

Sonia Tarazona

See Also

readData, addData, dat, explo.plot.

Examples

```
## Load the input object with the expression data and the annotations
data(myCounts)

## Generating data for the plot "biodetection" and samples in columns 3 and 4 of expression data
mydata2plot = dat(mydata, type = "biodetection", k = 0)

## Save the relevant information cleanly
mydata2save = dat2save(mydata2plot)
```

Data_Exploration *Exploration of expression data.*

Description

Take the expression data and the feature annotations to generate the results that will be used for the exploratory plots (explo.plot) or saved by the user to perform other analyses.

Usage

```
dat(input, type = c("biodetection","cd","countsbio","GCbias","lengthbias","saturation","PCA"),
              k = 0, ndepth = 6, factor = NULL, norm = FALSE, refColumn = 1, logtransf = FALSE)
```

Arguments

input	Object of eSet class with expression data and optional annotation.
type	Type of plot for which the data are to be generated. It can be one of: "biodetection","cd","countsbio","GCbias","lengthbias","saturation".
k	A feature is considered to be detected if the corresponding number of read counts is > k. By default, k = 0. This parameter is used by types "biodetection" and "saturation".
ndepth	Number of different sequencing depths to be simulated and plotted apart from the real depth. By default, ndepth = 6. This parameter is only used by type "saturation".
factor	If factor = NULL (default), the calculations are done for each sample independently. When the factor is specified, the calculations are done for each experimental condition. Samples within the same condition are summed up ("biodetection") or averaged and normalized by sequencing depth ("countsbio", "GCbias" and "lengthbias").
norm	To indicate if provided data are already normalized (TRUE) or they are raw data (FALSE), which is the default. This parameter is used by types "cd", "lengthbias", "GCbias" and "countsbio".
refColumn	Column number in input data that is taken as the reference sample to compute M values. This parameter is only used by type "cd".
logtransf	To indicate if the data are already log-transformed (TRUE) or not (FALSE). If data are not log-transformed, a log-transformation will be applied before computing the Principal Component Analysis.

Value

dat() function returns an S4 object to be used by explo.plot() or to be converted into a more friendly formatted object by the dat2save() function.

Author(s)

Sonia Tarazona

See Also

Biodetection,CD,CountsBio,GCbias,lengthbias,Saturation,PCA,readData,addData,dat2save,explo.plot

Examples

```
## Load the input object with the expression data and the annotations
data(myCounts)

## Generating data for the plot "biodetection" and samples in columns 3 and 4 of expression data
mydata2plot = dat(mydata, type = "biodetection", k = 0)

## Generating the corresponding plot
explo.plot(mydata2plot, samples = c(3,4))
```

---

degenes *Recovering differencially expressed features.*

Description

Recovering differencially expressed features for a given threshold from noiseq or noiseqbio output objects.

Usage

```
degenes(object, q = 0.95, M = NULL)
```

Arguments

object	Object of class Output.
q	Value for the probability threshold (by default, 0.95).
M	String indicating if all differentially expressed features are to be returned or only up or down-regulated features. The possible values are: "up" (up-regulated in condition 1), "down" (down-regulated in condition 1), or NULL (all differentially expressed features).

Value

A matrix containing the differencially expressed features, the statistics and the probability of differential expression.

Author(s)

Sonia Tarazona

References

Marioni, J.C. and Mason, C.E. and Mane, S.M. and Stephens, M. and Gilad, Y. (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, **18**: 1509–1517.

See Also

readData, noiseq, noiseqbio.

Examples

```
## Load the object mynoiseq generated by computing differential expression probability with noiseq() on Marioni
data(noiseq)

## Third, use degenes() function to extract differentially expressed features:
mynoiseq.deg = degenes(mynoiseq, q = 0.8, M = NULL)
```

`Differential expression plots`

Plotting differential expression results

Description

Function to generate plots showing different aspects of differential expression results. Expression plot is to compare the expression values in each condition for all features. Differentially expressed features can be highlighted. Manhattan plot is to compare the expression values in each condition across all the chromosome positions. Differentially expressed features can also be highlighted. MD plot shows the values for (M,D) statistics. Differentially expressed features can also be highlighted. Distribution plot displays the percentage of differentially expressed features per chromosome and biotype (if this information is provided by the user).

Usage

```
DE.plot(output, q = NULL, graphic = c("MD","expr","chrom","distr"), pch = 20, cex = 0.5, col = 1, pch
chromosomes = NULL, join = FALSE,...)
```

Arguments

`output`	Object of class `Output`.
`q`	Probability of differential expression threshold to determine differentially expressed features.
`graphic`	String indicating which kind of plot is to be generated. If "expr", the feature expression values are depicted. If "MD", the values for the (M,D) statistics when comparing both conditions are used. If "chrom", the feature expression values are depicted across their positions in the chromosomes (if chromosome information has been provided). If "distr", two plots showing the percentage of differentially expressed features per both chromosome and biotype are generated (only if this information is available).
`pch, cex, col, ...`	Graphical parameters as in any other R plot. See `par`. They do not apply for graphic="chrom".
`pch.sel, cex.sel, col.sel`	`pch`, `cex` and `col`, respectively, to represent differentially expressed features. They do not apply for graphic="chrom".
`log.scale`	If TRUE, log2(data+K) values are depicted instead of the expression data in the `Output` object. K is an appropriate constant to avoid negative values. It does not apply for graphic="MD" and graphic="distr".

chromosomes	Character vector indicating the chromosomes to be plotted. If NULL, all chromosomes are plotted. It only applies for graphic="chrom" and graphic="distr". For graphic="chrom", the chromosomes are plotted in the given order. In some cases (e.g. chromosome names are character strings), it is very convenient to specify the order although all chromosomes are being plotted. For graphic="distr", the chromosomes are plotted according to the number of features they contain (from the highest number to the lowest).
join	If FALSE, each chromosome is depicted in a separate line. If TRUE, all the chromosomes are depicted in the same line, consecutively (useful for prokaryote organisms). It only applies for graphic="chrom".

Author(s)

Sonia Tarazona

See Also

readData, noiseq, degenes.

Examples

```
## We load the object generated after running noiseq on Marioni's data
data(noiseq)

## Third, plot the expression values for all genes and highlighting the differentially expressed genes
DE.plot(mynoiseq, q = 0.8, graphic = "expr", log.scale = TRUE)
DE.plot(mynoiseq, q = 0.8, graphic = "MD")
DE.plot(mynoiseq, chromosomes = c(1,2), log.scale = TRUE,join = FALSE, q = 0.8, graphic = "chrom")
DE.plot(mynoiseq, chromosomes = NULL, q = 0.8, graphic = "distr")
```

---

example *Example of objects used and created by the NOISeq package*

Description

This is a quick view of the objects generated by the package. To take a look, see the usage information. These objects have been created from Marioni's reduce dataset (only chromosomes I to IV).

Usage

```
# To load the object myCounts generated by the readData() function from R objects containing expressi
data(myCounts)

# To load the object generated after running the noiseq() function to compute differential expressior
data(noiseq)
```

References

Marioni, J.C. and Mason, C.E. and Mane, S.M. and Stephens, M. and Gilad, Y. (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, **18**: 1509–1517.

Exploratory_Plots *Exploratory plots for expression data.*

Description

Standard generic function. Different types of plots showing the biological classification for detected features, the expression distribution across samples or biological groups, the detection of technical bias such as length, GCcontent or RNA composition, the dependence of expression on sequencing depth, etc.

Usage

```
explo.plot(object, ...)
```

Arguments

object	Object generated with dat() function.
...	Any argument from par.

Value

The explo.plot() function takes the object generated by dat() and draws the corresponding plot.

Author(s)

Sonia Tarazona

See Also

Biodetection,CD,CountsBio,GCbias,lengthbias,Saturation, PCA, readData, addData, dat.

Examples

```
## Load the input object with the expression data and the annotations
data(myCounts)

## Generating data for the plot "biodetection" and samples in columns 3 and 4 of expression data
mydata2plot = dat(mydata, type = "biodetection", k = 0)

## Generating the corresponding plot
explo.plot(mydata2plot)
```

FilterLowCounts *Methods to filter out low count features*

Description

Function to filter out the low count features according to three different methods.

Usage

```
filtered.data(dataset, factor, norm = TRUE, depth = NULL, method = 1, cv.cutoff = 100, cpm = 1, p.adj
```

Arguments

dataset	Matrix or data.frame containing the expression values for each sample (columns) and feature (rows).
factor	Vector or factor indicating which condition each sample (column) in dataset belongs to.
norm	Logical value indicating whether the data are already normalized (TRUE) or not (FALSE).
depth	Sequencing depth of samples (column totals before normalizing the data). Depth only needs to be provided when method = 3 and norm = TRUE.
method	Method must be one of 1,2 or 3. Method 1 (CPM) removes those features that have an average expression per condition less than cpm value and a coefficient of variation per condition higher than cv.cutoff (in percentage) in all the conditions. Method 2 (Wilcoxon) performs a Wilcoxon test per condition and feature where in the null hypothesis the median expression is 0 and in the alternative the median is higher than 0. Those features with p-value greater than 0.05 in all the conditions are removed. Method 3 (Proportion test) performs a proportion test on the counts per condition and feature (or pseudo-counts if data were normalized) where null hypothesis is that the feature relative expression (count proportion) is equal to cpm/10^6 and higher than cpm/10^6 for the alternative. Those features with p-value greater than 0.05 in all the conditions are removed.
cv.cutoff	Cutoff for the coefficient of variation per condition to be used in method 1 (in percentage).
cpm	Cutoff for the counts per million value to be used in methods 1 and 3.
p.adj	Method for the multiple testing correction. The same methods as in the p.adjust function in stats package can be chosen: "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none".

Author(s)

Sonia Tarazona

Examples

```
## Simulate some count data
datasim = matrix(sample(0:100, 2000, replace = TRUE), ncol = 4)

## Filtering low counts (method 1)
myfilt1 = filtered.data(datasim, factor = c("cond1", "cond1", "cond2", "cond2"), norm = FALSE, depth = NULL, me
```

```
## Filtering low counts (method 2)
myfilt2 = filtered.data(datasim, factor = c("cond1", "cond1", "cond2", "cond2"), norm = FALSE, method = 2)

## Filtering low counts (method 3)
myfilt3 = filtered.data(datasim, factor = c("cond1", "cond1", "cond2", "cond2"), norm = FALSE, method = 3, cpm =
```

GCbias *GCbias class*

Description

GCbias class generated from dat() function with type="GCbias". This object contains the trimmed mean of expression for each GC content bin of 200 features per sample or condition and also per biotype (if available). It also includes the corresponding spline regression model fitted to explain the relationship between length and expression.

Usage

```
## S4 method for signature 'GCbias'
explo.plot(object, samples = NULL, toplot = "global", ...)
## S4 method for signature 'GCbias'
dat2save(object)
```

Arguments

object	Object generated with dat() function.
toplot	Biological group to be plotted (features not belonging to that group are discarded). It may be a number or a text with the name of the biological group. If toplot=1 or toplot="global", all features are used for the plot.
samples	Samples (or conditions) to be plotted. If NULL, all the samples are plotted. If samples > 2, only a descriptive plot will be generated. If not, diagnostic plots will be obtained showing both the R-squared and model p-value from the spline regression model describing the relationship between the GC content and the expression.
...	Any argument from par.

Slots/List Components

Objects of this class contain (at least) the following list components:

dat: List containing the information generated by dat() function. This list has the following elements:

data2plot: A list with as many elements as biological groups (the first element correspond to all the features). Each element of the list is a matrix containing the GC content bins in the first column and an additional column for the trimmed mean expression per bin for each sample or condition. RegressionModels: A list with as many elements as samples or conditions. Each element is an "lm" class object containing the spline regression model relating GC content and expression for that sample or condition (considering all the features).

Methods

This class has an specific show method to print a summary of spline regression models and a dat2save method to save the GC content bin information. It also has an explo.plot method to plot the data contained in the object.

Author(s)

Sonia Tarazona

---

lengthbias *lengthbias class*

Description

lengthbias class generated from dat() function with type="lengthbias". This object contains the trimmed mean of expression for each length bin of 200 features per sample or condition and also per biotype (if available). It also includes the corresponding spline regression models fitted to explain the relationship between length and expression.

Usage

```
## S4 method for signature 'lengthbias'
explo.plot(object, samples = NULL, toplot = "global", ...)
## S4 method for signature 'lengthbias'
dat2save(object)
```

Arguments

object	Object generated with dat() function.
toplot	Biological group to be plotted (features not belonging to that group are discarded). It may be a number or a text with the name of the biological group. If toplot=1 or toplot="global", all features are used for the plot.
samples	Samples (or conditions) to be plotted. If NULL, all the samples are plotted. If samples > 2, only a descriptive plot will be generated. If not, diagnostic plots will be obtained showing both the R-squared and model p-value from the spline regression model describing the relationship between the length and the expression.
...	Any argument from par.

Slots/List Components

Objects of this class contain (at least) the following list components:

dat: List containing the information generated by dat() function. This list has the following elements:

data2plot: A list with as many elements as biological groups (the first element correspond to all the features). Each element of the list is a matrix containing the length bins in the first column and an additional column for the trimmed mean expression per bin for each sample or condition. RegressionModels: A list with as many elements as samples or conditions. Each element is an "lm" class object containing the spline regression model relating length and expression for that sample or condition (considering all the features).

Methods

This class has an specific `show` method to print a summary of spline regression models and a `dat2save` method to save the length bin information. It also has an `explo.plot` method to plot the data contained in the object.

Author(s)

Sonia Tarazona

`Marioni` *Marioni's dataset*

Description

This is a reduced version for the RNA-seq count data from Marioni et al. (2008) along with additional annotation such as gene biotype, gene length, GC content, chromosome, start position and end position for genes in chromosomes I to IV. The expression data consists of 10 samples from kidney and liver tissues. There are five technical replicates (lanes) per tissue.

Usage

```
data(Marioni)
```

References

Marioni, J.C. and Mason, C.E. and Mane, S.M. and Stephens, M. and Gilad, Y. (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, **18**: 1509–1517.

`myCounts` *Class myCounts*

Description

This is the main class which contains the information needed to do the different analyses.

Extends

Class `eSet` (package 'Biobase').

Quick View

This object will contain the expression data and further information needed to do the exploratory analysis or the normalization such as the length, GC content, biotypes, chromosomes and positions for each feature.

Internally, the data is stored as follows:

As `myCounts` derives from `eSet`, we have used the slot `assayData` to store all the expression data, `phenoData` to store the factors with the conditions, `featureData` which will contain the variables `Length`, `GCcontent`, `Biotype`, `Chromosome`, Start Position, `End Position` for each feature. It has been used the slot `experimentData` derived from `MIAME-class` which will contain the type of replicates (biological replicates, technical replicates or no replicates at all).

Author(s)

Sonia Tarazona

See Also

If you need further information to know the methods that can be used, see `eSet`, `AnnotatedDataFrame-class`, `MIAME-class`.

`noiseq` *Differential expression method for technical replicates or no replicates at all*

Description

`noiseq` computes differential expression between two experimental conditions from read count data (e.g. RNA-seq).

Usage

```
noiseq(input, k = 0.5, norm = c("rpkm","uqua","tmm","n"),
replicates = c("technical","biological","no"),
factor=NULL, conditions=NULL, pnr = 0.2, nss = 5, v = 0.02, lc = 0)
```

Arguments

`input`	Object of eSet class coming from `readData` function or other R packages such as DESeq.
`factor`	A string indicating the name of factor whose levels are the conditions to be compared.
`conditions`	A vector containing the two conditions to be compared by the differential expression algorithm (needed when the `factor` contains more than 2 different conditions).
`replicates`	In this argument, the type of replicates to be used is defined: "technical", "biological" or "no" replicates. By default, "technical" replicates option is chosen.
`k`	Counts equal to 0 are replaced by k. By default, k = 0.5.
`norm`	Normalization method. It can be one of "rpkm" (default), "uqua" (upper quartile), "tmm" (trimmed mean of M) or "n" (no normalization).
`lc`	Length correction is done by dividing expression by length^lc. By default, lc = 0.
`pnr`	Percentage of the total reads used to simulated each sample when no replicates are available. By default, pnr = 0.2.
`nss`	Number of samples to simulate for each condition (nss>= 2). By default, nss = 5.
`v`	Variability in the simulated sample total reads. By default, v = 0.02. Sample total reads is computed as a random value from a uniform distribution in the interval [(pnr-v)*sum(counts), (pnr+v)*sum(counts)]

Value

The function returns an object of class Output

Author(s)

Sonia Tarazona

References

Bullard J.H., Purdom E., Hansen K.D. and Dudoit S. (2010) Evaluation of statistical methods for normalization and differential expression in mRNA-seq experiments. *BMC Bioinformatics* 11(1):94+.

Mortazavi A., Williams B.A., McCue K., Schaeer L. and Wold B. (2008) Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nature Methods* 5(7):621-628.

Robinson M.D. and Oshlack A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology* 11(3):R25+.

Marioni, J.C. and Mason, C.E. and Mane, S.M. and Stephens, M. and Gilad, Y. (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, **18**: 1509–1517.

See Also

readData.

Examples

```
## Load the input object from Marioni's data as returned by readData()
data(myCounts)

## Computing differential expression probability on RPKM-normalized data by NOISeq-real using factor "Tissue"
mynoiseq = noiseq(mydata, k = 0.5, norm = "rpkm", replicates = "technical", factor="Tissue",
pnr = 0.2, nss = 5, v = 0.02, lc = 1)


## Computing differential expression probability on Upper Quartile normalized data by NOISeq-real
## using factor "TissueRun" and comparing samples in Run 1 (levels "Kidney_1" and "Liver_1")
mynoiseq.uqua = noiseq(mydata, k = 0.5, norm = "uqua", replicates = "technical", factor="TissueRun",
conditions = c("Kidney_1","Liver_1"), pnr = 0.2, nss = 5, v = 0.02, lc = 1)
```

---

noiseqbio *Differential expression method for biological replicates*

Description

noiseqbio computes differential expression between two experimental conditions from read count data (e.g. RNA-seq).

Usage

```
noiseqbio(input, k = 0.5, norm = c("rpkm","uqua","tmm","n"), nclust = 15, plot = FALSE,
                      factor=NULL, conditions = NULL, lc = 0, r = 50, adj = 1.5,
                     a0per = 0.9, random.seed = 12345, filter = 1, depth = NULL,
                      cv.cutoff = 500, cpm = 1)
```

Arguments

input	Object of eSet class coming from readData function or other R packages such as DESeq.
k	Counts equal to 0 are replaced by k. By default, k = 0.5.
norm	Normalization method. It can be one of "rpkm" (default), "uqua" (upper quartile), "tmm" (trimmed mean of M) or "n" (no normalization).
factor	A string indicating the name of factor whose levels are the conditions to be compared.
conditions	A vector containing the two conditions to be compared by the differential expression algorithm (needed when the factor contains more than 2 different conditions).
lc	Length correction is done by dividing expression by length^lc. By default, lc = 0.
r	Number of permutations to generate noise distribution by resampling.
adj	Smoothing parameter for the Kernel Density Estimation of noise distribution. Higher values produce smoother curves.
nclust	Number of clusters for the K-means algorithm. Used when the number of replicates per condition is less than 5.
plot	If TRUE, a plot is generated showing the mixture distribution (f) and the noise distribution (f0) of theta values.
a0per	M and D values are corrected for the biological variability by being divided by S + a0, where S is the standard error of the corresponding statistic and a0 is determined by the value of a0per parameter. If a0per is NULL, a0 = 0. If a0per is a value between 0 and 1, a0 is the a0per percentile of S values for all features. If a0per = "B", a0 takes the highest value given by 100*max(S).
random.seed	Random seed. In order to get the same results in different runs of the method (otherwise the resampling procedure would produce different resulst), the random seed is set to this parameter value.
filter	Method to filter out low count features before computing differential expression analysis. If filter=0, no filtering is performed. If 1, CPM method is applied. If 2, Wilcoxon test method (not recommended when the number of replicates per condition is less than 5), If 3, proportion test method. Type ?filtered.data for more details.
depth	Sequencing depth of each sample to be used by filtering method. It must be data provided when the data is already normalized and filtering method 3 is to be applied.
cv.cutoff	Cutoff for the coefficient of variation per condition to be used in filtering method 1.
cpm	Cutoff for the counts per million value to be used in filtering methods 1 and 3.

Value

The function returns an object of class Output

Author(s)

Sonia Tarazona

References

Bullard J.H., Purdom E., Hansen K.D. and Dudoit S. (2010) Evaluation of statistical methods for normalization and differential expression in mRNA-seq experiments. *BMC Bioinformatics* 11(1):94+.

Mortazavi A., Williams B.A., McCue K., Schaeer L. and Wold B. (2008) Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nature Methods* 5(7):621-628.

Robinson M.D. and Oshlack A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology* 11(3):R25+.

Marioni, J.C. and Mason, C.E. and Mane, S.M. and Stephens, M. and Gilad, Y. (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, **18**: 1509–1517.

See Also

readData.

Examples

```
## Load the input object from Marioni's data as returned by readData()
data(myCounts)

## Computing differential expression probability by NOISeqBIO using factor "Tissue" (data will be RPKM-normaliz
mynoiseqbio = noiseqbio(mydata, k = 0.5, norm = "rpkm", factor="Tissue", lc = 1, r = 50, adj = 1.5, plot = FALSE,
                    a0per = 0.9, random.seed = 12345, filter = 1, cv.cutoff = 500, cpm = 1)
```

Normalization *Normalization methods*

Description

Normalization procedures such as RPKM (Mortazavi et al., 2008), Upper Quartile (Bullard et al., 2010) and TMM (Trimmed Mean of M) (Robinson and Oshlack, 2010). These normalization functions are used within the noiseq or noiseqbio functions but may be also used by themselves to normalize a dataset.

Usage

```
uqua(datos, long = 1000, lc = 0, k = 0)
rpkm(datos, long = 1000, lc = 1, k = 0)
tmm(datos, long = 1000, lc = 0, k = 0, refColumn = 1, logratioTrim = 0.3, sumTrim = 0.05, doWeighting =
```

Arguments

datos	Matrix containing the read counts for each sample.
long	Numeric vector containing the length of the features. If long == 1000, no length correction is applied (no matter the value of parameter lc).
lc	Correction factor for length normalization. This correction is done by dividing the counts vector by (length/1000)^lc. If lc = 0, no length correction is applied. By default, lc = 1 for RPKM and lc = 0 for the other methods.
k	Counts equal to 0 are changed to k in order to avoid indeterminations when applying logarithms, for instance. By default, k = 0.
refColumn	Column to use as reference (only needed for tmm function).
logratioTrim	Amount of trim to use on log-ratios ("M" values) (only needed for tmm function).
sumTrim	Amount of trim to use on the combined absolute levels ("A" values) (only needed for tmm function).
doWeighting	Logical, whether to compute (asymptotic binomial precision) weights (only needed for tmm function).
Acutoff	Cutoff on "A" values to use before trimming (only needed for tmm function).

Details

tmm normalization method was taken from *edgeR* package (Robinson et al., 2010).

Although Upper Quartile and TMM methods themselves do not correct for the length of the features, these functions in NOISeq allow users to combine the normalization procedures with an additional length correction whenever the length information is available.

Author(s)

Sonia Tarazona

References

Bullard J.H., Purdom E., Hansen K.D. and Dudoit S. (2010) Evaluation of statistical methods for normalization and differential expression in mRNA-seq experiments. *BMC Bioinformatics* 11(1):94+.

Mortazavi A., Williams B.A., McCue K., Schaeer L. and Wold B. (2008) Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nature Methods* 5(7):621-628.

Robinson M.D. and Oshlack A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology* 11(3):R25+.

Robinson M.D., McCarthy D.J. and Smyth G.K. (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26(1):139-140.

Examples

```
## Simulate some count data and the features length
datasim = matrix(sample(0:100, 2000, replace = TRUE), ncol = 4)
lengthsim = sample(100:1000, 500)

## RPKM normalization
myrpkm = rpkm(datasim, long = lengthsim, lc = 1, k = 0)

## Upper Quartile normalization, dividing normalized data by the square root of the features length and replacin
```

```
myuqua = uqua(datasim, long = lengthsim, lc = 0.5, k = 1)

## TMM normalization with no length correction
mytmm = tmm(datasim, long = 1000, lc = 0, k = 0)
```

`Output` *Output class of NOISeq*

Description

Output object containing the results from differential expression analysis by `noiseq` or `noiseqbio`.

Slots/List Components

Objects of this class contain (at least) the following list components:

`comparison`: String indicating the two experimental conditions being compared and the sense of the comparison.

`factor`: String indicating the factor chosen to compute the differential expression.

`k`: Value to replace zeroes in orden to avoid inderminations when computing logarithms.

`lc`: Correction factor for length normalization. Counts are divided by length^lc.

`method`: Normalization method chosen. It can be one of "rpkm" (default), "uqua" (Upper Quartile), "tmm" (Trimmed Mean of M) or "n" (no normalization).

`replicates`: Type of replicates: "technical" for technical replicates and "biological" for biological ones.

`results`: R data frame containing the differential expression results, where each row corresponds to a feature. The columns are: Expression values for each condition to be used by `noiseq` or `noiseqbio` (the columns names are the levels of the factor); differential expression statistics (columns "M" and "D" for `noiseq` or "theta" for `noiseqbio`); probability of differential expression ("prob"); "ranking", which is a summary statistic of "M" and "D" values equal to -sign(M)*sqrt(M^2 + D^2), than can be used for instance in gene set enrichment analysis (only when `noiseq` is used); "length" and "GC" of each feature (if provided); chromosome where the feature is ("Chrom"), if provided; start and end position of the feature within the chromosome ("GeneStart", "GeneEnd"), if provided.

`nss`: Number of samples to be simulated for each condition (only when there are not replicates available).

`pnr`: Percentage of the total sequencing depth to be used in each simulated replicate (only when there are not replicates available). If, for instance, pnr = 0.2 , each simulated replicate will have 20% of the total reads of the only available replicate in that condition.

`v`: Variability of the size of each simulated replicate (only used by NOISeq-sim).

Methods

This class has an specific `show` method in order to work and print a summary of the elements which are contained.

Author(s)

Sonia Tarazona

PCA *PCA class*

Description

PCA class generated from dat() function with type="PCA". This object contains the results of the PCA on the data matrix as well as the arguments used.

Usage

```
## S4 method for signature 'PCA'
explo.plot(object, samples = 1:2, plottype = "scores", factor = NULL)
## S4 method for signature 'PCA'
dat2save(object)
```

Arguments

object	Object generated from dat() function.
samples	Principal components to be plotted. If NULL, the two first components are plotted.
plottype	If plottype="scores", the experimental samples are displayed in the plot and colored according to the values of the selected factor. If plottype="loadings", the genes are plotted.
factor	The samples in the score plot will be colored according to the values of the selected factor. If NULL, the first factor is chosen.

Slots/List Components

An object of this class contains an element (dat) which is a list with the following components:

result: List containing the output of PCA. It contains the following elements: "eigen" (eigenvalues and eigenvectors from the PCA decomposition), "var.exp" (variance explained by each Principal Component), "scores" (coefficients of samples in each PC), "loadings" (coefficients of genes in each PC).

factors: Data.frame with factors inherited from object generated by readData() function.

norm: Value provided for argument "norm".

logtransf: Value provided for argument "logtransf".

Methods

This class has an specific show method in order to work and print a summary of the elements which are contained and a dat2save method to save the relevant information in an object cleanly. It also has an explo.plot method to plot the data contained in the object.

Author(s)

Sonia Tarazona

PCA.GENES *Principal Component Analysis*

Description

Computes a Principal Component Analysis on any data matrix.

Usage

```
PCA.GENES(X)
```

Arguments

X	Matrix or data.frame with variables (e.g. genes) in columns and observations (e.g. samples) in rows.

Author(s)

Maria Jose Nueda

Examples

```
## Simulate data matrix with 500 variables and 10 observations
datasim = matrix(sample(0:100, 5000, replace = TRUE), nrow = 10)

## PCA
myPCA = PCA.GENES(datasim)

## Extracting the variance explained by each principal component
myPCA$var.exp
```

QCreport *Quality Control report for expression data*

Description

Generate a report with the exploratory plots for count data that can be generated from the biological information provided. This report is designed to compare two samples or two experimental conditions.

Usage

```
QCreport(input, file = NULL, samples = NULL, factor = NULL, norm = FALSE)
```

Arguments

input	Object of eSet class coming from readData function or other R packages such as DESeq.
file	String indicating the name of the PDF file that will contain the report. It should be in this format: "filename.pdf". The default name is like this: "QCreport_2013Sep26_15:58:16.pdf"
samples	Vector with the two samples to be compared in the report when "factor" is NULL. If "factor" is not NULL and has more than two levels, samples has to indicate the two conditions to be compared. It can be numeric or character (when names of samples or conditions are provided).
factor	If NULL, individual samples indicated in "samples" are compared. Otherwise, it should be a string indicating the factor containing the experimental conditions to be compared in the report.
norm	TRUE to indicate that data are already normalized.

Value

A pdf file.

Author(s)

Sonia Tarazona

References

Marioni, J.C. and Mason, C.E. and Mane, S.M. and Stephens, M. and Gilad, Y. (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, **18**: 1509–1517.

Examples

```
## Load the input object from Marioni's data as returned by readData()
data(myCounts)

## Generate the report
QCreport(mydata, samples = NULL, factor = "Tissue")
```

---

readData *Creating an object of eSet class*

Description

This function is to create an object of eSet class to be used by NOISeq functions from matrix or data.frame R objects.

Usage

```
readData(data, factors, length = NULL, biotype = NULL, chromosome = NULL, gc = NULL)
addData(data, length = NULL, biotype = NULL, chromosome = NULL, factors = NULL, gc = NULL)
```

Arguments

data	Matrix or data.frame containing the counts (or expression data) for each feature and sample. Features must be in rows and samples must be in columns.
factors	A data.frame containing the experimental condition or group for each sample (columns in the data object).
biotype	Optional argument.Vector, matrix or data.frame containing the biological group (biotype) for each feature. In case of giving a vector, the names of the vector must be the feature names or ids with the same type of identifier used in data. If a matrix or a data.frame is provided, and it has two columns, it is expected that the feature names or ids are in the first column and the biotypes of the features in the second. If it only has one column containing the biotypes, the rownames of the object must be the feature names or ids.
chromosome	Optional argument. A matrix or data.frame containing the chromosome, start position and end position of each feature. The rownames must be the feature names or ids with the same type of identifier used in data.
gc	Optional argument.Vector, matrix or data.frame containing the GC content of each feature. In case of giving a vector, the names of the vector must be the feature names or ids with the same type of identifier used in data. If a matrix or a data.frame is provided, and it has two columns, it is expected that the feature names or ids are in the first column and the GC content of the features in the second. If it only has one column containing the GC content, the rownames of the object must be the feature names or ids.
length	Optional argument.Vector, matrix or data.frame containing the length of each feature. In case of giving a vector, the names of the vector must be the feature names or ids with the same type of identifier used in data. If a matrix or a data.frame is provided, and it has two columns, it is expected that the feature names or ids are in the first column and the length of the features in the second. If it only has one column containing the length, the rownames of the object must be the feature names or ids.

Value

It returns an object of eSet class myCounts with all the information defined and ready to be used.

Author(s)

Sonia Tarazona

References

Marioni, J.C. and Mason, C.E. and Mane, S.M. and Stephens, M. and Gilad, Y. (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, **18**: 1509–1517.

Examples

```
# Load an object containing the information explained above
data(Marioni)

# Create the object with the data
mydata <- readData(data=mycounts, biotype=mybiotypes, chromosome=mychroms, factors=myfactors)
```

```
# Add length annotation to the existing data object
mydata <- addData(mydata, length=mylength)
```

`Saturation` *Saturation class*

Description

Saturation class generated from dat() function with type="saturation". This object contains the number of detected features per biotype at increasing sequencing depths and also the new detections per each million of new sequencing reads.

Usage

```
## S4 method for signature 'Saturation'
explo.plot(object, samples = NULL, toplot = 1, yleftlim = NULL, yrightlim = NULL, ...)
## S4 method for signature 'Saturation'
dat2save(object)
```

Arguments

`object`	Object generated from `dat()` function.
`toplot`	This parameter indicates which biological group is to be plotted. It may be a number or a text with the name of the biological group. If toplot=1 (or "global"), a global plot considering features from all the biological groups will be generated.
`samples`	The samples to be plotted. If NULL, all the samples are plotted for Saturation object.
`yleftlim`	Range for Y left-axis (on the left-hand side of the plot) when new detections are plotted (this occurs when the number of samples to be plotted is 1 or 2). If NULL (default), an appropriate range is computed.
`yrightlim`	Range for Y right-axis (on the right-hand side of the plot) when new detections are plotted (this occurs when the number of samples to be plotted is 1 or 2). If NULL (default), an appropriate range is computed.
`...`	Any argument from `par`.

Slots/List Components

Objects of this class contain (at least) the following list components:

`dat`: List containing the information generated by dat() function. This list has the following elements:

`saturation`: List containing for all the biological classes (and also a global class with all of them together) the saturation data to be plotted for each sample (in Y left axis).

`bionum`: Vector containing for all the biological classes (and also a global class with all of them together) the number of features for that group.

`depth`: List containing for each selected sample the increasing values of sequencing depth to be plotted.

`newdet`: List containing for all the biological classes (and also a global class with all of them together) the new detection data to be plotted for each selected sample (in Y right axis).

`real`: List with as many elements as the number of biological classes (plus one for the global). Each element contains the real sequencing depth for each sample and the corresponding number of detected features at that sequencing depth.

Methods

This class has an specific `show` method in order to work and print a summary of the elements which are contained and a `dat2save` method to save the relevant information in an object cleanly. It also has an `explo.plot` method to plot the data contained in the object.

Author(s)

Sonia Tarazona

Index

∗ **ASCA, ANOVA, PCA, batch**
ARSyNseq, 2
∗ **classes**
Biodetection, 4
CD, 5
CountsBio, 6
GCbias, 14
lengthbias, 15
myCounts, 16
Output, 22
PCA, 23
Saturation, 27
∗ **datasets**
example, 11
Marioni, 16

addData, *7, 8*, *12*
addData *(*readData*)*, 25
ARSyNseq, 2
arsynseq *(*ARSyNseq*)*, 2

Biodetection, 4, *8*, *12*
Biodetection-class *(*Biodetection*)*, 4

CD, 5, *8*, *12*
CD-class *(*CD*)*, 5
CountsBio, 6, *8*, *12*
CountsBio-class *(*CountsBio*)*, 6

dat, *7*, *12*
dat *(*Data_Exploration*)*, 8
dat2save, *8*
dat2save *(*Data2Save*)*, 7
dat2save,Biodetection-method *(*Biodetection*)*, 4
dat2save,CD-method *(*CD*)*, 5
dat2save,CountsBio-method *(*CountsBio*)*, 6
dat2save,GCbias-method *(*GCbias*)*, 14
dat2save,lengthbias-method *(*lengthbias*)*, 15
dat2save,PCA-method *(*PCA*)*, 23
dat2save,Saturation-method *(*Saturation*)*, 27
Data2Save, 7
Data_Exploration, 8
DE.plot *(*Differential expression plots*)*, 10
degenes, 9, *11*
Differential expression plots, 10

example, 11
explo.plot, *7, 8*
explo.plot *(*Exploratory_Plots*)*, 12
explo.plot,Biodetection-method *(*Biodetection*)*, 4
explo.plot,CD-method *(*CD*)*, 5
explo.plot,CountsBio-method *(*CountsBio*)*, 6
explo.plot,GCbias-method *(*GCbias*)*, 14
explo.plot,lengthbias-method *(*lengthbias*)*, 15
explo.plot,PCA-method *(*PCA*)*, 23
explo.plot,Saturation-method *(*Saturation*)*, 27
Exploratory_Plots, 12

filtered.data *(*FilterLowCounts*)*, 13
FilterLowCounts, 13

GCbias, *8*, *12*, 14
GCbias-class *(*GCbias*)*, 14

lengthbias, *8*, *12*, 15
lengthbias-class *(*lengthbias*)*, 15

Marioni, 16
mybiotypes *(*Marioni*)*, 16
mychroms *(*Marioni*)*, 16
myCounts, 16, *26*
mycounts *(*Marioni*)*, 16
myCounts-class *(*myCounts*)*, 16
mydata *(*example*)*, 11
myfactors *(*Marioni*)*, 16
mygc *(*Marioni*)*, 16
mylength *(*Marioni*)*, 16
mynoiseq *(*example*)*, 11

noiseq, *9*, *11*, 17
noiseqbio, *9*, 18

Normalization, 20

Output, *9, 10*, *18*, *20*, 22
Output-class *(*Output*)*, 22

par, *10*
PCA, *8*, *12*, 23
PCA-class *(*PCA*)*, 23
PCA.GENES, 24

QCreport, 24

readData, *7–9*, *11, 12*, *17–20*, *25*, 25
rpkm *(*Normalization*)*, 20

Saturation, *8*, *12*, 27
saturation *(*Saturation*)*, 27
Saturation-class *(*Saturation*)*, 27
show,Biodetection-method
*(*Biodetection*)*, 4
show,CD-method *(*CD*)*, 5
show,CountsBio-method *(*CountsBio*)*, 6
show,GCbias-method *(*GCbias*)*, 14
show,lengthbias-method *(*lengthbias*)*, 15
show,Output-method *(*Output*)*, 22
show,PCA-method *(*PCA*)*, 23
show,Saturation-method *(*Saturation*)*, 27

tmm *(*Normalization*)*, 20

uqua *(*Normalization*)*, 20