Package 'RNAontheBENCH'

March 25, 2016

Type Package

Title RNAontheBENCH: A benchmarking ressource for RNAseq quantification and differential expression analysis methods

Version 1.2

Date 2016-03-18

Author Pierre-Luc Germain

Maintainer `<pierre.germain@ieo.eu>`

Depends MASS, edgeR, limma

Suggests DESeq, DESeq2, sleuth, rminer, rmarkdown, knitr

URL `https://github.com/plger/RNAontheBENCH`

Description RNA on the Benchmark of Expression by nCounter Hybridization (RNAontheBENCH) is a resource for benchmarking RNAseq quantification and differential expression analysis methods on the basis of empirical data.

License GPL

RoxygenNote 5.0.1

VignetteBuilder rmarkdown,
knitr

R topics documented:

`analyzeSpikein` *Comparison quantification with the expected spike-in concentrations.*

Description

Loads and processes the RNAseq data before comparing it with the spike-in concentrations. Will produce a number of benchmarking plots and files in the current working directory.

Usage

```
analyzeSpikein(ANALYSIS_NAME, rnaseq = NULL, qt, fc.undetected = 1)
```

Arguments

`ANALYSIS_NAME`	A string indicating the name of the analysis/pipeline. Will be used in filenames, plot titles, etc.
`rnaseq`	The path to the gene-level RNAseq expression matrix. If not given, will look for relevant files in the working directory. The expression matrix should have gene symbols in the first column/row.names, and sample names (e.g. 'AJ80' for the 12-samples dataset, "A_1" for SEQC, etc) as column headers.
`qt`	A string indicating the unit of the expression matrix (either "FPKM", "TPM" or "COUNTS").
`fc.undetected`	The foldchange to assign to undetected spike-ins (should be either 1 or NA, default 1)

Value

Nothing, but produces many files in the working directory...

Examples

```
# first we create a directory and put the example quantification file in it:
data(exampledata)
dir.create("example")
write.table(exampleGeneLevel,"w12.genes.quant",sep="\t",quote=F)
# then we run the function, giving a name to the analysis,
# specifying the file and type of quantification:
analyzeSpikein("tophat.featureCount", "w12.genes.quant", qt="COUNTS")
```

`auc` *auc*

Description

Computes the area under a curve.

Usage

```
auc(x, y, dens = 100)
```

Arguments

`x`	x values.
`y`	corresponding y values.
`dens`	Number of points.

Value

The area under the curve.

`benchmarkWrapper` *A wrapper that performs the whole benchmark analysis*

Description

Performs the whole series of benchmark analysis (see `compareWithNanostring`, `analyzeSpikein`, `compareWithPCR`, and `compareSimulated`). The function expects quantification files with the right column headers ("AJ80" and so on, or "s1","s2" and so on for the simulated data) to be in the rpath folder, and to bear some kind of recognizable name. However, to avoid confusion, we suggest using the filename 'transcripts.quant' for transcript-level quantification, 'genes.quant' for gene-level quantification (optional), and 'simulated.quant' for transcript-level quantification of the simulated dataset (optional). If you are benchmarking both the core (12-samples) dataset and the validation (6-samples) dataset, prefix the files with, respectively, 'w12.' and 'w6.' (see example below). If you instead want to specify files manually, use the individual underlying functions.

Usage

```
benchmarkWrapper(rpath, ANALYSIS_NAME, qt)
```

Arguments

`rpath`	The path were the quantification files are stored, and where the output files will be saved.
`ANALYSIS_NAME`	The name of the analysis pipeline
`qt`	A string indicating the unit of the expression matrix (either "FPKM", "TPM" or "COUNTS").

Value

Nothing, but saves a bunch of files in 'rpath' and opens an html page to browse the results.

Examples

```
# first we create a directory and put the example quantification file in it:
data(exampledata)
dir.create("example")
write.table(exampleTranscriptLevel,"w12.transcripts.quant",sep="\t",quote=F)
write.table(exampleGeneLevel,"w12.genes.quant",sep="\t",quote=F)
# run the wrapper, specifying that folder:
benchmarkWrapper("example", "tophat.featureCount", qt="COUNTS")
```

`checkMatIntegrity` *checkMatIntegrity*

Description

Replaces missing and invalid values with 0, and makes sure the matrix has all rows and samples in 'against'.

Usage

```
checkMatIntegrity(x, against = NULL)
```

Arguments

`x`	a matrix or data.frame
`against`	(optional) another matrix or data.frame, of which all rows and columns should be present in x.

Value

a clean version of x, or an error if there are missing columns.

Examples

```
checkMatIntegrity(matrix(c(1:8,NA),nrow=3))
```

`compareSimulated` *Transcript-level benchmark of the simulated data*

Description

Loads and processes the RNAseq data before comparing it with the real foldchanges. Will produce a number of benchmarking plots and files in the current working directory.

Usage

```
compareSimulated(ANALYSIS_NAME, txfile = "simulated.quant")
```

Arguments

`ANALYSIS_NAME`	A string indicating the name of the analysis/pipeline. Will be used in filenames, plot titles, etc.
`txfile`	The path to the transcript-level RNAseq expression matrix. If not given, will look for relevant files in the working directory. The expression matrix should have gene symbols in the first column/row.names, and sample names (either 's1', 's2', etc., or '1' '2', etc..) as column headers.
`qt`	A string indicating the unit of the expression matrix (either "FPKM", "TPM" or "COUNTS").

`normMethod`	The normalization method to use (see `donorm`). Defaults to 'TMM'.
`uniquelyMappableLengths`	Logical, whether to use uniquely mappable length, rather than full length, for FPKM calculation. Default TRUE.
`requireAll`	logical, whether all samples are required to proceed.

Value

Nothing, but produces many files in the working directory...

`compareSpikeinDEcalls` *compareSpikeinDEcalls*

Description

Benchmarks a list of differential expression calls between the two spike-in mixes.

Usage

```
compareSpikeinDEcalls(tests, thres = 0.01, colors = NULL)
```

Arguments

`tests`	A list of tests, each element of which should be a data.frame with a "p" column indicating the p-value under that test, and have ERCC IDs as row.names.
`thres`	Numeric value between 0 and 1, indicating the p-value treshold to use. Default 0.01.
`colors`	A vector of colors to be used (R arbitrary colors if NULL), or "greys" to use different tones of grey.

Value

A layout of two plots, with ROC curves on the left and a barplot of accuracy measurements on the right.

`compareWithNanostring` *Comparison with Nanostring*

Description

Loads and processes the RNAseq data before comparing it with the Nanostring quantification. Will produce a number of benchmarking plots and files in the current working directory.

Usage

```
compareWithNanostring(ANALYSIS_NAME, rnaseq = NULL, qt, normMethod = NULL)
```

Arguments

`ANALYSIS_NAME`	A string indicating the name of the analysis/pipeline. Will be used in filenames, plot titles, etc.
`rnaseq`	The path to the gene-level RNAseq expression matrix. If not given, will look for relevant files in the working directory. The expression matrix should have refseq id or gene symbols in the first column/row.names, and sample names (e.g. 'AJ80') as column headers.
`qt`	A string indicating the unit of the expression matrix (either "FPKM", "TPM" or "COUNTS").
`normMethod`	The normalization method to use (see `donorm`). Defaults to 'housekeeping' for gene-level, and 'TMM' for transcript-level.

Value

Nothing, but produces many files in the working directory...

Examples

```
# first we create a directory and put the example quantification file in it:
data(exampledata)
dir.create("example")
write.table(exampleGeneLevel,"w12.genes.quant",sep="\t",quote=F)
# then we run the function, giving a name to the analysis,
# specifying the file and type of quantification:
compareWithNanostring("tophat.featureCount", "w12.genes.quant", qt="COUNTS")
```

`compareWithPCR` *Gene-level comparison with RT-qPCR data*

Description

Loads and processes the RNAseq data before comparing it with the RT-qPCR quantification. Will produce some benchmarking plots and files in the current working directory.

Usage

```
compareWithPCR(ANALYSIS_NAME, rnaseq = NULL, qt)
```

Arguments

`ANALYSIS_NAME`	A string indicating the name of the analysis/pipeline. Will be used in filenames, plot titles, etc.
`rnaseq`	The path to the gene-level RNAseq expression matrix. If not given, will look for relevant files in the working directory. The expression matrix should have gene symbols in the first column/row.names, and sample names (e.g. 'AJ80', etc) as column headers.
`qt`	A string indicating the unit of the expression matrix (either "FPKM", "TPM" or "COUNTS").

Value

Nothing, but produces many files in the working directory...

`convertTx2Genes` *Summarizes transcript quantifications to the gene level*

Description

Summarizes transcript quantifications to the gene level by summing transcript values belonging to the same gene.

Usage

```
convertTx2Genes(tx)
```

Arguments

`tx`	A data.frame of transcript values, with refseq IDs as row.names.

Value

A data.frame of gene values.

`counts2fpkm` *Convert counts to FPKM values*

Description

Convert fragment counts to FPKM values, using the sum as library size. By default, effective length is used, set mean.frag.size to 0 to use total (given) length.

Usage

```
counts2fpkm(counts, lengths, mean.frag.size = 220)
```

Arguments

`counts`	A numeric vector of counts (for one sample).
`lengths`	A numeric vector of same length as 'counts', indicating the length of each transcript.
`mean.frag.size`	A numeric value indicating the mean fragment size (default 220) used for effective length calculation.

Value

A numeric vector of the corresponding FPKM values.

`counts2fpkmWrapper` *Convert counts matrix to FPKM matrix*

Description

Convert fragment counts to FPKM values, using the column sums as library sizes and effective length (see `counts2fpkm`).

Usage

```
counts2fpkmWrapper(r, level = "transcript", uniquelyMappableLengths = NULL)
```

Arguments

`r`	A numeric matrix or data.frame with samples as columns and genes/transcripts as rows.
`level`	Character, either "gene" or "tx", used to fetch the right lengths.
`uniquelyMappableLengths`	Whether to use the uniquely mappable lengths for transcripts (default TRUE).

Value

A matrix/data.frame of the corresponding FPKM values.

`deNanostring` *deNanostring*

Description

Runs a differential expression analysis and compares it to a log-t-test performed on the nanostring data.

Usage

```
deNanostring(rnaseq = NULL, method = "edgeR", norm = "TMM",
  quantification = "", threshold = 0.01)
```

Arguments

`rnaseq`	Either the (gene-level) count matrix, or a character indicating the location of the file.
`method`	The differential expression method to use. Either 'edgeR', 'DESeq', 'DESeq2', 'EBSeq', 'voom', 't' (t-test), or 'logt' (t-test on log-transformed values).
`norm`	The normalization method to use (either "linear", or any of the methods supported by edgeR's `calcNormFactors`. Defaults to 'TMM'.
`quantification`	A string indicating the quantification that was used to produce the expression matrix (use for labeling)
`threshold`	The p-value threshold applied on the Nanostring data to identify whether a gene is differentially-expressed or not.

Value

A deNanostring.compare plot

Examples

```
data(exampledata)
deNanostring(exampleGeneLevel, method="edgeR", norm="TMM",
  quantification="Tophat-featureCounts")
```

---

deNanostring.compare *deNanostring.compare*

Description

Compares the p-values and estimated foldchanges from a DEA analysis to a log-t-test performed on the nanostring data.

Usage

```
deNanostring.compare(results, method, norm, quantification = "",
  threshold = 0.01)
```

Arguments

`results`	A dataframe as produced by the `deNanostring` function, with gene symbols as row names, and containing the columns "p" (p-value) and "log2FC".
`method`	The differential expression method used (for reporting only)
`norm`	The normalization method used (for reporting only)
`quantification`	The quantification used (for reporting only)
`threshold`	The p-value threshold applied on the Nanostring data to identify whether a gene is differentially-expressed or not.

Value

A layout with 3 plots

`deSpikein` *Runs a spike-in differential expression analysis.*

Description

Runs a differential expression analysis and compares it to real differences between spike-in mixes. Sleuth is handled in a different function (see `sleuthWrapper`) A warning is given if the specified/expected mix distribution does not match the observed one.

Usage

```
deSpikein(dat, method = "edgeR", norm = "TMM", quantification = "",
  homogenize.mixes = T, saveResults = F, savePlot = F, mix1 = NULL)
```

Arguments

`dat`	The counts matrix or data.frame, with gene symbols or transcript Refseq IDs as row.names, and sample names as column headers.
`method`	The differential expression method to use. Either 'edgeR', 'DESeq', 'DESeq2', 'EBSeq', 'voom', 't' (t-test), or 'logt' (t-test on log-transformed values). A string indicating the unit of the expression matrix (either "FPKM", "TPM" or "COUNTS").
`norm`	The normalization method to use (either "linear", or any of the methods supported by edgeR's `calcNormFactors`. Defaults to 'TMM'.
`quantification`	A string indicating the name of the analysis/pipeline form which the quantification comes. Will be used in filenames, plot titles, etc.
`homogenize.mixes`	logical, whether the two spike-in mixes should be homogenized for the purpose of calculating normalization factors.
`saveResults`	Logical, whether to save the results of the DEA in the current working directory (default FALSE).
`savePlot`	Logical, whether to save the plot in the current working directory (default FALSE).
`mix1`	A character vector indicating the column names of ‘dat‘ that have been spiked with mix 1. If you are using the SEQC data or the dataset at the basis of this package, leave this to NULL.

Value

A data.frame with the results of the differential expression analysis, as well as a plot.

Examples

```
data(exampledata)
res <- deSpikein(exampleGeneLevel, method="edgeR", norm="TMM",
  quantification="Tophat-featureCounts")
```

`deSpikein.compare` *deSpikein.compare*

Description

A wrapper to create spike-in DEA benchmarking plots

Usage

```
deSpikein.compare(d, quantification, norm = "", method = "")
```

Arguments

`d`	A data.frame with at least the columns "p" (for p-value) and "log2FC", as produced in the `differentialExpression` function.
`quantification`	Name of the pipeline that produced the underlying quantification, used in plot titles.
`norm`	Normalization method, used in plot titles.
`method`	DEA method, used in plot titles.

Value

Nothing.

`donorm` *normalizes a dataset*

Description

Wrapper to normalize a dataset, using the given method.

Usage

```
donorm(dataset, method = "TMM")
```

Arguments

`dataset`	is a data.frame
`method`	is a character string of either 'linear', 'housekeeping', 'quantile', or any method supported by edgeR's `calcNormFactors`.

Value

The normalized data.frame.

Examples

```
donorm(matrix(1:12,nrow=4),"linear")
```

`fc2mean` *foldchange-to-the-mean*

Description

Returns the foldchange to the row's mean for each value.

Usage

```
fc2mean(x)
```

Arguments

`x`	a numeric matrix or data.frame.

Value

The matrix of corresponding foldchanges to the row's mean.

Examples

```
fc2mean(matrix(1:12,nrow=3))
```

`fcdev` *Plots deviation from real foldchange by p-value*

Description

Plots deviation from real foldchange by p-value

Usage

```
fcdev(d, real.log2fc, title = "")
```

Arguments

`d`	A data.frame as produced in the `differentialExpression` function.
`real.log2fc`	A vector of length nrow(d) containing the real foldchanges.
`title`	Plot title.

Value

Nothing.

`foldchange` *foldchange*

Description

Returns the foldchange, or the non-0 value +1 (or its inverse) if one of the values is 0, or 1/NA (as specified) if both are zero.

Usage

```
foldchange(x, y, na.fc = NA)
```

Arguments

`x`	a numeric vector
`y`	a numeric vector of same length as x.
`na.fc`	the value to return is both x and y are 0

Value

A vector of same length as x containing the foldchange of y/x

Examples

```
foldchange( c(5,Inf,0,6,0), c(0,1,2,3,0) )
```

`fpkm2tpm` *Convert FPKM values to TPM values*

Description

Convert FPKM values to transcripts per million (TPM).

Usage

```
fpkm2tpm(fpkm)
```

Arguments

`fpkm`	A numeric vector of fpkm values.

Value

A numeric vector of the corresponding TPM values.

`getLinearNormalizers` *getLinearNormalizers*

Description

Returns linear scale factors for each column of the dataset, using the first column as a reference and fitting a linear model between each other colum and the first.

Usage

```
getLinearNormalizers(dataset, tryrobust = F)
```

Arguments

`dataset`	a numeric data.frame or matrix.
`tryrobust`	logical, whether to try to fit a robust linear model instead of a normal one.

Value

A vector of length ncol(dataset) containing the scale factors of each column of dataset.

Examples

```
getLinearNormalizers(matrix(1:12,nrow=3))
```

`getLogFC` *Retrieves the log2(foldchange)*

Description

Retrieves the log2(foldchange).

Usage

```
getLogFC(x, groups)
```

Arguments

`x`	a numeric vector.
`groups`	a vector of same length as x, indicating to which of two groups each value belongs. If more than two groups, only the first two are used.

Value

Returns the log2 foldchange, or the log2 of the largest mean if one of the mean is 0, or 0 if both means are 0.

`homomixes` *Homogenizes the two spike-in mixes.*

Description

Divides spike-ins quantifications by their expected foldchange, thereby making all samples in principle equal. This is used for the purpose of calculating normalization factors.

Usage

```
homomixes(sp, mix1 = NULL)
```

Arguments

`sp`	Expression matrix or data.frame, with spike-in IDs as row.names.
`mix1`	The column headers of the samples that were spiked with mix1 (optional). This is detected automatically for the datasets included in this study.

Value

A data.frame including only the spike-ins, and with the two mixes 'homogenized'.

`isCleanData` *isCleanData*

Description

Returns which couples of x and y values are clean (non-missing, non-infinite and non-NaN)

Usage

```
isCleanData(x, y)
```

Arguments

`x`	a vector
`y`	a vector of same length as x

Value

indices that are 'clean' (non-missing, non-infinite and non-NaN) in both x and y.

Examples

```
isCleanData(c(2,NA,3),c(0,5,NA))
```

`maketrans` *Make transparent*

Description

Adds transparency to a color.

Usage

```
maketrans(tcol, alpha = 100)
```

Arguments

`alpha`	a numeric value between 0 and 255 indicating the degree of transparency.
`x`	a color specification (see `col2rgb`

Value

A color code.

Examples

```
maketrans("blue")
```

`MdAE` *Median absolute error*

Description

Returns the median absolute error between vectors x and y.

Usage

```
MdAE(x, y)
```

Arguments

`x`	a numeric vector
`y`	a numeric vector of same length as x.

Value

The median absolute error between x and y.

Examples

```
MdAE(1:3,jitter(1:3))
```

`mROC` *mROC*

Description

Plots superposed ROC curves.

Usage

```
mROC(tests, sig, show.AUC = T, thres = 0.01, lwd = 2, rounding = 3,
  na.rm = T, colors = NULL, ...)
```

Arguments

`tests`	A list of tests, each element of which should be a data.frame with a "p" column indicating the p-value under that test.
`sig`	A logical vector indicating whether each row of elements in ’tests’ is actually differentially-expressed (TRUE) or not. If this is a named vector, the names will be used to order the rows in ’tests’, otherwise they are assumed to be ordered in the same way.
`show.AUC`	Logical; whether to show the area under the curve in the figure’s legend. Default TRUE.
`thres`	Numeric value between 0 and 1, indicating the p-value treshold to be plotted on each curve. Set to NULL to plot no threshold.
`lwd`	Line width, passed to the plotting functions. Default 2.
`rounding`	The rounding (number of digits) of log-transformed p-values at which to plot (default 3). Increasing this number will increase the resolution of the curve, but also increase execution time.
`colors`	A vector of colors for the different tests. If NULL (default), R basic colors are used.
`...`	Any further argument passed to the initial plot function.

`myc` *Plots sensitivity and specificity by p-value*

Description

Plots sensitivity and specificity by p-value

Usage

```
myc(logp, sig, title = "", xlab = "log10(p-value)")
```

Arguments

`logp`	The log10(p-value) of each feature.
`sig`	A logical vector indicating whether each element in logp is actually differentially-expressed (TRUE) or not.
`xlab`	Plot xlab, defaults to "log10(p-value)".

Value

Nothing.

`package_sf_as_kal` *Convert sailfish results for one or more samples to kallisto HDF5*

Description

TAKEN FROM SAILFISH/SALMON DEVELOPERS COMMIT ON THE SLEUTH GIT and provided as is; users are advised to double-check for updates on how to use salmon/sailfish data with sleuth.

Usage

```
package_sf_as_kal(sf_dirs)
```

Arguments

`sf_dirs`	a character vector of length greater than one where each string points to a sailfish output directory

---

`plColorMap` *plColorMap*

Description

Maps colors onto numeric values.

Usage

```
plColorMap(x)
```

Arguments

`x`	a numeric vector.

Value

A vector of colors of the same length as x.

Examples

```
plColorMap(1:3)
```

`plGetModel` *Retrieves a linear model, if possible.*

Description

Retrieves a linear model, if possible.

Usage

```
plGetModel(x, y, no.intercept = T, tryrobust = F)
```

Arguments

`x`	a numeric vector containing the independent variable.
`y`	a numeric vector of same length as x, containing the dependent variable.
`no.intercept`	logical, indicating whether to disable the intersect in the linear model (defaults TRUE, i.e. models must go through the origin)
`tryrobust`	logical, indicating whether to try to fit a robust linear model (see `rlm`) instead of a normal one.

Value

Returns the linear model, if the fit was possible, NA otherwise.

`plTryRead` *plTryRead*

Description

Tries to find and read a file in the working directory, using an ordered list of regular expression and returning the first matching file.

Usage

```
plTryRead(patterns)
```

Arguments

`patterns`	a vector of regular expressions.

Value

The content of a tab-delimited file, if any matches the set of expressions, otherwise NULL.

Examples

```
plTryRead(c("test"))
```

`posplot` *Positives plot*

Description

Produces a 'positives plot', i.e. a plot analogous to the ROC curve, showing the positives across conditions against the positives within condition, across a significance thresholds.

Usage

```
posplot(p, fp1, fp2 = NULL, subsamp = 200, pround = 2, add = F,
  xlab = "DEGs between replicates", ylab = "DEGs between conditions",
  main = "Positives plot", col = "black", xlim = NULL, ylim = NULL,
  lwd = 3, lty = 1, auc.plotted = F)
```

Arguments

`p`	A vector of p-values across conditions.
`fp1`	A vector of p-values within condition A.
`fp2`	An optional vector of p-values within condition B.
`subsamp`	The maximum number of datapoints to plot. Default 200.
`pround`	Rounding of the log10 p-values (default 2 digits). Increasing this will increase the precision of the curve, at a speed in cost.
`add`	Whether to add the data series on top of the current graph. Default FALSE (produces a new plot).
`xlab`	Passed to the plot function.
`ylab`	Passed to the plot function.
`main`	Passed to the plot function.
`col`	Passed to the plot function, default black.
`xlim`	Passed to the plot function.
`ylim`	Passed to the plot function.
`lwd`	Passed to the plot function, default 3.
`lty`	Passed to the plot function, default 1.
`auc.plotted`	Whether to compute the area under the plotted curve (i.e. within plotting limits) instead of that of the full curve (default FALSE).

Value

Produces a plot and returns the area under the curve.

`pval2fc` *Plots p-values according to real foldchange.*

Description

Plots p-values according to real foldchange.

Usage

```
pval2fc(d, real.log2fc, jitter.arrows = T, title = "")
```

Arguments

`d`	A data.frame as produced in the `differentialExpression` function.
`real.log2fc`	A vector of length nrow(d) containing the real foldchanges.
`jitter.arrows`	Logical, whether the position of arrows outside the plotting range should be jittered to better see their number.
`title`	Plot title.

Value

Nothing.

`ROC` *ROC*

Description

Plots the ROC curve.

Usage

```
ROC(p, sig, main = NULL, rounding = 5)
```

Arguments

`p`	The p-value.
`sig`	A logical vector indicating whether each element in ’p’ is actually differentially-expressed (TRUE) or not.
`main`	Plot title.
`rounding`	The rounding (number of digits) of log-transformed p-values at which to plot (default 3). Increasing this number will increase the resolution of the curve, but also increase execution time.

Value

Nothing.

`runTests` *Runs a differential expression analysis.*

Description

Runs a differential expression analysis using the selected method. This is used by the `differentialExpression` function. This is simply a wrapper to make sure that the different analysis methods use the same normalization method and return results in the same format.

Usage

```
runTests(data, groups, norm = "TMM", de = "edgeR", homogenize.mixes = T)
```

Arguments

`data`	The expression matrix or data.frame, with gene symbols or transcript Refseq IDs as row.names.
`groups`	A logical vector (or coercible to logical) of length ncol(data), indicating to which group each sample (i.e. column in 'data') belongs. There can be only two groups.
`norm`	The normalization method to use (either "linear", or any of the methods supported by edgeR's `calcNormFactors`. Defaults to 'TMM'.
`de`	The differential expression method to use. Either 'edgeR', 'DESeq', 'DESeq2', 'EBSeq', 'voom', 't' (t-test), or 'logt' (t-test on log-transformed values). A string indicating the unit of the expression matrix (either "FPKM", "TPM" or "COUNTS").
`homogenize.mixes`	logical, whether the two spike-in mixes should be homogenized for the purpose of calculating normalization factors.

Value

A data.frame with the results of the differential expression analysis.

`seqc.diff` *seqc.diff*

Description

Runs a comparison of differential expression calls on the SEQC data, comparing calls across conditions to calls within conditions. The assumption here (and made by the SEQC consortium) is that differential expression across replicates are false positives. See `seqc.diff.example` for an example usage.

Usage

```
seqc.diff(e = NULL, norm = "TMM", de = "edgeR", site = "BGI",
  between.groups = 1:5, inner.groups = list(c(1, 2, 3), c(4, 5)),
  do.plot = T)
```

Arguments

`e`	The count matrix or data.frame. Anything can be used as row names (e.g. gene symbols, transcript ids...). If NULL, the 'site' parameter is used to fetch the quantification from the seqc package (if installed).
`norm`	The normalization method to use (see `donorm`). Defaults to 'TMM'.
`de`	The differential expression method to use. Either 'edgeR', 'DESeq', 'DESeq2', 'EBSeq', 'voom', 't' (t-test), or 'logt' (t-test on log-transformed values).
`site`	If 'e' is NULL, data from the specified sequencing site will be fetched from the seqc package. Default BGI.
`between.groups`	A vector of integers from 1 to 5 indicating the replicates to be used for comparison across conditions. Default 1:5.
`inner.groups`	A list of two vectors, each containing the replicates to be used for comparison inside each condition. Default list(c(1,2,3),c(4,5)).
`do.plot`	Logical, whether to produce Positives plots (see posplot). Default TRUE.

Value

A list of differential expression calls.

`seqc.diff.example` *seqc.diff.example*

Description

Compares differential expression analysis (DEA) methods using the `diff.seqc` function, and produces ROC-like plots.

Usage

```
seqc.diff.example(e = NULL, site = "BGI", tests = c("edgeR", "voom",
  "DESeq2"), between.groups = 1:5, inner.groups = list(c(1, 2, 3), c(4, 5)),
  do.plot = T, returnData = F)
```

Arguments

`e`	The count matrix or data.frame. Anything can be used as row names (e.g. gene symbols, transcript ids...). If NULL, the 'site' parameter is used to fetch the quantification from the seqc package (if installed).
`site`	If 'e' is NULL, data from the specified sequencing site will be fetched from the seqc package. Default BGI.
`tests`	A vector of the DEA methods to be compared.
`between.groups`	A vector of integers from 1 to 5 indicating the replicates to be used for comparison across conditions. Default 1:5.
`inner.groups`	A list of two vectors, each containing the replicates to be used for comparison inside each condition. Default list(c(1,2,3),c(4,5)).
`do.plot`	Logical, whether to produce Positives plots (see `posplot`). Default TRUE.
`returnData`	Logical, whether to return the resulting data.

Value

Produces a plot, and returns a list if returnData=T.

`seqc.diff.plot` *seqc.diff.plot*

Description

Produces a combination of 'positives plot' (see `posplot`) for a list of differential expression calls on the SEQC data. The x/ylim parameters control the 'zoomed plots'; the default settings are good for gene-level using all replicates. For gene-level with only 3 samples/group, use: xlim=c(0,60), ylimAB=c(5000,15500), ylimCD=c(2000,12000) For transcript-level with all samples, use: xlim=c(0,250), ylimAB=c(10000,24000), ylimCD=c(4000,17000) For transcript-level with only 3 samples/group, use: xlim=c(0,200), ylimAB=c(0,23000), ylimCD=c(0,14000)

Usage

```
seqc.diff.plot(ps, xlim = c(0, 60), ylimAB = c(15000, 19000),
  ylimCD = c(9000, 15000))
```

Arguments

`ps`	A list with softwares as names, and lists as elements, as produced by the 'diff.seqc.example' function. Each sublist contains the results of comparisons between A vs B, C vs D, and within each group.
`xlim`	x coordinates for the zoomed plot.
`ylimAB`	y coordinates for the AvsB zoomed plot.
`ylimCD`	y coordinates for the CvsD zoomed plot.

Value

Produces 4 plots and returns a list of accuracy values for each test

---

`seqc.diff.sleuth` *seqc.diff.sleuth*

Description

Sleuth wrapper for the analysis of SEQC data.

Usage

```
seqc.diff.sleuth(folders, between.groups = 1:5, inner.groups = list(c(1, 2,
  3), c(4, 5)), do.plot = T)
```

Arguments

`folders`	Folders containing the quantification and bootstraps. The folders should be named "A_1" and so on.
`between.groups`	A vector of integers from 1 to 5 indicating the replicates to be used for comparison across conditions. Default 1:5.
`inner.groups`	A list of two vectors, each containing the replicates to be used for comparison inside each condition. Default list(c(1,2,3),c(4,5)).
`do.plot`	Logical, whether to produce Positives plots (see `posplot`). Default TRUE.

Value

A list of differential expression calls.

`seqc.prepareData` *seqc.prepareData*

Description

Fetches and prepare SEQC data using the `seqc` package.

Usage

```
seqc.prepareData(site = "BGI", removeERCC = T)
```

Arguments

`site`	Sequencing site, default BGI.
`removeERCC`	Logical; whether to remove the spike-ins (default TRUE)

Value

A data.frame of counts, with gene symbols as row.names.

`sf_to_hdf5` *Convert sailfish results for one sample to kallisto HDF5*

Description

TAKEN FROM SAILFISH/SALMON DEVELOPERS COMMIT ON THE SLEUTH GIT and provided as is; users are advised to double-check for updates on how to use salmon/sailfish data with sleuth.

Usage

```
sf_to_hdf5(sf_dir)
```

Arguments

`sf_dir`	path to a sailfish output directory

simStats *Writes a table of metrics comparing two vectors*

Description

Writes a table of metrics comparing two vectors

Usage

```
simStats(x, y, ANALYSIS_NAME, fname, clean = T, norm = F, incZeros = F)
```

Arguments

x	A numeric vector of observed values.
y	A numeric vector of expected values (same length as 'x').
ANALYSIS_NAME	The name of the analysis pipeline
fname	The filename for saving the metrics
clean	Logical, whether to clean the data (default TRUE), see isCleanData.
norm	Logical, whether to linearly normalize the two vectors before comparison (default FALSE).
incZeros	Logical, whether to keep 0 values (default TRUE).

Value

Nothing.

sleuthWrapper *Runs a spike-in differential expression analysis using sleuth.*

Description

Runs a differential expression analysis using sleuth and compares it to real differences between spike-in mixes. Currently only for the 12-samples dataset.

Usage

```
sleuthWrapper(name, folders = NULL, norm = "TMM", savePlot = F)
```

Arguments

name	A string indicating the name of the analysis/pipeline form which the quantification comes. Will be used in filenames, plot titles, etc.
norm	The normalization method to use (either "linear", or any of the methods supported by edgeR's calcNormFactors. Defaults to 'TMM'.
savePlot	Logical, whether to save the plot in the current working directory (default TRUE).
folder	A character vector containing the path to the Salmon/Sailfish/Kallisto quantification folders. If missing, will look for names matching "AJ" in the current working directory.

Value

A data.frame with the results of the differential expression analysis, as well as a plot.

`sleuth_prep_sailfish` *sleuth_prep_sailfish*

Description

Constructor for a 'sleuth' object using sailfish data

Usage

```
sleuth_prep_sailfish(sf_dirs, sample_to_covariates, full_model,
  filter_fun = basic_filter, target_mapping = NULL, max_bootstrap = NULL,
  ...)
```

Arguments

`sf_dirs`	a character vector of length greater than one where each string points to a sailfish output directory

Details

Converts sailfish results to kallisto HDF5 format, then return the results of `sleuth_prep` on the converted data. TAKEN FROM SAILFISH/SALMON DEVELOPERS COMMIT ON THE SLEUTH GIT and provided as is; users are advised to double-check for updates on how to use salmon/sailfish data with sleuth.

See `sleuth_prep` for parameters other than sf_dirs.

`zscore` *z-score*

Description

Wrapper to calculate the z-score

Usage

```
zscore(x)
```

Arguments

`x`	a numeric matrix or data.frame.

Value

The matrix of corresponding row z-scores.

Examples

```
zscore(matrix(1:12,nrow=3))
```

Index