# Package 'noisyr'

January 6, 2021

**Type** Package

**Title** Noise Quantification in High Throughput Sequencing Output

**Version** 0.1.0

**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

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

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

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

**RoxygenNote** 7.1.1

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2021-01-06 09:50:02 UTC

# R topics documented:



---

`calculate_distance_matrices_counts` *Calcualte the distance matrices using the count matrix*

---

## Description

This function generates an average correlation/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_distance_matrices_counts(
  expression.matrix,
  method = "correlation_pearson",
  n.elements.per.window = NULL,
  nstep = NULL,
  nstep.frac = 0.05
)
```

## Arguments

| | |
|---|---|
| `expression.matrix` | expression matrix, can be normalized or not |
| `method` | 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 |
| `nstep` | step size to slide across, default 1% of n.elements.per.window |
| `nstep.frac` | an alternative way to specify the step size, as a fraction of the window length; default is 5% |

## Value

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

## Examples

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

---

# calculate_distance_matrices_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.

## Usage

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

## Arguments

`path.bams, 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

`path.gtf` the path to the gtf/gff annotation file; if unspecified, looks for one in the working directory

`genes` a tibble of the genes extracted from the gtf file; this is meant for speed if the output of cast_gtf_to_genes() is already generated

`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 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; 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

`uniqueOnly` 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 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

`method` one of the distance metrics to be used, defaults to pearson correlation CURRENTLY ONLY CORRELATIONS!

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

`ncores` Number of cores to use for parallel computation; needs doParallel installed; defaults to sequential computation

## Value

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

## 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)
```

```
obj <- calculate_distance_matrices_transcript(
  bams = bams,
  genes = genes,
  mapq.unique = 99
)
```

---

`calculate_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_profile(
  gene,
  bams,
  uniqueOnly = 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 |
| `uniqueOnly` | 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" = 1600)
profile <- calculate_profile(
  gene = genes[1,],
  bams = bams,
  mapq.unique = 99
)
```

---

`calculate_threshold_fixed_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 abundances 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 pprocess.

## Usage

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

---

`calculate_threshold_noise`

*Function to calculate the noise threshold for a given expression matrix*

---

## Description

This function is used to calculate the noise threshold for a given expression matrix. It uses as input a distance matrix and the corresponding abundance matrix. A variety of methods are available to obtain an abundance threshold using an input distance threshold.

## Usage

```
calculate_threshold_noise(
  expression.matrix,
  abn.matrix = NULL,
  dist.matrix = NULL,
  dist.thresh = 0.25,
  binsize = 0.1,
  dump.stats = NULL,
  method.chosen = NULL
)
```

## Arguments

| | |
|---|---|
| `expression.matrix` | expression matrix, should be the one used in calculate_distance_matrices() |
| `abn.matrix, dist.matrix` | the input distance and abundance matrices as calculated by calculate_distance_matrices(); if either is not supplied, only a fixed threshold is calculated based on the density |
| `dist.thresh` | correlation threshold to be used to find corresponding abundance threshold. The default, 0.25 is usually suitable for the Pearson correlation (the default method) |
| `binsize` | size of each bin in the boxplot methods; defaults to 0.1 (on a log-scale) |
| `dump.stats` | name of csv to export different thresholds calculated (optional) |
| `method.chosen` | method to use to obtain a single vector of thresholds, must be one of get_methods_calculate_noise_thresho if set, it skips all other methods; this is meant for speed, to be used internally or if rerunning an analysis and is not recommended as a first approach. |

## Value

Normal output is a tibble containing information on thresholds calculated using different methods (returned silently and optionally written in a csv file). If method.chosen is set to one of the methods in get_methods_calculate_noise_threshold(), then the output is a vector of noise thresholds, the same length as the number of columns in the expression matrix.

## Examples

```
obj <- calculate_distance_matrices_counts(
    expression.matrix = matrix(1:100, ncol=5),
    method="correlation_pearson",
    n.elements.per.window=3)
calculate_threshold_noise(obj$exp, obj$abn, obj$dist,
    method.chosen="Boxplot-IQR")
```

---

# 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")
```

## Arguments

| | |
|---|---|
| filename | path to the gtf file |
| feature | the feature to extract from the last column of the gtf/gff file; default is exon |

## Value

Returns a tibble of the ids, names, chromosomes, start and end positions of each exon found in the gtf file. If refGenome is installed, that is used for the reading and is faster, otherwise the gtf is read manually with a warning

## Examples

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

---

# cast_matrix_to_double *Cast a matrix of any type to double*

---

## Description

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

## Usage

```
cast_matrix_to_double(expression.matrix)
```

### Arguments

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

### Value

The expression matrix transformed to double, preserving row and column names

### Examples

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

---

## `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_threshold_noise().

### Usage

```
get_methods_calculate_noise_threshold()
```

### Value

A character vector of options for the method arguement of calculate_threshold_noise()

### 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_distance_matrices_*().

### 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 calculate_distance_matrices_*(); if names=FALSE, a vector of types (similarity/dissimilarity measure) of the same length

### Examples

```
get_methods_correlation_distance()
```

## 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_distance_matrices_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,
  method = "correlation_pearson",
  winlen.min = NULL,
  winlen.max = NULL,
  winlen.by = NULL,
```

```
  nstep.frac = 0.05,
  iternum = 1000,
  save.plot = NULL
)
```

## Arguments

| | |
|---|---|
| expression.matrix | expression matrix, can be normalized or not |
| method | one of the correlation or distance metrics to be used, defaults to pearson correlation; list of all methods in get_methods_correlation_distance() |
| winlen.min, winlen.max, winlen.by | definition of the parameter search space |
| nstep.frac | an alternative way to specify the step size, as a fraction of the window length; default is 5% |
| iternum | number of iterations for the subsampling and calculation of JSE; subsampling is needed because shorter windows have fewer points |
| save.plot | name of the pdf in which to print the output plot showing the distribution of JSE by window |

## Value

A single value of the optimal number of elements per window

## Examples

```
optimise_window_length(
matrix(1:100+runif(100), ncol=5, byrow=TRUE),
  winlen.min=3, winlen.max=5, iternum=5)
```

---

# plot_distance_abundance

*Plot the distance*

---

## Description

Creates the abundance-distance line and box plots for each sample.

## Usage

```
plot_distance_abundance(
  abn.matrix,
  dist.matrix,
  sample.names = paste("Sample", 1:ncol(abn.matrix)),
  distance.name = "Pearson correlation",
  log.transform = TRUE,
  min.y = NULL,
```

```
  max.y = NULL,
  smooth.span = 0.1,
  only.boxplot = FALSE,
  binsize = 0.5,
  last.together = 30,
  show.counts = TRUE,
  add.threshold = NULL,
  file.name = NULL
)
```

## Arguments

| | |
|---|---|
| abn.matrix, dist.matrix | abundance and distance matrix, as calculated by calculate_distance_matrices() |
| sample.names | names for the plots, enumerates the samples by default |
| distance.name | distance 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 dist.matrix; 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 (console by default) |

## Value

A list of all the plots (returned silently), which are also plotted to the console, or specified pdf file

## Examples

```
plot_distance_abundance(
  abn.matrix=matrix(2^(10*seq(0,1,length.out=100))),
  dist.matrix=matrix(seq(0,1,length.out=100)+(runif(100)/5)))
```

`remove_noise_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_matrix(
  expression.matrix,
  abn.thresh,
  add.thresh = TRUE,
  average.thresh = TRUE,
  remove.noisy.features = TRUE,
  export.csv = NULL
)
```

## Arguments

| | |
|---|---|
| `expression.matrix` | the expression matrix |
| `abn.thresh` | a vector of abundance thresholds; must be the same length as the number of columns of the expression matrix |
| `add.thresh` | whether to add the noise threshold to all values in the expression matrix (default), or set entries below the threshold to the threshold |
| `average.thresh` | 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. |

## 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.

## Examples

```
remove_noise_matrix(
    expression.matrix = matrix(1:100, ncol=5),
    abn.thresh=c(5,30,45,62,83))
```

`remove_noise_method` *Function to remove the noisy reads from the expression matrix using a specified method*

## Description

This function is used to remove the noisy reads from the expression matrix. It can use thresholds precalculated by calculate_threshold_noise(), if available. If a vector of thresholds needs to be input manually, use remove_noise_matrix() instead.

## Usage

```
remove_noise_method(
  expression.matrix,
  method.chosen = "Boxplot-IQR",
  stats.df = NULL,
  abn.matrix = NULL,
  dist.matrix = NULL,
  dist.thresh = 0.25,
  binsize = 0.1,
  add.thresh = TRUE,
  average.thresh = TRUE,
  remove.noisy.features = TRUE,
  export.csv = NULL
)
```

## Arguments

| | |
|---|---|
| `expression.matrix` | the expression matrix |
| `method.chosen` | the method to be used for the noise removal; must be one of get_methods_calculate_noise_threshold() |
| `stats.df` | a tibble, as output by calculate_threshold_noise(); if supplied and the chosen method appears in it, the corresponding threshold is used directly. |
| `abn.matrix, dist.matrix` | the input distance and abundance matrices as calculated by calculate_distance_matrices(); only needed if stats.df is not supplied; if either is not supplied, only a fixed threshold is calculated based on the density |
| `dist.thresh` | a distance threshold to be used if the noise thresholds are not pre-calculated; the default 0.25 is suitable for correlation measures |
| `binsize` | size of each bin in the boxplot methods; defaults to 0.1 (on a log-scale) |
| `add.thresh` | whether to add the noise threshold to all values in the expression matrix (default), or set entries below the threshold to the threshold |
| `average.thresh` | 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.

## Value

Returns a matrix of the same dims as the expression matrix, with the noise removed using the specified method. All entries below the noise threshold are replaced with the noise threshold.

## Examples

```
obj <- calculate_distance_matrices_counts(
    expression.matrix = matrix(1:100, ncol=5),
    method="correlation_pearson",
    n.elements.per.window=3)
remove_noise_method(
    expression.matrix=obj$exp,
    abn.matrix=obj$abn,
    dist.matrix=obj$dist)
```

# Index