Package 'riboWaltz'

February 12, 2019

Type Package

Title Optimization of ribosome P-site positioning in ribosome profiling data

Version 1.0.1

Description riboWaltz is an R package designed for the analysis of ribosome profiling (RiboSeq) data aimed at the identification of the P-site offset. The P-site offset (PO) is specified by the localization of the P-site of ribosomes within the fragments of the RNA (reads) resulting from RiboSeq assays. It is defined as the distance of the P-site from the two ends of the reads. Determining the PO is a crucial step for a variety of RiboSeq-based analyses such as verify the so-called 3-nt periodicity of ribosomes along the coding sequence, derive translation initiation and elongation rates and reveal new translational events in unannotated open reading frames and ncRNAs. riboWaltz performs accurate computation of the PO for all the lengths of reads from single or multiple samples, taking advantage from an original two-step algorithm. Moreover, riboWaltz provides the user a variety of graphical representations, laying the groundwork for further positional analyses and new biological discoveries.

License MIT

LazyData TRUE

Depends R (>= 3.3.0)

Imports Biostrings (>= 2.46.0),
data.table (>= 1.10.4.3),
GenomicAlignments (>= 1.14.1),
GenomicFeatures (>= 1.24.5),
GenomicRanges (>= 1.24.3),
ggplot2 (>= 2.2.1),
ggrepel (>= 0.6.5),
IRanges (>= 2.12.0)

biocViews

RoxygenNote 6.0.1

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

`bamtobed` *From BAM files to BED files.*

Description

This function reads one or multiple BAM files converting them into BED files that contain, for each read: i) the name of the corresponding reference sequence (i.e. of the transcript on which it aligns); ii) its leftmost and rightmost position with respect to the 1st nucleotide of the reference sequence; iii) its length; iv) the strand on which it aligns. Please note: this function relies on the *bamtobed* utility of the BEDTools suite and can be only run on UNIX, LINUX and Apple OS X operating systems. Moreover, to generate R data structures containing reads information, the `bedtolist` must be run on the resulting BED files. For these reasons the authors suggest the use of `bamtolist`.

Usage

```
bamtobed(bamfolder, bedfolder = NULL)
```

Arguments

bamfolder	Character string specifying the path to the folder storing BAM files. Please note: the function looks for BAM files recursively starting from the specified folder.
bedfolder	Character string specifying the path to the directory where BED files shuold be stored. If the specified folder doesn't exist, it is automatically created. If NULL (the default), BED files are stored in a new subfolder of the working directory, called *bed*.

Examples

```
## path_bam <- "path/to/BAM/files"
## path_bed <- "path/to/output/directory"
## bamtobed(bamfolder = path_bam, bedfolder = path_bed)
```

---

bamtolist *From BAM files to lists of data tables or GRangesList objects.*

Description

This function reads one or multiple BAM files converting them into data tables or GRanges objects, arranged in a list or a GRangesList, respectively. In both cases the list elements contain, for each read: i) the name of the corresponding reference sequence (i.e. of the transcript on which it aligns); ii) its leftmost and rightmost position with respect to the 1st nucleotide of the reference sequence; iii) its length; iv) the leftmost and rightmost position of the annotated CDS of the reference sequence (if any) with respect to its 1st nucleotide. Please note: start and stop codon positions for transcripts without annotated CDS are set to 0.

Usage

```
bamtolist(bamfolder, annotation, transcript_align = TRUE,
  name_samples = NULL, rm_version = FALSE, granges = FALSE)
```

Arguments

bamfolder	Character string specifying the path to the folder storing BAM files.
annotation	Data table as generated by create_annotation. Please make sure the name of reference transcripts in the annotation data table match those in the BAM files (see rm_version).
transcript_align	Logical value whether BAM files in bamfolder come from a transcriptome alignment (intended as an alignment against reference transcript sequences, see Details). If TRUE (the default), reads mapping on the negative strand should not be present and, if any, they are automatically removed.

name_samples	Named character string vector specifying the desired name for the output list elements. A character string for each BAM file in bamfolder is required. Plase be careful to name each element of the vector after the correct corresponding BAM file in bamfolder, leaving their path and extension out. No specific order is required. Default is NULL i.e. list elements are named after the name of the BAM files, leaving their path and extension out.
rm_version	Logical value whether to remove the transcript version at the end of their ID, usually dot-separated. It might be required to make the transcripts IDs in the BAM files match those in the annotation table. Default is FALSE.
granges	Logical value whether to return a GRangesList object. Default is FALSE i.e. a list of data tables is returned instead (the required input for length_filter, psite, psite_info, rends_heat and rlength_distr).

Details

riboWaltz only works for read alignments based on transcript coordinates. This choice is due to the main purpose of RiboSeq assays to study translational events through the isolation and sequencing of ribosome protected fragments. Most reads from RiboSeq are supposed to map on mRNAs and not on introns and intergenic regions. Nevertheless, BAM based on transcript coordinates can be generated in two ways: i) aligning directly against transcript sequences; ii) aligning against standard chromosome sequences, requiring the outputs to be translated in transcript coordinates. The first option can be easily handled by many aligners (e.g. Bowtie), given a reference FASTA file where each sequence represents a transcript, from the beginning of the 5' UTR to the end of the 3' UTR. The second procedure is based on reference FASTA files where each sequence represents a chromosome, usually coupled with comprehensive gene annotation files (GTF or GFF). The STAR aligner, with its option –quantMode TranscriptomeSAM (see Chapter 6 of its manual), is an example of tool providing such a feature.

Value

A list of data tables or a GRangesList object.

Examples

```
## path_bam <- "path/to/BAM/files"
## bamtolist(bamfolder = path_bam, annotation = mm81cdna)
```

---

bedtolist *From BED files to lists of data tables or GRangesList objects.*

Description

This function reads one or multiple BED files, as generated by bamtobed, converting them into data tables or GRanges objects, arranged in a list or a GRangesList, respectively. In both cases two columns are attached to the original data containing, for each read, the leftmost and rightmost position of the annotated CDS of the reference sequence (if any) with respect to its 1st nucleotide. Please note: start and stop codon positions for transcripts without annotated CDS are set to 0.

Usage

```
bedtolist(bedfolder, annotation, transcript_align = TRUE,
  name_samples = NULL, rm_version = FALSE, granges = FALSE)
```

Arguments

bedfolder	Character string specifying the path to the folder storing BED files as generated by bamtobed.
annotation	Data table as generated by create_annotation. Please make sure the name of reference transcripts in the annotation data table match those in the BED files (see also rm_version).
transcript_align	Logical value whether BED files in bedfolder come from a transcriptome alignment (intended as an alignment against reference transcript sequences, see Details). If TRUE (the default), reads mapping on the negative strand should not be present and, if any, they are automatically removed.
name_samples	Named character string vector specifying the desired name for the output list elements. A character string for each BED file in bedfolder is required. Plase be careful to name each element of the vector after the correct corresponding BED file in bedfolder, leaving their path and extension out. No specific order is required. Default is NULL i.e. list elements are named after the name of the BED files, leaving their path and extension out.
rm_version	Logical value whether to remove the transcript version at the end of their ID, usually dot-separated. It might be required to make the transcripts IDs in the BED files match those in the annotation table. Default is FALSE.
granges	Logical value whether to return a GRangesList object. Default is FALSE i.e. a list of data tables is returned instead (the required input for length_filter, psite, psite_info, rends_heat and rlength_distr).

Details

riboWaltz only works for read alignments based on transcript coordinates. This choice is due to the main purpose of RiboSeq assays to study translational events through the isolation and sequencing of ribosome protected fragments. Most reads from RiboSeq are supposed to map on mRNAs and not on introns and intergenic regions. Nevertheless, BAM based on transcript coordinates can be generated in two ways: i) aligning directly against transcript sequences; ii) aligning against standard chromosome sequences, requiring the outputs to be translated in transcript coordinates. The first option can be easily handled by many aligners (e.g. Bowtie), given a reference FASTA file where each sequence represents a transcript, from the beginning of the 5' UTR to the end of the 3' UTR. The second procedure is based on reference FASTA files where each sequence represents a chromosome, usually coupled with comprehensive gene annotation files (GTF or GFF). The STAR aligner, with its option –quantMode TranscriptomeSAM (see Chapter 6 of its manual), is an example of tool providing such a feature.

Value

A list of data tables or a GRangesList object.

Examples

```
## path_bed <- "path/to/BED/files"
## bedtolist(bedfolder = path_bed, annotation = mm81cdna)
```

---

cds_coverage *Number of in-frame P-sites per coding sequence.*

Description

This function generates a data table containing, for each transcript: i) its name; ii) its length; iii) the number of in-frame P-sites falling in its annotated coding sequence (if any) for all samples. A chosen number of nucleotides at the beginning and/or at the end of the CDSs can be excluded for restricting the analysis to a subregion of the original sequence. Please note: transcripts without annotated CDS are automatically discarded.

Usage

```
cds_coverage(data, annotation, start_nts = 0, stop_nts = 0)
```

Arguments

data	List of data tables from psite_info.
annotation	Data table as generated by create_annotation.
start_nts	Positive integer specifying the number of nucleotides at the beginning of the coding sequences to be excluded from the analisys. Default is 0.
stop_nts	Positive integer specifying the number of nucleotides at the end of the coding sequences to be excluded from the analisys. Default is 0.

Value

A data table.

Examples

```
data(reads_psite_list)
data(mm81cdna)

## Compute the number of in-frame P-sites per whole coding sequence.
psite_cds <- cds_coverage(reads_psite_list, mm81cdna)

## Compute the number of in-frame P-sites per the coding sequence exluding
## the first 15 nucleotides and the last 10 nucleotides.
psite_cds <- cds_coverage(reads_psite_list, mm81cdna, start_nts = 15, stop_nts = 10)
```

codon_coverage *Number of reads per codon.*

Description

This function computes transcript-specific codon coverages, defined as the number of either read footprints or P-sites mapping on each triplet of coding sequences and UTRs (see *Details*). The resulting data table contains, for each triplet: i) the name of the corresponding reference sequence (i.e. of the transcript to which it belongs); ii) its leftmost and rightmost position with respect to the 1st nucleotide of the reference sequence; iii) its position with respect to the 1st and the last codon of the annotated CDS of the reference sequence; iv) the region of the transcript (5' UTR, CDS, 3' UTR) it is in; v) the number of read footprints or P-sites falling in that region for all samples.

Usage

```
codon_coverage(data, annotation, sample = NULL, psite = FALSE,
  min_overlap = 1, granges = FALSE)
```

Arguments

data	List of data tables from psite_info. Data tables generated by bamtolist and bedtolist can be used if psite is FALSE (the default).
annotation	Data table as generated by create_annotation.
sample	Character string vector specifying the name of the sample(s) of interest. Default is NULL i.e. all samples in data are processed.
psite	Logical value whether to return the number of P-sites per codon. Default is TRUE. If FALSE, the number of read footprints per codon is returned instead.
min_overlap	Positive integer specifying the minimum number of overlapping positions (in nucleotides) between reads and codons to be considered overlapping. If psite is TRUE this parameter must be 1 (the default).
granges	Logical value whether to return a GRangesList object. Default is FALSE i.e. a list of data tables is returned instead.

Details

The sequence of every transcript is divided in triplets starting from the annotated translation initiation site (if any) and proceeding towards the UTRs extremities, possibly discarding the exceeding 1 or 2 nucleotides at the extremities of the transcript. Please note: transcripts not associated to any annotated 5' UTR, CDS and 3'UTR and transcripts whose coding sequence length is not divisible by 3 are automatically discarded.

Value

A data table or a GRanges object.

Examples

```
data(reads_psite_list)
data(mm81cdna)

## Compute the codon coverage based on the number of ribosome footprint per
## codon, setting the minimum overlap between reads and triplets to 3 nts:
## coverage_dt <- codon_coverage(reads_psite_list, mm81cdna, min_overlap = 3)

## Compute the coverage based on the number of P-sites per codon:
## coverage_dt <- codon_coverage(reads_psite_list, mm81cdna, psite = TRUE)
```

---

codon_usage_psite *Empirical codon usage indexes.*

Description

This function computes empirical codon usage indexes based on either ribosome P-sites, A-site or E-site frequency associated to in-frame P-sites along coding sequences. Given one sample, it computes 64 codon usage indexes (one for each triplet, optionally normalized for their frequency in CDSs) and generates a bar plot of the resulting values. If two samples are specified, the function also compares the two sets of codon usage indexes returning a scatter plot. The same output is generated specifying one sample and providing 64 triplet-specific values. Scatter plots report the result of linear regressions between the two sets of values and the corresponding Pearson correlation coefficient.

Usage

```
codon_usage_psite(data, annotation, sample, site = "psite",
  fastapath = NULL, fasta_genome = TRUE, bsgenome = NULL,
  gtfpath = NULL, txdb = NULL, dataSource = NA, organism = NA,
  transcripts = NULL, frequency_normalization = TRUE, codon_values = NULL,
  scatter_label = FALSE, aminoacid = FALSE)
```

Arguments

`data`	List of data tables from `psite_info`. Each data table may or may not include one or more columns among *p_site_codon*, *a_site_codon* and *e_site_codon* reporting the three nucleotides covered by the P-site, A-site and E-site, respectively. These columns can be previously generated by the `psite_info` function. Otherwise, the column of interest can be specified by `site` and it is automatically generated starting from a FASTA file or a BSgenome data package.
`annotation`	Data table as generated by `create_annotation`.
`sample`	Either character string or character string vector specifying one or two sample names, respectively. If one sample name is specified, a bar plot displaying the 64 codon usage indexes is generated. If two sample names are specified, the function also compares the two sets of codon usage indexes returning a scatter plot. In the latter case it also performes a linear regression and computes the Pearson correlation coefficient.

`site` Either "psite, "asite", "esite". It specifies if the empirical codon usage indexes should be based on ribosome P-sites ("psite"), A-sites ("asite") or E-sites ("esite"). Default is "psite".

`fastapath` Character string specifying the FASTA file used in the alignment step, including its path, name and extension. This file can contain reference nucleotide sequences either of a genome assembly or of all the transcripts (see `Details` and `fasta_genome`). Please make sure the sequences derive from the same release of the annotation file used in the `create_annotation` function. Note: either `fastapath` or `bsgenome` is required to compute the frequency in sequences of each codon, used as normalization factors, even if `data` includes one or more columns among *p_site_codon*, *a_site_codon* and *e_site_codon*. Default is NULL.

`fasta_genome` Logical value whether the FASTA file specified by `fastapath` contains nucleotide sequences of a genome assembly. If TRUE (the default), an annotation object is required (see `gtfpath` and `txdb`). FALSE implies nucleotide sequences of all transcripts are provided instead.

`bsgenome` Character string specifying the BSgenome data package with the genome sequences to be loaded. If not already present in the system, it is automatically installed through the biocLite.R script (check the list of available BSgenome data packages by running the `available.genomes` function of the BSgenome package). This parameter must be coupled with an annotation object (see `gtfpath` and `txdb`). Please make sure the sequences included in the specified BSgenome data pakage are in agreement with the sequences used in the alignment step. Note: either `fastapath` or `bsgenome` is required to compute the frequency in sequences of each codon, used as normalization factors, even if `data` includes one or more columns among *p_site_codon*, *a_site_codon* and *e_site_codon*. Default is NULL.

`gtfpath` Character string specifying the location of a GTF file, including its path, name and extension. Please make sure the GTF file and the sequences specified by `fastapath` or `bsgenome` derive from the same release. Note that either `gtfpath` or `txdb` is required if and only if nucleotide sequences of a genome assembly are provided (see `fastapath` or `bsgenome`). Default is NULL.

`txdb` Character string specifying the TxDb annotation package to be loaded. If not already present in the system, it is automatically installed through the biocLite.R script (check here the list of available TxDb annotation packages). Please make sure the TxDb annotation package and the sequences specified by `fastapath` or `bsgenome` derive from the same release. Note that either `gtfpath` or `txdb` is required if and only if nucleotide sequences of a genome assembly are provided (see `fastapath` or `bsgenome`). Default is NULL.

`dataSource` Optional character string describing the origin of the GTF data file. This parameter is considered only if `gtfpath` is specified. For more information about this parameter please refer to the description of *dataSource* of the `makeTxDbFromGFF` function included in the `GenomicFeatures` package.

`organism` Optional character string reporting the genus and species of the organism of the GTF data file. This parameter is considered only if `gtfpath` is specified. For more information about this parameter please refer to the description of

organism of the makeTxDbFromGFF function included in the GenomicFeatures package.

`transcripts` Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used. Please note: transcripts without annotated CDS and transcripts whose coding sequence length is not divisible by 3 are automatically discarded.

`frequency_normalization` Logical value whether to normalize the 64 codon usage indexes for the corresponding codon frequencies in coding sequences. Default is TRUE.

`codon_values` Data table containing 64 triplet-specific values. If specified, the provided values are compared with the empirical codon usage indexes computed for the sample of interest. The data table must contain the DNA or RNA nucleotide sequence of the 64 codons and corresponding values arranged in two columns named *codon* and *value*, respectively. Default is NULL.

`scatter_label` Logical value whether to label the dots in the scatter plot. Each dot is labeled using either the nucleotide sequence of the codon or the corresponding amino acid symbol (see `aminoacid`). This parameter is considered only if two sample names are specified in `sample` or `codon_values` is provided. Default is FALSE.

`aminoacid` Logical value whether to use amino acid symbols to label the dots of the scatter. Default is FALSE i.e. codon nucleotide sequences are used instead. This parameter is considered only if two sample names are specified in `sample` or `codon_values` is provided and `scatter_label` is TRUE.

Details

riboWaltz only works for read alignments based on transcript coordinates. This choice is due to the main purpose of RiboSeq assays to study translational events through the isolation and sequencing of ribosome protected fragments. Most reads from RiboSeq are supposed to map on mRNAs and not on introns and intergenic regions. Nevertheless, BAM based on transcript coordinates can be generated in two ways: i) aligning directly against transcript sequences; ii) aligning against standard chromosome sequences, requiring the outputs to be translated in transcript coordinates. The first option can be easily handled by many aligners (e.g. Bowtie), given a reference FASTA file where each sequence represents a transcript, from the beginning of the 5' UTR to the end of the 3' UTR. The second procedure is based on reference FASTA files where each sequence represents a chromosome, usually coupled with comprehensive gene annotation files (GTF or GFF). The STAR aligner, with its option –quantMode TranscriptomeSAM (see Chapter 6 of its manual), is an example of tool providing such a feature.

Value

A list containing ggplot2 objects and a data table with the associated data ("dt"). If only one sample name is specified in `sample`, one ggplot2 object is returned (a bar plot named "plot"). If two sample names are specified in `sample`, three ggplot2 objects are returned (two bar plots named "plot_NameSample1" and "plot_NameSample2" and a scatter plot named "plot_comparison"). If `codon_values` is specified, two ggplot2 objects are returned (one bar plots named "plot" and a scatter plot named "plot_comparison"). Please note: before plotting, the 64 values are scaled to make them ranging between 0 and 1. If `frequency_normalization` is TRUE, the data table contains raw, normalized and scaled codon usage indexes, only raw and scaled data otherwise.

Examples

```
## ## codon usage from transcriptome alignment
## path_fasta <- "path/to/transcriptome/FASTA/file"
## codon_usage_psite(data = reads_psite_list, annotation = mm81cdna,
##                     sample = "Samp1",
##                     fastapath = path_fasta, fasta_genome = FALSE)
##
## ## codon usage from genome alignment
## path_fasta <- "path/to/genome/FASTA/file"
## codon_usage_psite(data = reads_psite_list, annotation = mm81cdna,
##                     sample = "Samp1",
##                     fastapath = path_fasta, fasta_genome = TRUE)
```

---

create_annotation *Annotation data table.*

Description

This function generates transcript basic annotation data tables starting from GTF files or TxDb objects. Annotation data tables include a column named *transcript* reporting the name of the reference transcripts and four columns named *l_tr*, *l_utr5*, *l_cds* and *l_utr3* reporting the length of the transcripts and of their annotated 5' UTRs, CDSs and 3' UTRs, respectively. Please note: if a transcript region is not annotated its length is set to 0.

Usage

```
create_annotation(gtfpath = NULL, txdb = NULL, dataSource = NA,
  organism = NA)
```

Arguments

gtfpath	A character string specifying the path to a GTF file, including its name and extension. Please make sure the GTF file derives from the same release of the sequences used in the alignment step. Note that either gtfpath or txdb must be specified. Default is NULL.
txdb	Character string specifying the TxDb annotation package to be loaded. If not already present in the system, it is automatically installed through the biocLite.R script (check here the list of available TxDb annotation packages). Please make sure the TxDb annotation package derives from the same release of the sequences used in the alignment step. Note that either gtfpath or txdb must be specified. Default is NULL.
dataSource	Optional character string describing the origin of the GTF data file. This parameter is considered only if gtfpath is specified. For more information about this parameter please refer to the description of *dataSource* of the makeTxDbFromGFF function included in the GenomicFeatures package.

organism — Optional character string reporting the genus and species of the organism of the GTF data file. This parameter is considered only if gtfpath is specified. For more information about this parameter please refer to the description of *organism* of the makeTxDbFromGFF function included in the GenomicFeatures package.

Value

A data table.

Examples

```
## gtf_file <- "path/to/GTF/file.GTF"
## create_annotation(gtfpath = gtf_file, dataSource = "gencode6", organism = "Mus musculus")
```

---

frame_psite *Percentage of P-sites per reading frame.*

Description

This function computes the percentage of P-sites falling in the three possible translation reading frames and generates a bar plot of the resulting values. It only handles annotated 5' UTRs, coding sequences and 3' UTRs, separately.

Usage

```
frame_psite(data, sample = NULL, transcripts = NULL, region = "all",
  length_range = "all", plot_title = NULL)
```

Arguments

data — List of data tables from psite_info.

sample — Character string vector specifying the name of the sample(s) of interest. Default is NULL i.e. all samples in data are processed.

transcripts — Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used.

region — Character string specifying the region(s) of the transcripts to be analysed. It can be either "5utr", "cds", "3utr" for 5' UTRs, CDSs and 3' UTRs, respectively. Default is "all" i.e. all regions are considered. According to this parameter the bar plots are differently arranged to optimise the organization and the visualization of the data.

length_range — Integer or an integer vector specyfying the read length(s) to be included in the analysis. Default is "all" i.e. all read lengths are used.

plot_title — Character string specifying the title of the plot. If "auto", the title of the plot reports the region specified by region (if any) and the considered read length(s). Default is NULL i.e. no title is plotted.

Value

A list containing a ggplot2 object ("plot") and the data table with the associated data ("dt").

Examples

```
data(reads_psite_list)

## Generate the bar plot for all read lengths:
frame_whole <- frame_psite(reads_psite_list, sample = "Samp1")

## Generate the bar plot restricting the analysis to coding sequences and
## reads of 28 nucleotides:
frame_sub <- frame_psite(reads_psite_list, sample = "Samp1", region = "cds",
length_range = 28)
```

---

frame_psite_length *Percentage of P-sites per reading frame stratified by read length.*

Description

Similar to frame_psite, but the results are stratified by read lengths and plotted as heatmaps.

Usage

```
frame_psite_length(data, sample = NULL, transcripts = NULL,
  region = "all", cl = 100, length_range = "all", plot_title = NULL)
```

Arguments

data	List of data tables from psite_info.
sample	Character string vector specifying the name of the sample(s) of interest. Default is NULL i.e. all samples in data are processed.
transcripts	Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used.
region	Character string specifying the region(s) of the transcripts to be analysed. It can be either "5utr", "cds", "3utr" for 5' UTRs, CDSs and 3' UTRs, respectively. Default is "all" i.e. all regions are considered. According to this parameter the heatmaps are differently arranged to optimise the organization and the visualization of the data.
cl	Integer value in [1,100] specifying a confidence level for restricting the analysis to a sub-range of read lengths i.e. to the cl read lengths associated to the highest signals. Default is 100. This parameter has no effect if length_range is specified.
length_range	Integer or an integer vector specyfying the read length(s) to be included in the analysis. Default is "all" i.e. all read lengths are used. If specified, this parameter prevails over cl.

plot_title — Character string specifying the title of the plot. If "auto", the title of the plot reports the region specified by region (if any) and the considered read length(s). Default is NULL i.e. no title is displayed.

Value

A list containing a ggplot2 object ("plot") and the data table with the associated data ("dt").

Examples

```
data(reads_psite_list)

## Generate the heatmap for all read lengths:
frame_len_whole <- frame_psite_length(reads_psite_list, sample = "Samp1")

## Generate the heatmap restricting the analysis to coding sequences and a
## sub-range of read lengths:
frame_len_sub <- frame_psite_length(reads_psite_list, sample = "Samp1",
region = "cds", cl = 90)
```

---

length_filter *Read length filtering.*

Description

Read length filtering.

Usage

```
length_filter(data, length_filter_mode, length_filter_vector = NULL,
  periodicity_threshold = 50, granges = FALSE)
```

Arguments

data — List of data tables from bamtolist, bedtolist or psite_info.

length_filter_mode — Either "custom" or "periodicity". It specifies how read length selection should be performed. "custom": only read lengths specified by the user are kept (see length_filter_vector); "periodicity": only read lengths satisfying a periodicity threshold (see periodicity_threshold) are kept. The latter mode enables the removal of all reads with low or no periodicity.

length_filter_vector — Integer or an integer vector specifying a read length or a range of read lengths to keep, respectively. This parameter is considered only if length_filter_mode is set to "custom".

periodicity_threshold
: Integer in [10, 100]. Only read lengths satisfying this threshold (i.e. a higher percentage of read extremities falls in one of the three reading frames along the CDS) are kept. This parameter is considered only if length_filter_mode is to "periodicity". Default is 50.

granges
: Logical value whether to return a GRangesList object. Default is FALSE i.e. a list of data tables is returned instead (the required input for psite, psite_info, rends_heat and rlength_distr).

Value

A list of data tables or a GRangesList object.

Examples

```
data(reads_list)

## Keep reads of length between 27 and 30 nucleotides (included):
filtered_list <- length_filter(reads_list, length_filter_mode = "custom",
length_filter_vector = 27:30)

## Keep reads of lengths satisfying a periodicity threshold (70%):
filtered_list <- length_filter(reads_list, length_filter_mode = "periodicity",
periodicity_threshold = 70)
```

---

metaheatmap_psite *Ribosome occupancy metaheatmaps at single-nucleotide resolution.*

Description

This function generates two heatmap-like metaprofiles (metaheatmaps) displaying the abundance of P-sites around the start and the stop codon of annotated CDSs. It works similarly to metaprofile_psite but the intensity of signal is represented by a continuous color scale rather than by the height of a line chart. This graphical output is a good option to visualize several profiles at once and compare results obtained with different read lengths or in multiple conditions.

Usage

```
metaheatmap_psite(data, annotation, sample, scale_factors = NULL,
  length_range = "all", transcripts = NULL, utr5l = 25, cdsl = 50,
  utr3l = 25, log_colour = F, colour = "black", plot_title = NULL)
```

Arguments

data
: List of data tables from psite_info.

annotation
: Data table as generated by create_annotation.

`sample` List of either character strings specifying the name of the sample(s) of interest or character string vectors specifying the name of their replicates. In the latter case the final metaheatmaps for each element of the list are generated by merging the results for the corresponding replicates exploiting the scale factors specified by `scale_factors`. The row(s) of the final plot are labelled according to the name of the elements of the list.

`scale_factors` Named numeric vector specifying the scale factors for generating metaprofiles from multiple replicates (see `sample`). Scale factors can be defined for a subset of list elements of `sample` i.e. for all replicates of selected samples. If so, the remaining scale factors are set automatically to 1. Please be careful to name each element of the vector after the correct corresponding string in `sample`. No specific order is required. Default is NULL i.e. all scale factors are automatically set to 1.

`length_range` Integer or an integer vector specyfying the read length(s) to be included in the analysis. Default is "all" i.e. all read lengths are used.

`transcripts` Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used. Please note: transcripts with either 5' UTR, coding sequence or 3' UTR shorter than `utr5l`, 2∗`cdsl` and `utr3l`, respectively, are automatically discarded.

`utr5l` Positive integer specifying the length (in nucleotides) of the 5' UTR region flanking the start codon to be considered in the analysis. Default is 25.

`cdsl` Positive integer specifying the length (in nucleotides) of the CDS regions flanking both the start and stop codon to be considered in the analysis. Default is 50.

`utr3l` Positive integer specifying the length (in nucleotides) of the 3' UTR region flanking the stop codon to be considered in the analysis. Default is 25.

`log_colour` Logical value whether to use a logarithmic colour scale (strongly suggested in case of large signal variations). Default is FALSE.

`colour` Character string specifying the colour of the plot. The colour scheme is as follow: tiles corresponding to the lowest signal are always white, tiles corresponding to the highest signal are of the specified colour and the progression between these two colours follows either linear or logarithmic gradients (see `log_colour`). Default is "black".

`plot_title` Character string specifying the title of the plot. If "auto", the title of the plot reports the number of transcripts and the read length(s) employed for generating the metaprofiles. Default is NULL i.e. no title is displayed.

Details

The intensity of signal in the metaprofiles corresponds, for each nucleotide, to the sum of the number of P-sites (defined by their leftmost position) mapping on that position for all transcripts in one or multiple replicates.

Value

A list containing a ggplot2 object ("plot") and the data table with the associated data ("dt").

Examples

```
data(reads_psite_list)

## Generate metaheatmaps employing all read lengths:
metaheat_whole <- metaheatmap_psite(reads_psite_list, mm81cdna, sample = list("Whole"=c("Samp1")))

## Generate metaprofiles employing reads of 27, 28 and 29 nucleotides and a
## subset of transcripts (in this example only transcripts with at least one
## P-site mapping on the translation initiation site are kept):
sample_name <- "Samp1"
sub_reads_psite_list <- subset(reads_psite_list[[sample_name]], psite_from_start == 0)
transcript_names <- as.character(sub_reads_psite_list$transcript)
metaheat_sub <- metaheatmap_psite(reads_psite_list, mm81cdna, sample = list("sub"=sample_name),
length_range = 27:29, transcripts = transcript_names, plot_title = "auto")

## Generate two sets of metaheatmaps, displayed in the same plot. In this
## example one set of metaheatmaps is based on all read lengths while the
## other one is generated employing only reads of 28 nucleotides:
sample_name <- "Samp1"
metaheat_df <- list()
metaheat_df[["subsample_28nt"]] <- subset(reads_psite_list[[sample_name]], length == 28)
metaheat_df[["whole_sample"]] <- reads_psite_list[[sample_name]]
names_list <- list("Only_28" = c("subsample_28nt"), "All" = c("whole_sample"))
metaheat_comparison <- metaheatmap_psite(metaheat_df, mm81cdna, sample = names_list)
```

---

metaprofile_psite *Ribosome occupancy metaprofiles at single-nucleotide resolution.*

Description

This function generates two metaprofiles displaying the abundance of P-sites around the start and the stop codon of annotated CDSs.

Usage

```
metaprofile_psite(data, annotation, sample, scale_factors = NULL,
  length_range = "all", transcripts = NULL, utr5l = 25, cdsl = 50,
  utr3l = 25, plot_title = NULL)
```

Arguments

data	List of data tables from psite_info.
annotation	Data table as generated by create_annotation.
sample	Either a character string specifying the name of the sample of interest or a character string vector specifying the name of its replicates. In the latter case the final metaprofiles are generated by merging the results for each replicate exploiting the scale factors specified by scale_factors.

`scale_factors` Named numeric vector the same length as `sample` specifying the scale factors for generating metaprofiles from multiple replicates (see `sample`). Please be careful to name each element of the vector after the correct corresponding string in `sample`. No specific order is required. Default is NULL i.e. all scale factors are automatically set to 1.

`length_range` Integer or an integer vector specyfying the read length(s) to be included in the analysis. Default is "all" i.e. all read lengths are used.

`transcripts` Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used. Please note: transcripts with either 5' UTR, coding sequence or 3' UTR shorter than `utr5l`, 2∗`cdsl` and `utr3l`, respectively, are automatically discarded.

`utr5l` Positive integer specifying the length (in nucleotides) of the 5' UTR region flanking the start codon to be considered in the analysis. Default is 25.

`cdsl` Positive integer specifying the length (in nucleotides) of the CDS regions flanking both the start and stop codon to be considered in the analysis. Default is 50.

`utr3l` Positive integer specifying the length (in nucleotides) of the 3' UTR region flanking the stop codon to be considered in the analysis. Default is 25.

`plot_title` Character string specifying the title of the plot. If "auto", the title of the plot reports the sample(s) specified by `sample` as well as the number of transcripts and the read length(s) employed for generating the metaprofiles. Default is NULL i.e. no title is displayed.

Details

The intensity of signal in the metaprofiles corresponds, for each nucleotide, to the sum of the number of P-sites (defined by their leftmost position) mapping on that position for all transcripts in one or multiple replicates.

Value

A list containing a ggplot2 object ("plot") and the data table with the associated data ("dt").

Examples

```
data(reads_psite_list)
data(mm81cdna)

## Generate metaprofiles employing all read lengths:
metaprof_whole <- metaprofile_psite(reads_psite_list, mm81cdna, sample = "Samp1")
metaprof_whole[["plot"]]

## Generate metaprofiles employing reads of 27, 28 and 29 nucleotides and a
## subset of transcripts (in this example only transcripts with at least one
## P-site mapping on the translation initiation site are kept):
sample_name <- "Samp1"
sub_reads_psite_list <- reads_psite_list[[sample_name]][psite_from_start == 0]
transcript_names <- as.character(sub_reads_psite_list$transcript)
```

```
metaprof_sub <- metaprofile_psite(reads_psite_list, mm81cdna, sample = sample_name,
length_range = 27:29, transcripts = transcript_names)
```

`mm81cdna` *Annotation*

Description

A dataset containing basic information about 109,712 mouse mRNA (Ensembl v81 transcript annotation).

Usage

```
mm81cdna
```

Format

A data table with 109,712 rows and 5 variables:

transcript Name of the transcript (ENST ID and version, dot separated)

l_tr Length of the transcript, in nucleotides

l_utr5 Length of the annotated 5' UTR (if any), in nucleotides

l_cds Length of the annotated CDS (if any), in nucleotides

l_utr3 Length of the annotated 3' UTR (if any), in nucleotides

---

`psite` *Ribosome P-sites position within reads.*

Description

This function identifies the exact position of the ribosome P-site within each read, determined by the localisation of its first nucleotide (see `Details`). It returns a data table containing, for all samples and read lengths: i) the percentage of reads in the whole dataset, ii) the percentage of reads aligning on the start codon (if any); iii) the distance of the P-site from the two extremities of the reads before and after the correction step; iv) the name of the sample. Optionally, this function plots a collection of read length-specific occupancy metaprofiles displaying the P-site offsets computed through the process.

Usage

```
psite(data, flanking = 6, start = TRUE, extremity = "auto",
  plot = FALSE, plot_dir = NULL, plot_format = "png", cl = 99)
```

Arguments

`data`	List of data tables from `bamtolist`, `bedtolist` or `length_filter`.
`flanking`	Integer value specifying for the selected reads the minimum number of nucleotides that must flank the reference codon in both directions. Default is 6.
`start`	Logical value whether to use the translation initiation site as reference codon. Default is TRUE. If FALSE, the second to last codon is used instead.
`extremity`	Either "5end", "3end" or "auto". It specifies if the correction step should be based on 5' extremities ("5end") or 3' extremities ("3end"). Default is "auto" i.e. the optimal extremity is automatically selected.
`plot`	Logical value whether to plot the occupancy metaprofiles displaying the P-site offsets computed in both steps of the algorithm. Default is FALSE.
`plot_dir`	Character string specifying the directory where read length-specific occupancy metaprofiles shuold be stored. If the specified folder doesn't exist, it is automatically created. If NULL (the default), the metaprofiles are stored in a new subfolder of the working directory, called *offset_plot*. This parameter is considered only if `plot` is TRUE.
`plot_format`	Either "png" (the default) or "pdf". This parameter specifies the file format storing the length-specific occupancy metaprofiles. It is considered only if `plot` is TRUE.
`cl`	Integer value in [1,100] specifying a confidence level for generating occupancy metaprofiles for to a sub-range of read lengths i.e. for the cl 99. This parameter is considered only if `plot` is TRUE.

Details

The P-site offset (PO) is defined as the distance between the extremities of a read and the first nucleotide of the P-site itself. The function processes all samples separately starting from reads mapping on the reference codon (either the start codon or the second to last codon, see `start`) of any annotated coding sequences. Read lengths-specific POs are inferred in two steps. First, reads mapping on the reference codon are grouped according to their length, each group corresponding to a bin. Reads whose extremities are too close to the reference codon are discarded (see `flanking`). For each bin temporary 5' and 3' POs are defined as the distances between the first nucleotide of the reference codon and the nucleotide corresponding to the global maximum found in the profiles of the 5' and the 3' end at the left and at the right of the reference codon, respectively. After the identification of the P-site for all reads aligning on the reference codon, the POs corresponding to each length are assigned to each read of the dataset. Second, the most frequent temporary POs associated to the optimal extremity (see `extremity`) and the predominant bins are exploited as reference values for correcting the temporary POs of smaller bins. Briefly, the correction step defines for each length bin a new PO based on the local maximum, whose distance from the reference codon is the closest to the most frequent temporary POs. For further details please refer to the **riboWaltz** article (available here).

Value

A data table.

Examples

```
data(reads_list)

## Compute the P-site offset automatically selecting the optimal read
## extremity for the correction step and not plotting any metaprofile:
psite(reads_list, flanking = 6, extremity="auto")

## Compute the P-site offset specifying the extremity used in the correction
## step and plotting the length-specific occupancy metaprofiles for a
## sub-range of read lengths (the middle 95%). The plots will be placed in
## the current working directory:
psite_offset <- psite(reads_list, flanking = 6, extremity = "3end", plot = TRUE, cl = 95)
```

---

psite_info *Update reads information according to the inferred P-sites.*

Description

This function provides additional reads information according to the position of the P-site identfied by psite. It attaches to each data table in a list four columns reporting i) the P-site position with respect to the 1st nucleotide of the transcript, ii) the P-site position with respect to the start and the stop codon of the annotated coding sequence (if any) and iii) the region of the transcript (5' UTR, CDS, 3' UTR) that includes the P-site. Please note: for transcripts not associated to any annotated CDS the position of the P-site with respect to the start and the stop codon is set to NA. Optionally, additional columns reporting the three nucleotides covered by the P-site, the A-site and the E-site are attached, based on FASTA files or BSgenome data packages containing the transcript nucleotide sequences.

Usage

```
psite_info(data, offset, site = NULL, fastapath = NULL,
  fasta_genome = TRUE, bsgenome = NULL, gtfpath = NULL, txdb = NULL,
  dataSource = NA, organism = NA, granges = FALSE)
```

Arguments

data	List of data tables from bamtolist, bedtolist or length_filter.
offset	Data table from psite.
site	Either "psite, "asite", "esite" or a combination of these strings. It specifies if additional column(s) reporting the three nucleotides covered by the ribosome P-site ("psite"), A-site ("asite") and E-site ("esite") should be added. Note: either fastapath or bsgenome is required for this purpose. Default is NULL.
fastapath	Character string specifying the FASTA file used in the alignment step, including its path, name and extension. This file can contain reference nucleotide sequences either of a genome assembly or of all the transcripts (see Details and fasta_genome). Please make sure the sequences derive from the same release of the annotation file used in the create_annotation function. Note: either

fastapath or bsgenome is required to generate additional column(s) specified by site. Default is NULL.

fasta_genome — Logical value whether the FASTA file specified by fastapath contains nucleotide sequences of a genome assembly. If TRUE (the default), an annotation object is required (see gtfpath and txdb). FALSE implies the nucleotide sequences of all the transcripts is provided instead.

bsgenome — Character string specifying the BSgenome data package with the genome sequences to be loaded. If not already present in the system, it is automatically installed through the biocLite.R script (check the list of available BSgenome data packages by running the available.genomes function of the BSgenome package). This parameter must be coupled with an annotation object (see gtfpath and txdb). Please make sure the sequences included in the specified BSgenome data pakage are in agreement with the sequences used in the alignment step. Note: either fastapath or bsgenome is required to generate additional column(s) specified by site. Default is NULL.

gtfpath — Character string specifying the location of a GTF file, including its path, name and extension. Please make sure the GTF file and the sequences specified by fastapath or bsgenome derive from the same release. Note that either gtfpath or txdb is required if and only if nucleotide sequences of a genome assembly are provided (see fastapath or bsgenome). Default is NULL.

txdb — Character string specifying the TxDb annotation package to be loaded. If not already present in the system, it is automatically installed through the biocLite.R script (check here the list of available TxDb annotation packages). Please make sure the TxDb annotation package and the sequences specified by fastapath or bsgenome derive from the same release. Note that either gtfpath or txdb is required if and only if nucleotide sequences of a genome assembly are provided (see fastapath or bsgenome). Default is NULL.

dataSource — Optional character string describing the origin of the GTF data file. This parameter is considered only if gtfpath is specified. For more information about this parameter please refer to the description of *dataSource* of the makeTxDbFromGFF function included in the GenomicFeatures package.

organism — Optional character string reporting the genus and species of the organism of the GTF data file. This parameter is considered only if gtfpath is specified. For more information about this parameter please refer to the description of *organism* of the makeTxDbFromGFF function included in the GenomicFeatures package.

granges — Logical value whether to return a GRangesList object. Default is FALSE i.e. a list of data tables (the required input for downstream analyses and graphical outputs provided by riboWaltz) is returned instead.

Details

riboWaltz only works for read alignments based on transcript coordinates. This choice is due to the main purpose of RiboSeq assays to study translational events through the isolation and sequencing of ribosome protected fragments. Most reads from RiboSeq are supposed to map on mRNAs and not on introns and intergenic regions. Nevertheless, BAM based on transcript coordinates can be generated in two ways: i) aligning directly against transcript sequences; ii) aligning against

standard chromosome sequences, requiring the outputs to be translated in transcript coordinates. The first option can be easily handled by many aligners (e.g. Bowtie), given a reference FASTA file where each sequence represents a transcript, from the beginning of the 5' UTR to the end of the 3' UTR. The second procedure is based on reference FASTA files where each sequence represents a chromosome, usually coupled with comprehensive gene annotation files (GTF or GFF). The STAR aligner, with its option –quantMode TranscriptomeSAM (see Chapter 6 of its manual), is an example of tool providing such a feature.

Value

A list of data tables or a GRangesList object.

Examples

```
data(reads_list)
data(psite_offset)
data(mm81cdna)

reads_psite_list <- psite_info(reads_list, psite_offset)
```

---

psite_offset *P-site offsets*

Description

An example dataset containing length-specific ribosome P-site offsets as returned by `psite` applied to `reads_list`.

Usage

```
psite_offset
```

Format

A data table with 31 rows and 9 variables:

length Length of the read, in nucleotides

total_percentage Percentage of reads of the considered length in the whole dataset

start_percentage Percentage of reads of the considered length aligning on the start codon (if any)

around_start A logical value whether at least one read of the considered length aligns on the start codon (T = yes, F = no)

offset_from_5 Temporary P-site offset from the 5' end of the read, in nucleotides (before the correction step)

offset_from_3 Temporary P-site offset from the 3' end of the read, in nucleotides (before the correction step)

corrected_offset_from_5 P-site offset from the 5' end of the read, in nucleotides (after the correction step)

corrected_offset_from_3 P-site offset from the 3' end of the read, in nucleotides (after the correction step)

sample Name of the sample

`reads_list` *Reads information*

Description

An example dataset containing details on reads mapping on the mouse transcriptome, generated from BAM or BED files. A subset of the original dataset is provided, including only reads aligning on the translation initiation site. Please contact the authors for more information.

Usage

```
reads_list
```

Format

A list of data tables with 1 object (named *Samp1*) of 393,338 rows and 6 variables:

transcript Name of the transcript (ENST ID and version, dot separated)

end5 Position of the 5' end of the read with respect to the first nucleotide of the transcript, in nucleotides

end3 Position of the 3' end of the read with respect to the first nucleotide of the transcript, in nucleotides

length Length of the read, in nucleotides

cds_start Leftmost position of the annotated CDS with respect to the first nucleotide of the transcript, in nucleotides

cds_stop Rightmost position of the annotated CDS with respect to the first nucleotide of the transcript, in nucleotides

---

`reads_psite_list` *Reads details updated with P-site information*

Description

An example dataset that combines details on reads mapping on the mouse transcriptome (see `reads_list`) and length-specific ribosome P-site offsets (see `psite_offset`), as returned by `psite_info`.

Usage

```
reads_psite_list
```

Format

A list of data tables with 1 object (named *Samp1*) of 393,338 rows and 10 variables:

transcript Name of the transcript (ENST ID and version, dot separated)

end5 Position of the 5' end of the read with respect to the first nucleotide of the transcript, in nucleotides

psite Position of the P-site with respect to the first nucleotide of the transcript, in nucleotides

end3 Position of the 3' end of the read with respect to the first nucleotide of the transcript, in nucleotides

length Length of the read, in nucleotides

cds_start Leftmost position of the CDS with respect to the first nucleotide of the transcript, in nucleotides

cds_stop Rightmost position of the CDS with respect to the first nucleotide of the transcript, in nucleotides

psite_from_start Position of the P-site with respect to the first nucleotide of the annotated CDS (if any), in nucleotides

psite_from_stop Position of the P-site with respect to the last nucleotide of the annotated CDS (if any), in nucleotides

psite_region Region of the transcript that includes the P-site (5utr, cds, 3utr)

---

region_psite *Percentage of P-sites per transcript region.*

Description

This function computes the percentage of P-sites falling in the three annotated regions of the transcripts (5' UTR, CDS and 3' UTR) and generates a bar plot of the resulting values.

Usage

```
region_psite(data, annotation, sample = NULL, transcripts = NULL,
  label_sample = NULL, colour = c("gray70", "gray40", "gray10"))
```

Arguments

`data`	List of data tables from `psite_info`.
`annotation`	Data table as generated by `create_annotation`.
`sample`	Character string vector specifying the name of the sample(s) of interest. Default is NULL i.e. all samples in `data` are processed.
`transcripts`	Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used. Please note: transcripts without annotated 5' UTR, CDS and 3' UTR are automatically discarded.

label_sample	Named character string vector the same length as sample specifying the sample names to be displayed in the plot. Plase be careful to name each element of the vector after the correct corresponding string in sample. No specific order is required. Default is NULL i.e. sample names in sample are used.
colour	Character string vector of three elements specifying the colour for the 5' UTR, CDS and 3' UTR bars, respectively. Default is a grayscale.

Details

Column "RNAs" reports the percentage of region length for the transcripts included in the analysis, based on the cumulative nucleotide length of 5' UTRs, CDSs and 3' UTRs. These values reflect the expected read distribution from a random fragmentation of RNA and can be used as a baseline to verify the expected enrichment of ribosome (P-site) signal in CDSs.

Value

A list containing a ggplot2 object ("plot") and the data table with the associated data ("dt").

Examples

```
data(reads_psite_list)
data(mm81cdna)

reg_psite <- region_psite(reads_psite_list, mm81cdna, sample = "Samp1")
reg_psite[["plot"]]
```

---

rends_heat *Metaheatmaps of the two extremities of the reads.*

Description

This function generates four metaheatmaps displaying the abundance of the 5' and 3' extremity of reads mapping around the start and the stop codon of annotated CDSs, stratified by their length.

Usage

```
rends_heat(data, annotation, sample, transcripts = NULL, cl = 95,
  utr5l = 50, cdsl = 50, utr3l = 50, log_colour = F, colour = "black")
```

Arguments

data	List of data tables from bamtolist, bedtolist, length_filter or psite_info.
annotation	Data table as generated by create_annotation.
sample	Character string specifying the name of the sample of interest.
transcripts	Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used. Please note: transcripts with either 5' UTR, coding sequence or 3' UTR shorter than utr5l, 2∗cdsl and utr3l, respectively, are automatically discarded.

cl	Integer value in [1,100] specifying a confidence level for restricting the plot to a sub-range of read lengths i.e. to the cl read lengths associated to the highest signals. Default is 95.
utr5l	Positive integer specifying the length (in nucleotides) of the 5' UTR region flanking the start codon to be considered in the analysis. Default is 50.
cdsl	Positive integer specifying the length (in nucleotides) of the CDS regions flanking both the start and stop codon to be considered in the analysis. Default is 50.
utr3l	Positive integer specifying the length (in nucleotides) of the 3' UTR region flanking the stop codon to be considered in the analysis. Default is 50.
log_colour	Logical value whether to use a logarithmic colour scale (strongly suggested in case of large signal variations). Default is FALSE.
colour	Character string specifying the colour of the plot. The colour scheme is as follow: tiles corresponding to the lowest signal are always white, tiles corresponding to the highest signal are of the specified colour and the progression between these two colours follows either linear or logarithmic gradients (see log_colour). Default is "black".

Value

A list containing a ggplot2 object ("plot") and the data table with the associated data ("dt").

Examples

```
data(reads_list)
data(mm81cdna)

## Generate metaheatmaps for all read lengths:
heatend_whole <- rends_heat(reads_list, mm81cdna, sample = "Samp1", cl = 100)

## Generate metaheatmaps for a sub-range of read lengths shortening the
## flanking regions around the start and stop codon:
heatend_sub95 <- rends_heat(reads_list, mm81cdna, sample = "Samp1", cl = 95,
utr5l = 30, cdsl = 40, utr3l = 30)
```

---

rlength_distr *Read length distributions.*

Description

This function generates read length distributions.

Usage

```
rlength_distr(data, sample, transcripts = NULL, cl = 100)
```

Arguments

data	List of data tables from bamtolist, bedtolist, length_filter or psite_info.
sample	Character string specifying the name of the sample of interest.
transcripts	Character string vector listing the name of transcripts to be included in the analysis. Default is NULL i.e. all transcripts are used.
cl	Integer value in [1,100] specifying a confidence level for restricting the plot to a sub-range of read lengths i.e. to the cl read lengths associated to the highest signals. Default is 100.

Value

List containing a ggplot2 object ("plot") and the data table with the associated data ("dt").

Examples

```
data(reads_list)

## Generate the length distribution for all read lengths:
lendist_whole <- rlength_distr(reads_list, sample = "Samp1", cl = 100)
lendist_whole[["plot"]]

## Generate the length distribution for a sub-range of read lengths:
lendist_sub95 <- rlength_distr(reads_list, sample = "Samp1", cl = 95)
lendist_sub95[["plot"]]
```

Index