Package 'noisyr'

July 22, 2025

Type Package

Title Noise Quantification in High Throughput Sequencing Output

Version 1.0.0

Maintainer Ilias Moutsopoulos <im383@cam.ac.uk>

Description Quantifies and removes technical noise from high-throughput sequencing data. Two approaches are used, one based on the count matrix, and one using the alignment BAM files directly. Contains several options for every step of the process, as well as tools to quality check and assess the stability of output.

Depends R (>= 3.1.2)

Imports utils, grDevices, tibble, dplyr, magrittr, ggplot2, preprocessCore, IRanges, GenomicRanges, Rsamtools, philentropy, doParallel, foreach

Suggests testthat, roxygen2, knitr, rmarkdown

License GPL-2

Encoding UTF-8

URL https://github.com/Core-Bioinformatics/noisyR

BugReports https://github.com/Core-Bioinformatics/noisyR/issues

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Ilias Moutsopoulos [aut, cre],
Irina Mohorianu [aut, ctb],
Hajk-Georg Drost [ctb],
Elze Lauzikaite [ctb]

Repository CRAN

Date/Publication 2021-04-16 10:20:02 UTC

Contents

`calculate_expression_profile`
Calculate the expression profile of a gene

Description

This function calculates the expression profile of an exon in a selection of BAM files. The expression profile is defined as the number of reads overlapping with each position of the exon's transcript.

Usage

```
calculate_expression_profile(
  gene,
  bams,
  unique.only = TRUE,
  mapq.unique = c(50, 255),
  slack = 200
)
```

Arguments

`gene`	The exon for which the expression profile is calculated; this should be a row from the tibble generated by `cast_gtf_to_genes`; for a manual input, a tibble with 1 row and named columns (seqid, start, end) would be needed
`bams`	a vector of paths to the BAM files from which the profile is extracted

unique.only	whether only uniquely mapped reads should contribute to the profile; default is TRUE
mapq.unique	The values of the mapping quality field in the BAM file that corresponds to uniquely mapped reads; by default, values of 50 and 255 are used as these correspond to the most popular aligners, but an adjustment might be needed
slack	slack needs to be >=readLength, adjust for efficiency; the default is 200, as it is higher than most modern sequencing experiments

Value

The function outputs a list: the first element is a matrix of expression profiles. Rows correspond to positions in the exon transcript and each column corresponds to an input BAM file. Each read is counted for all the positions with which it overlaps (so a read of length 100 that completely overlaps with the exon would be counted for all 100 positions). The second list element is a vector of raw expression of the gene in the different BAM files

Examples

```
bams <- rep(system.file("extdata", "ex1.bam", package="Rsamtools", mustWork=TRUE), 2)
genes <- data.frame("id" = 1:2,
                    "gene_id" = c("gene1", "gene2"),
                    "seqid" = c("seq1", "seq2"),
                    "start" = 1,
                    "end" = 100)
profile <- calculate_expression_profile(
  gene = genes[1,],
  bams = bams,
  mapq.unique = 99
)

ggplot2::ggplot(tibble::tibble(y = profile$profile[,1],
                               x = seq_along(y))) +
ggplot2::geom_bar(ggplot2::aes(x, y), stat = "identity") +
ggplot2::theme_minimal()
```

---

calculate_expression_similarity_counts

Calcualate the expression levels and expression levels similarity matrices using the count matrix

---

Description

This function generates an average similarity (correlation/inverse distance) coefficient for every sliding window, for each sample in the expression matrix. That is done by comparing the distribution of genes in each window across samples.

Usage

```
calculate_expression_similarity_counts(
  expression.matrix,
  similarity.measure = "correlation_pearson",
  n.elements.per.window = NULL,
  n.step = NULL,
  n.step.fraction = 0.05,
  ...
)
```

Arguments

`expression.matrix`	the expression matrix, can be normalized or not
`similarity.measure`	one of the correlation or distance metrics to be used, defaults to pearson correlation; list of all methods in `get_methods_correlation_distance`
`n.elements.per.window`	number of elements to have in a window, default 10% of the number of rows
`n.step`	step size to slide across, default 1% of n.elements.per.window
`n.step.fraction`	an alternative way to specify the step size, as a fraction of the window length; default is 5%
`...`	arguments passed on to other methods

Value

A list with three elements: the first element is the expression matrix, as supplied; the other two are the expression levels matrix and expression levels similarity matrix; they have the same # of columns as the expression matrix, and n.elements.per.window * n.step rows.

See Also

`calculate_expression_similarity_transcript`

Examples

```
calculate_expression_similarity_counts(
    expression.matrix = matrix(1:100, ncol = 5),
    similarity.measure = "correlation_pearson",
    n.elements.per.window = 3)
```

`calculate_expression_similarity_transcript` *Calcualte the distance matrices using the BAM files*

Description

This function generates an average correlation/distance coefficient for every exon present in the BAM files. This is done by calculating the point-to-point correlation/distance of the distribution of reads across the transcript of each exon and comparing it across samples. The reason why exons are used instead of full length genes is that long intronic regions artificially increase the correlation since there is consistently no expression there, across samples. The user has the option to use genes instead, by running `cast_gtf_to_genes` separately, with non default parameters.

Usage

```
calculate_expression_similarity_transcript(
  bams = NULL,
  path.bams = ".",
  genes = NULL,
  path.gtf = list.files(".", pattern = "\\.g[tf]f$"),
  expression.matrix = NULL,
  subsample.genes = FALSE,
  make.index = FALSE,
  unique.only = TRUE,
  mapq.unique = 255,
  slack = 200,
  similarity.measure = "correlation_pearson",
  save.image.every.1000 = FALSE,
  ncores = 1,
  ...
)
```

Arguments

`bams`, `path.bams`	either a path to the directory where the BAM files are or a vector of paths to each individual file; if a path is specified, it extracts all files that end in .bam; looks in the working directory by default
`genes`	a tibble of the exons extracted from the gtf file; this is meant for speed if the output of `cast_gtf_to_genes` is already generated, or if the user wants to only calculate similarity for a subset of exons
`path.gtf`	the path to the gtf/gff annotation file (only used if genes is not provided); if unspecified, looks for one in the working directory
`expression.matrix`	expression matrix; not necessary but is used to filter the gtf to fewer entries and for subsampling if subsample.genes=TRUE; if not provided, raw read counts per exon are extracted from the BAM files

`subsample.genes`
: logical, whether to subsample low abundance genes to decrease computational time; the first minimum of the distribution of abundances is calculated, and genes lower than it are subsampled to match the number of genes higher than it; the expression matrix needs to be provided for this calculation; a plot is generated to show that minimum

`make.index`
: whether a BAM index should be generated; if this is FALSE (the default) and no index exists, the function will exit with an error; the index needs to have the same name as each BAM file, but ending with .bam.bai

`unique.only`
: whether only uniquely mapped reads should contribute to the expression of a gene/exon; default is TRUE

`mapq.unique`
: The values of the mapping quality field in the BAM file that corresponds to uniquely mapped reads; by default, values of 255 are used as these correspond to the most popular aligners, but an adjustment might be needed; the mapq scores should be as follows: 255 for STAR, 60 for hisat2, 255 for bowtie in -k mode, 40 for bowtie2 default, 50 for tophat

`slack`
: slack needs to be >=readLength, adjust for efficiency; the default is 200, as it is higher than most modern sequencing experiments

`similarity.measure`
: one of the similarity metrics to be used, defaults to pearson correlation; currently, only correlation is supported

`save.image.every.1000`
: whether to save a workspace image after every 1000 exons are processed; default is FALSE

`ncores`
: Number of cores for parallel computation; defaults to sequential computation, but parallelisation is highly encouraged; it is set to detectCores() if higher

`...`
: arguments passed on to other methods

Value

A list with three elements: the first element is the expression matrix, as supplied or calculated; the other two are the expression levels matrix and expression levels similarity matrix; they have the same # of columns as the expression matrix, and as many rows as exons processed.

See Also

calculate_expression_similarity_counts

Examples

```
bams <- rep(system.file("extdata", "ex1.bam", package="Rsamtools", mustWork=TRUE), 2)
genes <- data.frame("id" = 1:2,
                    "gene_id" = c("gene1", "gene2"),
                    "seqid" = c("seq1", "seq2"),
                    "start" = 1,
                    "end" = 1600)
expression.summary <- calculate_expression_similarity_transcript(
  bams = bams,
```

```
  genes = genes,
  mapq.unique = 99
)
```

`calculate_first_minimum_density` *Function to find the first local minimum of the density of a vector*

Description

This function is used to estimate the first local minimum of the density of a vector. It is meant to be used on the distribution of expression of genes in a sample; since the distribution tails off, finding the global minimum is not appropriate. The plot option can be used to visualise the process.

Usage

```
calculate_first_minimum_density(
  mat,
  log.transform = TRUE,
  adjust = 2,
  makeplots = FALSE
)
```

Arguments

`mat`	matrix whose columns will be used; usually an expression matrix; it can also be a vector
`log.transform`	whether to log-transform the data before the density estimation; default is TRUE
`adjust`	adjust factor for the smoothing, passed to density(); default is 2
`makeplots`	a logical value of whether a plot with a vertical line on the minimum found should be printed for each column of the matrix.

Value

The function outputs a single value corresponding to the median of the minima calculated for each column of the matrix. floor() is taken as a conservative estimate

Examples

```
calculate_first_minimum_density(
  matrix(c(rep(0,100),rep(3,30),rep(10,50),12,13,15,20),ncol=1),
  log.transform=FALSE, makeplots=TRUE
)
```

`calculate_noise_threshold` *Function to calculate the noise threshold for a given expression matrix and parameters*

Description

This function is used to calculate the noise threshold for a given expression matrix. It uses as input an expression profile, or just an expression matrix for a simple calculation based on density. A variety of methods are available to obtain a noise threshold using an input similarity threshold.

Usage

```
calculate_noise_threshold(
  expression,
  similarity.threshold = 0.25,
  method.chosen = "Boxplot-IQR",
  binsize = 0.1,
  minimum.observations.per.bin = NULL,
  ...
)
```

Arguments

`expression`	either an expression summary (as calculated by `calculate_expression_similarity_counts` or `calculate_expression_similarity_transcript`), which should be a list with 3 slots: expression.matrix, expression.levels, expression.levels.similarity; alternatively, just an expression matrix; only density based methods are available for the latter case
`similarity.threshold`	similarity (correlation or inverse distance) threshold to be used to find corresponding noise threshold; the default, 0.25 is usually suitable for the Pearson correlation (the default similarity measure)
`method.chosen`	method to use to obtain a vector of noise thresholds, must be one of `get_methods_calculate_noise_thr` defaults to Boxplot-IQR
`binsize`	size of each bin in the boxplot methods; defaults to 0.1 (on a log-scale)
`minimum.observations.per.bin`	minumum number of observations allowed in each bin of the boxplot; if a bin has fewer observations, it is merged with the one to its left; default is calculated as: ceiling(number of observations / number of bins / 10)
`...`	arguments passed on to other methods

Value

The output is a vector of noise thresholds, the same length as the number of columns in the expression matrix, or a single value in the case of density based methods.

See Also

calculate_noise_threshold_method_statistics

Examples

```
expression.summary <- calculate_expression_similarity_counts(
    expression.matrix = matrix(1:100, ncol=5),
    method = "correlation_pearson",
    n.elements.per.window = 3)
calculate_noise_threshold(expression.summary)
```

---

calculate_noise_threshold_method_statistics

Function to tabulate statistics for different methods of calculating the noise threshold

---

Description

This function is used to tabulate and compare different combinations of similarity threshold and method to calculate the noise threshold for a given expression matrix.

Usage

```
calculate_noise_threshold_method_statistics(
  expression,
  similarity.threshold.sequence = 0.25,
  method.chosen.sequence = noisyr::get_methods_calculate_noise_threshold(),
  dump.stats = NULL,
  ...
)
```

Arguments

expression	either an expression summary (as calculated by calculate_expression_similarity_counts or calculate_expression_similarity_transcript), which should be a list with 3 slots: expression.matrix, expression.levels, expression.levels.similarity; alternatively, just an expression matrix; only density based methods are available for the latter case
similarity.threshold.sequence	similarity (correlation or inverse distance) threshold(s) to be used to find corresponding noise threshold; can be a single value or a numeric vector; the default, 0.25 is usually suitable for the Pearson correlation (the default similarity measure)
method.chosen.sequence	methods to use to calculate the noise thresholds, must be a subset of get_methods_calculate_noise_th defaults to all
dump.stats	name of csv to export different thresholds calculated (optional)
...	other arguments (for the boxplot methods) passed to calculate_noise_threshold

Value

A tibble containing information on noise thresholds calculated using the input similarity thresholds and methods (optionally written in a csv file). The columns list the chosen method and similarity threshold, the minimum, mean, coefficient of variation, and maximum of the noise thresholds, and all the noise thresholds concatenated as a string.

See Also

calculate_noise_threshold

Examples

```
expression.summary <- calculate_expression_similarity_counts(
    expression.matrix = matrix(1:100, ncol=5),
    method = "correlation_pearson",
    n.elements.per.window = 3)
calculate_noise_threshold_method_statistics(expression.summary)
```

---

cast_gtf_to_genes *Function to extract exon names and positions from a gtf file*

Description

This function is used to extract all exons and their positions in the genome from an input gtf file.

Usage

```
cast_gtf_to_genes(filename, feature = "exon", att_of_interest = "gene_id", ...)
```

Arguments

filename	path to the gtf file
feature	the feature type name to filter the feature (3rd) column of the gtf/gff file; default is exon
att_of_interest	the attribute to extract from the last column of the gtf/gff file; default in gene_id
...	arguments passed on to other methods

Value

A tibble of the ids, gene names, chromosomes, start and end positions of each exon found in the gtf file.

Examples

```
fl <- system.file("extdata", "example.gtf.gz", package="Rsamtools", mustWork=TRUE)
cast_gtf_to_genes(fl)
```

cast_matrix_to_numeric

Cast a matrix of any type to numeric

Description

Transforms values in the expression matrix to numeric, to make it compatible with the rest of the functions.

Usage

```
cast_matrix_to_numeric(expression.matrix)
```

Arguments

expression.matrix	The expression matrix (usually read from a file)

Value

The expression matrix transformed to numeric, preserving row and column names. Any values that are not coercible to numeric are replaced by 0.

Examples

```
cast_matrix_to_numeric(matrix(
    c(1, "2", 3.0, 4),
    ncol=2,
    dimnames=list(paste0("X", 1:2),
                  paste0("Y", 1:2))))
```

filter_genes_transcript

Function to filter the gene table for the transcript approach

Description

This function is used to filter the gene table (usually created with cast_gtf_to_genes), only keeping genes above the noise thresholds. It uses as input the gene table (usually containing individual exons), an expression matrix for each of these and a vector of abundance thresholds. This function is used internally by remove_noise_from_bams to determine which genes to retain.

Usage

```
filter_genes_transcript(
  genes,
  expression.matrix,
  noise.thresholds,
  filter.by = c("gene", "exon"),
  ...
)
```

Arguments

genes	a tibble of the exons extracted from the gtf file; (usually the the output of cast_gtf_to_genes)
expression.matrix	the expression matrix, usually calculated by calculate_expression_similarity_transcript
noise.thresholds	a vector of expression thresholds by sample
filter.by	Either "gene" (default) or "exon"; if filter.by="gene", a gene (as determined by its ENSEMBL id) is removed if and only if all of its exons are below the corresponding noise thresholds; if filter.by="exon", then each exon is individually removed if it is below the corresponding noise thresholds.
...	arguments passed on to other methods

Value

Returns a filtered tibble of exons, with the noise removed.

Examples

```
bams <- rep(system.file("extdata", "ex1.bam", package="Rsamtools", mustWork=TRUE), 2)
genes <- data.frame("id" = 1:2,
                    "gene_id" = c("gene1", "gene2"),
                    "seqid" = c("seq1", "seq2"),
                    "start" = 1,
                    "end" = 1600)
noise.thresholds <- c(0, 1)
expression.summary = calculate_expression_similarity_transcript(
  bams = bams,
  genes = genes,
  mapq.unique = 99
)
filter_genes_transcript(
    genes = genes,
    expression.matrix = expression.summary$expression.matrix,
    noise.thresholds = noise.thresholds,
)
```

get_methods_calculate_noise_threshold

Show the methods for calculating a noise threshold

Description

This function outputs the methods available for the calculation of the noise threshold. To be used as input in `calculate_noise_threshold`.

Usage

```
get_methods_calculate_noise_threshold()
```

Value

A character vector of options for the method.chosen arguement of `calculate_noise_threshold`

Examples

```
get_methods_calculate_noise_threshold()
```

get_methods_correlation_distance

Show the methods for calculating correlation or distance

Description

This function outputs the methods available for the calculation of the correlation or distance. The standard correlation methods use stats::cor and a wide variety of distance methods are available using the philentropy package. To be used as input in `calculate_expression_similarity_counts` or `calculate_expression_similarity_transcript`.

Usage

```
get_methods_correlation_distance(names = TRUE)
```

Arguments

`names`	whether to output names (default) or characterisation as similarity or dissimilarity (used internally to invert dissimilarity measures)

Value

A character vector of options for the method arguement of the similarity calculation; if names=FALSE, a vector of types (similarity/dissimilarity measure) of the same length

Examples

```
get_methods_correlation_distance()
```

`noisyr` *Run the noisyR pipeline*

Description

Calls one of noisyr_counts or noisyr_transcript, with the specified parameters. See the individual function documentation for more details and required arguments: `noisyr_counts`, `noisyr_transcript`

Usage

```
noisyr(approach.for.similarity.calculation = c("counts", "transcript"), ...)
```

Arguments

`approach.for.similarity.calculation`	which approach to use for the similarity calculation; defaults to counts
`...`	arguments to be passed on to noisyr_counts or noisyr_transcript; see their documentation for more details and required arguments

Value

For the counts approach, the denoised expression matrix. For the transcript approach, the numeric vector of noise thresholds per sample. For more details, see their respective documentation.

Examples

```
noisyr(approach.for.similarity.calculation = "counts",
       expression.matrix = matrix(1:100, ncol = 5))
```

`noisyr_counts` *Run the noisyR pipeline for the count matrix approach*

Description

Calls the functions to run each of the three steps of the pipeline (similarity calculation, noise quantification, noise removal), with the specified parameters. See the individual function documentation for more details and required arguments. Required steps: `calculate_expression_similarity_counts`, `calculate_noise_threshold`. `remove_noise_from_matrix`. Optional steps: `optimise_window_length`, `calculate_noise_threshold_method_statistics`

Usage

```
noisyr_counts(
  expression.matrix,
  n.elements.per.window = NULL,
  optimise.window.length.logical = FALSE,
  similarity.threshold = 0.25,
  method.chosen = "Boxplot-IQR",
  ...
)
```

Arguments

`expression.matrix`	the expression matrix used as input for the similarity calculation; this argument is required
`n.elements.per.window`	number of elements to have in a window passed to calculate_expression_similarity_counts(); default 10% of the number of rows
`optimise.window.length.logical`	whether to call optimise_window_length to try and optimise the value of n.elements.per.window
`similarity.threshold`, `method.chosen`	parameters passed on to `calculate_noise_threshold`; they can be single values or vectors; if they are vectors optimal values are computed by calling `calculate_noise_threshold_` and minimising the coefficient of variation across samples; all possible values for method.chosen can be viewed by `get_methods_calculate_noise_threshold`
`...`	arguments to be passed on to individual pipeline steps

Value

The denoised expression matrix.

See Also

`noisyr`, `noisyr_transcript`

Examples

```
noisyr_counts(
    expression.matrix = matrix(1:100, ncol = 5),
    similarity.measure = "correlation_pearson",
    n.elements.per.window = 3)
```

noisyr_transcript *Run the noisyR pipeline for the transcript approach*

Description

Calls the functions to run each of the three steps of the pipeline (similarity calculation, noise quantification, noise removal), with the specified parameters. See the individual function documentation for more details and required arguments. Required steps: calculate_expression_similarity_transcript, calculate_noise_threshold. remove_noise_from_bams. Optional steps: calculate_noise_threshold_method_stati

Usage

```
noisyr_transcript(
  bams = NULL,
  path.bams = ".",
  genes = NULL,
  path.gtf = list.files(".", pattern = "\\.g[tf]f$"),
  ncores = 1,
  similarity.threshold = 0.25,
  method.chosen = "Boxplot-IQR",
  ...
)
```

Arguments

bams, path.bams	either a path to the directory where the BAM files are or a vector of paths to each individual file; if a path is specified, it extracts all files that end in .bam; looks in the working directory by default
genes	a tibble of the exons extracted from the gtf file; this is meant for speed if the output of cast_gtf_to_genes is already generated, or if the user wants to only calculate similarity for a subset of exons
path.gtf	the path to the gtf/gff annotation file (only used if genes is not provided); if unspecified, looks for one in the working directory
ncores	Number of cores for parallel computation; defaults to sequential computation, but parallelisation is highly encouraged; it is set to detectCores() if higher
similarity.threshold, method.chosen	parameters passed on to calculate_noise_threshold; they can be single values or vectors; if they are vectors optimal values are computed by calling calculate_noise_threshold_ and minimising the coefficient of variation across samples; all possible values for method.chosen can be viewed by get_methods_calculate_noise_threshold; only boxplot based methods are accepted for the transcript approach due to the number of observations and high variance
...	arguments to be passed on to individual pipeline steps; see their documentation for more details and required arguments

Value

The denoised BAM files are created, as specified by the destination.files argument of remove_noise_from_bams()

See Also

noisyr, noisyr_counts

Examples

```
bams <- rep(system.file("extdata", "ex1.bam", package="Rsamtools", mustWork=TRUE), 2)
genes <- data.frame("id" = 1:2,
                    "gene_id" = c("gene1", "gene2"),
                    "seqid" = c("seq1", "seq2"),
                    "start" = 1,
                    "end" = 1600)
noisyr_transcript(
  bams = bams,
  genes = genes,
  destination.files = paste0(tempdir(), "/", basename(bams), ".noisefiltered.bam")
)
```

---

optimise_window_length

Optimise the elements per window for the count matrix approach

---

Description

This function optimises the number of elements per window that is used in calculate_expression_similarity_counts, by requiring the distribution of correlations/distances to stabilise to a uniform distribution. The Jensen-Shannon divergence is used to assess the stability.

Usage

```
optimise_window_length(
  expression.matrix,
  similarity.measure = "correlation_pearson",
  window.length.min = NULL,
  window.length.max = NULL,
  window.length.by = NULL,
  n.step.fraction = 0.05,
  iteration.number = 50,
  minimum.similar.windows = 3,
  save.plot = NULL
)
```

Arguments

`expression.matrix`
: expression matrix, can be normalized or not

`similarity.measure`
: one of the correlation or distance metrics to be used, defaults to pearson correlation; list of all methods in `get_methods_correlation_distance`

`window.length.min`, `window.length.max`, `window.length.by`
: definition of the parameter search space; default is between 1% and 33% of the number of rows in the expression matrix, incremented by 1%

`n.step.fraction`
: step size to slide across, as a fraction of the window length; default is 5%

`iteration.number`
: number of iterations for the subsampling and calculation of JSE; subsampling is needed because shorter windows have fewer points; default is 100

`minimum.similar.windows`
: number of windows that a window needs to be similar to (including itself) in order to be accepted as optimal; default is 3, but can be reduced to 2 if no optimum is found

`save.plot`
: name of the pdf in which to print the output plot showing the distribution of JSE by window; output to the console by default

Value

A single value of the optimal number of elements per window. If no optimal value was found, this function returns NULL.

Examples

```
optimise_window_length(
  matrix(1:100+runif(100), ncol=5, byrow=TRUE),
  window.length.min=3, window.length.max=5, iteration.number=5
)
```

`plot_expression_similarity` *Plot the similarity against expression levels*

Description

Creates the expression-similarity line and box plots for each sample.

Usage

```
plot_expression_similarity(
  expression.summary,
  sample.names = NULL,
  similarity.name = "Pearson correlation",
  log.transform = TRUE,
  min.y = NULL,
  max.y = NULL,
  smooth.span = 0.1,
  only.boxplot = FALSE,
  binsize = 1,
  last.together = 0,
  show.counts = TRUE,
  add.threshold = NULL,
  file.name = NULL
)
```

Arguments

expression.summary	list containing expression_levels and expression_levels_similarity matrices, as calculated by calculate_expression_similarity_counts or calculate_expression_similarity_t
sample.names	names for the plots, defaults to the column names of the expression matrix
similarity.name	similarity metric used (for the y-axis title)
log.transform	should the count matrix be log-transformed? If not, boxplot is skipped
min.y, max.y	limits for the y axis. If unset default to symmetric including all values in expression.levels.similarity; min is set to 0 if there are no negative values
smooth.span	span to be used for smoothing in the line plot; defaults to 0.1
only.boxplot	option to skip the line plot (usually a good idea if there are too many points and lines are too erratic); sets log.transform to TRUE
binsize	size of each bin in the boxplot; defaults to 0.5
last.together	groups observations so the highest abundance bin has at least this many
show.counts	whether to show how many observations are in each bin
add.threshold	adds a horizontal line at this value
file.name	name of pdf to output the plots; if not provided (default), no printing is done

Value

A list of all the plots (returned silently)

Examples

```
plots <- plot_expression_similarity(
  expression.summary=list(
    "expression.levels" = matrix(2^(10*seq(0,1,length.out=100))),
```

```
    "expression.levels.similarity" = matrix(seq(0,1,length.out=100)+(runif(100)/5))))
plots[[1]]
plots[[2]]
```

`remove_noise_from_bams` *Function to remove the noisy reads from the BAM files*

Description

This function is used to remove the noisy reads from the BAM files. It uses as input the BAM file names, a gene table (usually containing individual exons, made using `cast_gtf_to_genes`), an expression matrix for each of these genes and a vector of abundance thresholds.

Usage

```
remove_noise_from_bams(
  bams,
  genes,
  expression,
  noise.thresholds,
  destination.files = base::paste0(base::basename(bams), ".noisefiltered.bam"),
  filter.by = c("gene", "exon"),
  make.index = FALSE,
  unique.only = TRUE,
  mapq.unique = 255,
  ...
)
```

Arguments

`bams`	a character vector of the BAM file names
`genes`	a tibble of the exons extracted from the gtf file; (usually the the output of `cast_gtf_to_genes`)
`expression`	the expression matrix or expression summary list, as calculated by `calculate_expression_similarity`
`noise.thresholds`	a vector of expression thresholds by sample; must be the same length as the number of BAM files, or a singular value to be used as a fixed noise threshold
`destination.files`	names for the output denoised BAM files; by default the same as the original files, appended with ".noisefiltered.bam", but created in the working directory
`filter.by`	Either "gene" (default) or "exon"; if filter.by="gene", a gene is removed from all BAM files if and only if all of its exons are below the corresponding noise thresholds; if filter.by="exon", then each exon is individually removed (from all samples) if it is below the corresponding noise thresholds.

make.index	whether a BAM index should be generated; if this is FALSE (the default) and no index exists, the function will exit with an error; the index needs to have the same name as each BAM file, but ending with .bam.bai
unique.only	whether only uniquely mapped reads should contribute to the expression of a gene/exon; default is TRUE
mapq.unique	The values of the mapping quality field in the BAM file that corresponds to uniquely mapped reads; by default, values of 255 are used as these correspond to the most popular aligners, but an adjustment might be needed; the mapq scores should be as follows: 255 for STAR, 60 for hisat2, 255 for bowtie in -k mode, 40 for bowtie2 default, 50 for tophat
...	arguments passed on to other methods

Value

Returns a matrix of the same dims as the expression matrix, with the noise removed. This matrix has no entries remaining below the noise threshold.

See Also

remove_noise_from_matrix

Examples

```
bams <- rep(system.file("extdata", "ex1.bam", package="Rsamtools", mustWork=TRUE), 2)
genes <- data.frame("id" = 1:2,
                    "gene_id" = c("gene1", "gene2"),
                    "seqid" = c("seq1", "seq2"),
                    "start" = 1,
                    "end" = 1600)
noise.thresholds <- c(0, 1)
expression.summary = calculate_expression_similarity_transcript(
  bams = bams,
  genes = genes,
  mapq.unique = 99
)
remove_noise_from_bams(
    bams = bams,
    genes = genes,
    expression = expression.summary,
    noise.thresholds = noise.thresholds,
    destination.files = paste0(tempdir(), "/", basename(bams), ".noisefiltered.bam"),
    mapq.unique = 99
)
```

`remove_noise_from_matrix` *Function to remove the noisy reads from the expression matrix*

Description

This function is used to remove the noisy reads from the expression matrix. It uses as input a vector of abundance thresholds; all entries below the noise threshold are replaced with the noise threshold.

Usage

```
remove_noise_from_matrix(
  expression.matrix,
  noise.thresholds,
  add.threshold = TRUE,
  average.threshold = TRUE,
  remove.noisy.features = TRUE,
  export.csv = NULL,
  ...
)
```

Arguments

`expression.matrix`	the expression matrix
`noise.thresholds`	a vector of expression thresholds by sample; must be the same length as the number of columns of the expression matrix, or a singular value to be used as a fixed noise threshold
`add.threshold`	whether to add the noise threshold to all values in the expression matrix (default), or set entries below the threshold to the threshold
`average.threshold`	if TRUE (default), uses tthe average of the vector of thresholds across all samples; if FALSE, uses the thresholds as supplied
`remove.noisy.features`	logical, whether rows of the expression matrix that are fully under the noise threshold should be removed (default TRUE)
`export.csv`	option to write the matrix into a csv after the noise removal; should be NULL or the name of the output file
`...`	arguments passed on to other methods

Value

Returns the expression matrix with the noise removed. Under default parameters, the denoised matrix will have fewer rows than the input matrix and will have no entries remaining below the noise threshold.

See Also

remove_noise_from_bams

Examples

```
expression.matrix <- matrix(1:100, ncol=5)
noise.thresholds <- c(5,30,45,62,83)
remove_noise_from_matrix(
    expression.matrix = expression.matrix,
    noise.thresholds = noise.thresholds
)
```

Index