Package 'riboWaltz'

January 4, 2024

Type Package

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

Version 2.0

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 + file LICENSE

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 7.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

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 (intended as the width of the reference sequence region covered by the RNA fragment. For
further information about this choice please refer to section Details of function bamtolist); 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 function 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 (intended as the width of the reference sequence region covered by the RNA fragment, see parameter `indel_threshold` and *Details*); 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,
  indel_threshold = 5,
  refseq_sep = NULL,
  output_class = "datatable"
)
```

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 `refseq_sep`).

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 elements of the output list of data tables. 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. See the example for additional details.

indel_threshold

        Positive integer value specifying the maximum number of indels (insertions + deletions) allowed for each read. All reads associated to more indels than specified are discarded (see `Details`). Default is 5.

refseq_sep   Character specifying the separator between reference sequences' name and additional information to discard, stored in the same field (see `Details`). All characters before the first occurrence of the specified separator are kept. Default is NULL i.e. no string splitting is performed.

output_class   Either "datatable" or "granges". It specifies the format of the output i.e. a list of data tables or a GRangesList object. Default is "datatable".

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.

`indel_threshold` is aimed at keeping only reads showing small differences between the width of the RNA fragment and the length of the reference sequence region covered after their alignment (for additional details please read [here](#) about *qwidth* and *width*, defined in the GAlignments-class of the GenomicAlignments package). Strong differences between these values are usually due to many indels which in turn are caused by loose thresholds used in the alignment process. High numbers of consecutive deletions (nucleotides present in the reference sequence but not in the read) may be an indication of reads mapping on exon-exon junctions. Despite this eventuality is not in accordance with alignments based on transcript coordinates, stretches of deletions can be included by errors or inaccurate transcript annotation and should be discarded. In fact, **riboWaltz** deliberately identifies the length of the reference sequences covered by the reads with the reads length, referring to it as *length* in all data structures reporting reads or P-site offsets information (i.e. those generated by `bamtolist` itself, [bamtobed](#), [bedtolist](#), [psite](#) and [psite_info](#)). This value is indeed strongly connected to the identification of the P-site offsets since it determines the position of the 5' and 3' read extremities (columns *end5* and *end3* in many data structures generated by the package), used as starting points by **riboWaltz**'s core algorithm.

`refseq_sep` is intended to lighten the identifiers of the reference sequences included in the output list of data table or to modify them to match those in the annotation table. Many details about the reference sequence such as their version (usually dot-separated), their length, name variants, associated gene/transcript/protein names (usually pipe-separated) might indeed be stored in the FASTA file used for the alignment and automatically transferred in the BAM.

Value

A list of data tables or a GRangesList object.

Examples

```
## ## Let's suppose there are two BAM files ("Samp1.bam" and "Samp2.bam") in
```

```
## ## "path/to/BAM/files". We want to acquire them and assign to the
## ## corresponding data tables the names "Control" and "Treated",
## ## respectively.
## ## We first define the "name_samples" character string vector as follow:
## name_of_bams <- c("Control", "Treated")
## names(name_of_bams) <- c("Samp1", "Samp2")
##
## ## Then, we can acquire the two files:
## path_bam <- "path/to/BAM/files"
## reads_list <- bamtolist(bamfolder = path_bam, name_samples = name_of_bams,
## ##                         annotation = annotation_dt)
##
## ## read_list will be a list of two data tables, named "Control" (with
## ## mapping reads from "Samp1.bam") and "Treated" (with mapping reads
## ## from"Samp2.bam")
```

---

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,
  refseq_sep = FALSE,
  output_class = "datatable"
)
```

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 refseq_sep).

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.
refseq_sep	Character specifying the separator between reference sequences' name and additional information to discard, stored in the same field (see Details). All characters before the first occurrence of the specified separator are kept. Default is NULL i.e. no string splitting is performed.
output_class	Either "datatable" or "granges". It specifies the format of the output i.e. a list of data tables or a GRangesList object. Default is "datatable".

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.

refseq_sep is intended to lighten the identifiers of the reference sequences included in the final data table or to modify them to match those in the annotation table. Many details about the reference sequence such as their version (usually dot-separated), their length, name variants, associated gene/transcript/protein names (usually pipe-separated) might indeed be stored in the FASTA file used for the alignment and automatically transferred in the BAM.

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 or transcript.*

Description

This function generates a data table that for each transcript contains at least i) its name; ii) its length or the length of its annotated coding sequence (if any); iii) the number of P-sites falling in its annotated coding sequence or in the whole transcript for all samples. When the coverage is computed for coding sequences and not whole transcripts, 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 portion of the original coding sequence. In this case the output data table includes an additional column reporting the length of the selected region. Moreover, either all P-sites or only in-frame P-sites can be considered. Finally, transcripts without annotated CDS are automatically discarded.

Usage

```
cds_coverage(
  data,
  annotation,
  start_nts = 0,
  stop_nts = 0,
  in_frame = TRUE,
  whole_transcript = FALSE
)
```

Arguments

data	Either list of data tables or GRangesList object 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. This parameter is considered only if `whole_transcript` is FALSE.
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. This parameter is considered only if `whole_transcript` is FALSE.
in_frame	Logical value whether to consider only in-frame P-sites when computing the coverage. Default is TRUE. This parameter is considered only if `whole_transcript` is FALSE.
whole_transcript	
	Logical value whether to compute the coverage based on P-sites falling in any region of the transcript, i.e. CDS and UTRs. Default is FALSE.

Value

A data table.

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## compute and add p-site datails
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## ## Compute the number of in-frame P-sites per whole coding sequences.
## psite_cds <- cds_coverage(reads_psite_list, mm81cdna)
##
## ## Compute the number of in-frame P-sites per coding sequences exluding
## ## the first 15 nucleotides and the last 10 nucleotides.
## psite_cds <- cds_coverage(reads_psite_list, mm81cdna,
##                           start_nts = 15, stop_nts = 10)
```

```
##
## ## Compute the number of P-sites per transcripts.
## psite_cds <- cds_coverage(reads_psite_list, mm81cdna,
##                             whole_transcript = TRUE)
```

---

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,
  output_class = "datatable"
)
```

Arguments

data	Either list of data tables or GRangesList object from `psite_info`. Data tables and GRanges objects 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).
output_class	Either "datatable" or "granges". It specifies the format of the output i.e. a list of data tables or a GRangesList object. Default is "datatable".

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 GRanges object.

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## compute and add p-site datails
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## 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 the ribosome P-sites, A-site or E-site frequency associated with in-frame P-sites falling in the coding sequence. For each sample it computes 64 triplet-specific codon usage indexes ranging from 0 to 1 and optionally normalized for the frequency of the corresponding triplet within the CDSs. This function also allows to compare either two sets of codon usage indexes from different samples or the set of codon usage indexes computed for one sample and 64 triplet-specific values provided by the user. Multiple samples and replicates can be handled.

Usage

```
codon_usage_psite(
  data,
  annotation,
  sample,
  multisamples = "average",
  plot_style = "split",
  site = "psite",
  frequency_normalization = TRUE,
  contrast_sample = NULL,
  codon_values = NULL,
  fastapath = NULL,
  fasta_genome = TRUE,
  refseq_sep = NULL,
  bsgenome = NULL,
  gtfpath = NULL,
  txdb = NULL,
  dataSource = NA,
  organism = NA,
```

```
    transcripts = NULL,
    length_range = NULL,
    cl = 100,
    include_stop_codons = TRUE,
    label_scatter = FALSE,
    label_number = 64,
    label_aminoacid = FALSE
)
```

Arguments

data

Either list of data tables or GRangesList object 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 must 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, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled and organized according to `multisamples` and `plot_style`.

multisamples

Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in `sample`:

- if `sample` is a character string vector and `multisamples` is set to "average", the elements of the vector are considered as replicates of one sample.

- if `sample` is a character string vector and `multisamples` is set to "independent", each element of the vector is analysed independently of the others.

- if `sample` is a list, `multisamples` must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. Note: when this parameter is set to "average" the bar plot associated with each sample displays the triplet-specific mean signal computed across the replicates and the corresponding standard error is also reported. Scatter plots (see `contrast_sample`) display the mean signal but not the standard error to streamline the plot. Default is "average".

plot_style

Either "split" or "facet". It specifies how to organize and display multiple bar plots:

- "split": one bar plot for each sample is returned as an independent ggplot object. In each plot the bars are arranged in increasing order according to the codon usage indexes.

- "facet": the bar plots are placed one below the other, in independent boxes. The bars of all plots are arranged following the alphabetical order of the codons. Note: this parameter is considered only if `contrast_sample` is not specified and thus bar plots are generated. Default is "split".

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".

frequency_normalization

Logical value whether to normalize the 64 codon usage indexes for the corresponding codon frequencies in coding sequences. Default is TRUE. Note: the low frequency of the three stop codons often leads to codon usage indexes higher for these triplets than for the others. To discard the stop codons from the plots and avoid potential biases in the interpretation of the data, see `include_stop_codons`.

contrast_sample

Either character string or character string vector. It specifies the sample(s) (if any) to be considered for the comparison of codon usage indexes between: exactly two elements from those listed either in `sample` (if `sample` is a character string or a character string vector) or in `names(sample)` (if `sample` is a list).

- one sample and 64 triplet-specific values: `contrast_sample` must be one character string from those listed either in `sample` (if `sample` is a character string or a character string vector) or in `names(sample)` (if `sample` is a list). Moreover, `codon_values` must be provided. Note: when this parameter is specified the function generates a scatter plot instead of one or multiple bar plots, it performs a linear regression and computes the Pearson correlation coefficient. See also `label_scatter`, `label_number` and `label_aminoacid`. Default is "NULL".

codon_values

Data table containing 64 triplet-specific values to be compared with the empirical codon usage indexes computed for the sample specified in `contrast_sample`. The data table must contain the DNA or RNA nucleotide sequence of the 64 codons and the corresponding values arranged in two columns named *codon* and *value*, respectively. Additional columns are ignored. This parameter is considered only if `contrast_sample` is specified. 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 genome asseblies (chromosome sequences) or of 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 of each codon along sequences, 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 genome asseblies (chromosome sequences). If TRUE (the default), an annotation object is required (see `gtfpath` and `txdb`). FALSE implies nucleotide sequences of transcripts are provided instead.

refseq_sep

Character specifying the separator between reference sequences' name and additional information to discard, stored in the headers of the FASTA file specified by `fastapath` (if any). It might be required for matching the reference sequences' identifiers reported in the input list of data tables. All characters before the first occurrence of the specified separator are kept. Default is NULL i.e. no string splitting is performed.

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 of each codon along sequences, 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 genome assemblies (chromosome sequences) 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 genome assemblies (chromosome sequences) 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.

length_range   Integer or integer vector for restricting the plot to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used. If specified, this parameter prevails over `cl`.

cl   Integer value in 1,100 specifying a confidence level for restricting the plot to an automatically-defined range of read lengths. The new range is computed according to the most frequent read lengths, which accounts for the cl% of the sample and is defined by discarding the (100-cl)% of read lengths falling in the tails of the read lengths distribution. If multiple samples are analysed, a single range of read lengths is computed such that at least the cl% of all samples is represented. Default is 100.

include_stop_codons
                 Logical value whether to include the three stop codons in the plots. Default is TRUE.

label_scatter   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 `label_aminoacid`). This parameter is considered only if `contrast_sample` is specified. Default is FALSE.

label_number   Integer value in 1,64 specifying how many dots in the scatter plot should be labeled. Dots farthest from the confident interval of the regression line are automatically identified and labeled. Default is 64 i.e. all dots are labeled. This parameter is considered only if `label_scatter` is TRUE.

```
label_aminoacid
```

Logical value whether to use amino acid symbols to label the dots of the scatter plot. Default is FALSE i.e. codon nucleotide sequences are used instead. This parameter is considered only if `label_scatter` 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. BAM based on transcript coordinates can be generated in two ways: i) aligning directly against transcript sequences; ii) aligning against sequences of genome assemblies i.e. standard chromosome sequences, thus 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

List containing: one or more ggplot object(s) and the data table with the corresponding x- and y-axis values ("plot_dt"); an additional data table with raw, normalized and scaled number of P-sites associated with the 64 triplets for each sample contributing to the plot ("count_dt").

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## Generate fake samples and replicates
## for(i in 2:6){
##   samp_name <- paste0("Samp", i)
##   set.seed(i)
##   reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
## }
##
## ## Compute and add p-site details
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## ## Define the list of samples and replicate to use as input
## input_samples <- list("S1" = c("Samp1", "Samp2"),
##                       "S2" = c("Samp3", "Samp4", "Samp5"),
##                       "S3" = c("Samp6"))
##
## Generate bar plots, alignment on transcript sequences
## example_cu_barplot <- codon_usage_psite(reads_psite_list, mm81cdna,
##                                 sample = input_samples,
##                                 multisamples = "average",
##                                 plot_style = "facet",
##                               fastapath = "path/to/transcriptome/FASTA/file",
##                                 fasta_genome = FALSE)
##
## Generate bar plots,  alignment on chromosome sequences
```

```
## example_cu_barplot <- codon_usage_psite(reads_psite_list, mm81cdna,
##                                          sample = input_samples,
##                                          multisamples = "average",
##                                          plot_style = "facet",
##                                          fastapath = "path/to/chromosome/FASTA/file",
##                                          fasta_genome = TRUE)
##
## Generate scatterplot comparing two samples
## example_cu_barplot <- codon_usage_psite(reads_psite_list, mm81cdna,
##                                          sample = input_samples,
##                                          contrast_sample = c("S1", "S2"),
##                                        fastapath = "path/to/transcriptome/FASTA/file",
##                                          fasta_genome = FALSE,
##                                          frequency_normalization = FALSE)
```

---

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")
```

---

duplicates_filter   *Duplicates filtering.*

Description

This function provides multiple options for remove duplicated reads: when two or more reads are marked as duplicates, all of them are discarded but one.

Usage

```
duplicates_filter(
  data,
  sample = NULL,
  extremity = "both",
  keep = "shortest",
  output_class = "datatable",
  txt = FALSE,
  txt_file = NULL
)
```

Arguments

data	Either list of data tables or GRangesList object from bamtolist, bedtolist or length_filter.
sample	Character string or character string vector specifying the name of the sample(s) to process. Default is NULL i.e. all samples are processed.
extremity	Either "both", "5end", "3end". It specifies the criterion to define which reads should be considered duplicates. Reads are marked as duplicates if they map on the same transcript and share: both the 5' estremity and the 3' extremity ("both"), only the 5' extremity ("5end"), only the same 3' extremity ("3end "). For "5end" and "3end", reads of different lengths can be marked as duplicates. See keep to choose which one should be kept.
keep	Either "shortest" or "longest". It specifies wheter to keep the shortest or the longest read when duplicates display different lengths. This parameter is considered only if extremity is set to "5end" or "3end". Default is "shortest".
output_class	Either "datatable" or "granges". It specifies the format of the output i.e. a list of data tables or a GRangesList object. Default is "datatable".
txt	Logical value whether to write in a txt file statistics on the filtering step. Similar information are displayed by default in the console. Default is FALSE.

txt_file   Character string specifying the path, name and extension (e.g. "PATH/NAME.extension")
                  of the plain text file where statistics on the filtering step shuold be written. If the
                  specified folder doesn't exist, it is automatically created. If NULL (the default),
                  the information are written in *"duplicates_filtering.txt"*, saved in the working
                  directory. This parameter is considered only if txt is TRUE.

Value

A list of data tables or a GRangesList object.

Examples

```
#generate an \emph{ad hoc} dataset:
library(data.table)
dt <- data.table(transcript = rep("ENSMUST00000000001.4", 6),
                 end5 = c(92, 92, 92, 94, 94, 95),
                 end3 = c(119, 119, 122, 122, 123, 123)
                 )[, length := end3 - end5 + 1
                   ][, cds_start := 14
                     ][, cds_stop := 1206]
example_reads_list <- list()
example_reads_list[["Samp_example"]] <- dt

## Reads are duplicates if they share both the 5' estremity and the
## 3' extremity:
filtered_list <- duplicates_filter(example_reads_list,
                                   extremity = "both")

## Reads are duplicates if they only share the 5' estremity. Among duplicated
## reads we keep the shortes one:
filtered_list <- duplicates_filter(example_reads_list,
                                   extremity = "5end",
                                   keep = "shortest")
```

---

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

Description

This function computes, for each transcript region (5' UTR, coding sequence and 3' UTR), the
percentage of P-sites falling in the three possible translation reading frames. It takes into account
only transcripts with annotated regions of length > 0 and the resulting values are visualized as bar
plots. Multiple samples and replicates can be handled.

Usage

```
frame_psite(
  data,
  annotation,
  sample,
  multisamples = "average",
  plot_style = "split",
  transcripts = NULL,
```

```
  region = "all",
  length_range = NULL,
  cl = 100,
  colour = NULL
)
```

Arguments

data	Either list of data tables or GRangesList object from [`psite_info`]().
annotation	Data table as generated by [`create_annotation`]().
sample	Either character string, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled and visualised according to multisamples and plot_style.
multisamples	Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in sample:

> - if sample is a character string vector and multisamples is set to "average" the elements of the vector are considered as replicates of one sample and a single bar plot is returned.
> - if sample is a character string vector and multisamples is set to "independent", each element of the vector is analysed independently of the others. The number of plots returned and their organization is specified by plot_style.
> - if sample is a list, multisamples must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. The number of plots returned and their organization is specified by plot_style. Note: when this parameter is set to "average" the bar plot associated with each sample displays the frame-specific mean signal computed across the replicates and the corresponding standard error is also reported. Default is "average".

plot_style	Either "split", "facet", "dodge" or "mirror". It specifies how to organize and display multiple bar plots:

> - "split": one bar plot for each sample is returned as an independent ggplot object.
> - "facet": the bar plots are placed one next to the other, in independent boxes.
> - "dodge": all bar plots are displayed in one box and, for each frame, samples are placed side by side.
> - "mirror": sample must be either a character string vector or a list of exactly two elements and the resulting bar plots are mirrored along the x axis. Default is "split".

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", respectively for 5' UTRs, CDSs and 3' UTRs, or "all", i.e. all regions are considered. Note: according to this parameter the bar plots are differently arranged to optimise the organization and the visualization of the data.
length_range	Integer or integer vector for restricting the analysis to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used.

cl	Integer value in 1,100 specifying a confidence level for restricting the analysis to an automatically-defined range of read lengths. The new range is computed according to the most frequent read lengths, which accounts for the cl% of the sample and is defined by discarding the (100-cl)% of read lengths falling in the tails of the read lengths distribution. If multiple sample names are provided, one range of read lengths is computed, such that at least the cl% of all samples is represented. Default is 100.
colour	Character string or character string vector specifying the colour of the bar plot(s). If `plot_style` is set to either "dodge" or "mirror", a colour for each sample is required. Default is NULL, i.e. the default R colour palette is used.

Value

List containing: one or more ggplot object(s) and the data table with the corresponding x- and y-axis values ("plot_dt"); an additional data table with raw and scaled number of P-sites per frame for each sample ("count_dt").

See Also

frame_psite_length for a similar plot stratified by read length.

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## Generate fake samples and replicates
## for(i in 2:6){
##   samp_name <- paste0("Samp", i)
##   set.seed(i)
##   reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
## }
##
## ## Compute and add p-site details
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## ## Define the list of samples and replicate to use as input
## input_samples <- list("S1" = c("Samp1", "Samp2"),
##                       "S2" = c("Samp3", "Samp4", "Samp5"),
##                       "S3" = c("Samp6"))
##
## Generate bar plots
## example_frames <- frame_psite(reads_psite_list, mm81cdna,
##                                sample = input_samples,
##                                multisamples = "average",
##                                plot_style = "facet",
##                                region = "all",
##                                colour = c("#333f50", "#39827c", "gray70"))
```

---

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

Description

This function computes, for each transcript region (5' UTR, coding sequence and 3' UTR), the percentage of P-sites falling in the three possible translation reading frames, stratified by read length. It takes into account only transcripts with annotated regions of length > 0 and the resulting values are visualized as heatmaps. Multiple samples and replicates can be handled in several ways.

Usage

```
frame_psite_length(
  data,
  annotation,
  sample,
  multisamples = "average",
  plot_style = "split",
  transcripts = NULL,
  region = "all",
  length_range = NULL,
  cl = 100,
  colour = "#172969"
)
```

Arguments

data	Either list of data tables or GRangesList object from `psite_info`.
annotation	Data table as generated by `create_annotation`.
sample	Either character string, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled and visualised according to `multisample` and `plot_style`.
multisamples	Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in `sample`:

- if `sample` is a character string vector and `multisample` is set to "average" the elements of the vector considered as replicates of one sample and a single heatmap is returned.
- if `sample` is a character string vector and `multisample` is set to "independent", each element of the vector is analysed independently of the others. The number of plots returned and their organization is specified by `plot_style`.
- if `sample` is list, `multisample` must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. The number of plots returned and their organization is specified by `plot_style`. Note: when this parameter is set to "average" the heatmap associated with each sample displays the length- and frame- specific mean signal computed across the replicates. Default is "average".

plot_style   Either "split" or "facet". It specifies how to organize and display multiple heatmaps:

- "split": heatmap(s) for each sample are returned as an independent ggplot object.
- "facet": the heatmap(s) for each sample are placed one below the other (when region is "all") or one next to the other (when region is set to either "5utr", "cds", "3utr") in independent boxes. Default is "split".

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", respectively for 5' UTRs, CDSs and 3' UTRs, or "all", i.e. all regions are considered. Note: according to this parameter the heatmaps are differently arranged to optimise the organization and the visualization of the data.

length_range   Integer or integer vector for restricting the plot to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used.

cl   Integer value in 1,100 specifying a confidence level for restricting the plot to an automatically-defined range of read lengths. The new range is computed according to the most frequent read lengths, which accounts for the cl% of the sample and is defined by discarding the (100-cl)% of read lengths falling in the tails of the read lengths distribution. If multiple samples are analysed, a single range of read lengths is computed such that at least the cl% of all sample are represented. Default is 100.

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 a linear gradient. Default is dark blue.

Value

List containing: one or more ggplot object(s) and the data table with the corresponding x- and y-axis values and the z-values, defining the color of the tiles ("plot_dt"); an additional data table with raw and scaled number of read extremities mapping around the start and the stop codon, per length, for each sample ("count_dt").

See Also

[frame_psite](#) for a similar plot where read lengths are collapsed.

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## Generate fake samples and replicates
## for(i in 2:6){
##   samp_name <- paste0("Samp", i)
##   set.seed(i)
##   reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
## }
##
## ## Compute and add p-site details
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
```

```
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## ## Define the list of samples and replicate to use as input
## input_samples <- list("S1" = c("Samp1", "Samp2"),
##                       "S2" = c("Samp3", "Samp4", "Samp5"),
##                       "S3" = c("Samp6"))
##
## Generate heatmaps
## example_frames_stratified <- frame_psite_length(reads_psite_list, mm81cdna,
##                                                 sample = input_samples,
##                                                 multisamples = "average",
##                                                 plot_style = "split",
##                                                 region = "all",
##                                                 colour = "#333f50")
```

---

length_filter   *Read length filtering.*

Description

This function provides multiple options for filtering the reads according to their length. Read lengths to keep are either specified by the user or automatichally selected on the basis of the trinucleotide periodicity of reads mapping on the CDS.

Usage

```
length_filter(
  data,
  sample = NULL,
  length_filter_mode = "periodicity",
  periodicity_threshold = 50,
  length_range = NULL,
  output_class = "datatable",
  txt = FALSE,
  txt_file = NULL
)
```

Arguments

data	Either list of data tables or GRangesList object from [bamtolist](), [bedtolist]() or [duplicates_filter]().
sample	Character string or character string vector specifying the name of the sample(s) to process. Default is NULL i.e. all samples are processed.
length_filter_mode	
	Either "periodicity" or "custom". It specifies how read length selection should be performed. "periodicity": only read lengths satisfying a periodicity threshold (see periodicity_threshold) are kept. It ensures the removal of all reads with low or no periodicity; "custom": only read lengths specified by the user are kept (see length_range). Default is "periodicity".

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 set to "periodicity". Default is 50.

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

output_class   Either "datatable" or "granges". It specifies the format of the output i.e. a list of data tables or a GRangesList object. Default is "datatable".

txt   Logical value whether to write in a txt file statistics on the filtering step. Similar information are displayed by default in the console. Default is FALSE.

txt_file   Character string specifying the path, name and extension (e.g. "PATH/NAME.extension") of the plain text file where statistics on the filtering step shuold be written. If the specified folder doesn't exist, it is automatically created. If NULL (the default), the information are written in *"length_filtering.txt"*, saved in the working directory. This parameter is considered only if `txt` is TRUE.

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_range = 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 heatmap-like metaprofiles (metaheatmaps) displaying the abundance of P-sites around the start and the stop codon of annotated CDSs for multiple samples. 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 option always returns one heatmap displaying all the specified samples thus optimizing the visualization of several profiles in a small area.

Usage

```
metaheatmap_psite(
  data,
  annotation,
  sample,
```

```
    multisamples = "average",
    scale_factors = "auto",
    transcripts = NULL,
    length_range = NULL,
    cl = 100,
    utr5l = 25,
    cdsl = 50,
    utr3l = 25,
    colour = "black",
    log_colour = FALSE
)
```

Arguments

data	Either list of data tables or GRangesList object from `psite_info`.
annotation	Data table as generated by `create_annotation`.
sample	Either character string, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled according to `multisamples`.
multisamples	Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in `sample`:

- if `sample` is a character string vector and `multisamples` is set to "average" the elements of the vector are considered as replicates of one sample.
- if `sample` is a character string vector and `multisamples` is set to "independent", each element of the vector is analysed independently of the others.
- if `sample` is list, `multisamples` must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. Note: when this parameter is set to "average" the intensity of the profile associated with each sample reports the nucleotide-specific mean signal computed across the replicates. Default is "average".

scale_factors	Either "auto", a named numeric vector or "none". It specifies how metaprofiles should be scaled before merging multiple replicates (if any):

- "auto": each metaprofile is scaled so that the area under the curve is 1.
- named numeric vector: `scale_factors` must be the same length of unlisted `sample` and each scale factor must be named after the corresponding string in unlisted `sample`. No specific order is required. Each metaprofile is multiplied by the matching scale factor.
- "none": no scaling is applied. Default is "auto".

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.
length_range	Integer or integer vector for restricting the plot to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used. If specified, this parameter prevails over `cl`.

cl   Integer value in 1,100 specifying a confidence level for restricting the plot to
                    an automatically-defined range of read lengths. The new range is computed
                    according to the most frequent read lengths, which accounts for the cl% of the
                    sample and is defined by discarding the (100-cl)% of read lengths falling in the
                    tails of the read lengths distribution. If multiple samples are analysed, a single
                    range of read lengths is computed such that at least the cl% of all sample are
                    represented. Default is 100.

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 flank-
                    ing 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.

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 corre-
                    sponding 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".

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

Value

List containing: one or more ggplot object(s) and the data table with the corresponding x- and y-axis
values ("plot_dt"); an additional data table with raw and scaled number of P-sites per codon in the
selected region for each sample ("count_dt").

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## Generate fake samples and replicates
## for(i in 2:6){
##   samp_name <- paste0("Samp", i)
##   set.seed(i)
##   reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
## }
##
## ## Compute and add p-site details
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## ## Define the list of samples and replicate to use as input
## input_samples <- list("S1" = c("Samp1", "Samp2"),
##                       "S2" = c("Samp3", "Samp4", "Samp5"),
##                       "S3" = c("Samp6"))
##
## Generate metaheatmap
## example_metaheatmap <- metaheatmap_psite(reads_psite_list, mm81cdna,
##                                         sample = input_samples,
##                                         multisamples = "average",
```

```
##                                          utr5l = 20, cdsl = 40, utr3l = 20,
##                                          colour = "#333f50"))
```

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

Description

This function generates metaprofiles displaying the abundance of P-sites around the start and the stop codon of annotated CDSs. For each sample the intensity of the 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. Multiple samples and replicates can be handled.

Usage

```
metaprofile_psite(
  data,
  annotation,
  sample,
  multisamples = "average",
  plot_style = "split",
  scale_factors = "auto",
  transcripts = NULL,
  length_range = NULL,
  cl = 100,
  utr5l = 25,
  cdsl = 50,
  utr3l = 25,
  colour = NULL
)
```

Arguments

data	Either list of data tables or GRangesList object from `psite_info`.
annotation	Data table as generated by `create_annotation`.
sample	Either character string, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled and visualised according to `multisamples` and `plot_style`.
multisamples	Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in `sample`: • if `sample` is a character string vector and `multisamples` is set to "average" the elements of the vector are considered as replicates of one sample and a single metaprofile is returned. • if `sample` is a character string vector and `multisamples` is set to "independent", each element of the vector is analysed independently of the others. The number of plots returned and their organization is specified by `plot_style`.

- if sample is a list, multisamples must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. The number of plots returned and their organization is specified by plot_style. Note: when this parameter is set to "average" the bar plot associated with each sample displays the nucleotide-specific mean signal computed across the replicates and the corresponding standard error is also reported. Default is "average".

plot_style   Either "split", "facet", "overlap" or "mirror". It specifies how to organize and display multiple metaprofiles:

- "split": one metaprofile for each sample is returned as an independent ggplot object;
- "facet": the metaprofiles are placed one below the other, in independent boxes.
- "overlap": the metaprofiles are placed one on top of the other;
- "mirror": sample must be either a character string vector or a list of exactly two elements and the resulting metaprofiles are mirrored along the x axis. Default is "split".

scale_factors   Either "auto", a named numeric vector or "none". It specifies how metaprofiles should be scaled before merging multiple replicates (if any):

- "auto": each metaprofile is scaled so that the area under the curve is 1.
- named numeric vector: scale_factors must be the same length of unlisted sample and each scale factor must be named after the corresponding string in unlisted sample. No specific order is required. Each metaprofile is multiplied by the matching scale factor.
- "none": no scaling is applied. Default is "auto".

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.

length_range   Integer or integer vector for restricting the plot to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used. If specified, this parameter prevails over cl.

cl   Integer value in 1,100 specifying a confidence level for restricting the plot to an automatically-defined range of read lengths. The new range is computed according to the most frequent read lengths, which accounts for the cl% of the sample and is defined by discarding the (100-cl)% of read lengths falling in the tails of the read lengths distribution. If multiple samples are analysed, a single range of read lengths is computed such that at least the cl% of all samples is represented. Default is 100.

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.

colour   Character string or character string vector specifying the colour of the metaprofile(s). If sample is a list of multiple elements and multisamples is set to "average", a colour for each element of the list is required. If this parameter is not specified the R default palette is employed. Default is NULL.

Value

List containing: one or more ggplot object(s) and the data table with the corresponding x- and y-axis values ("plot_dt"); an additional data table with raw and scaled number of P-sites per codon in the selected region for each sample ("count_dt").

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## Generate fake samples and replicates
## for(i in 2:6){
##   samp_name <- paste0("Samp", i)
##   set.seed(i)
##   reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
## }
##
## ## Compute and add p-site details
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## ## Define the list of samples and replicate to use as input
## input_samples <- list("S1" = c("Samp1", "Samp2"),
##                       "S2" = c("Samp3", "Samp4", "Samp5"),
##                       "S3" = c("Samp6"))
##
## ## Generate metaprofiles
## example_metaprofile <- metaprofile_psite(reads_psite_list, mm81cdna,
##                                          sample = input_samples,
##                                          multisamples = "average",
##                                          plot_style = "facet",
##                                          utr5l = 20, cdsl = 40, utr3l = 20,
##                                          colour = c("#333f50", "#39827c", "gray70"))
```

---

mm81cdna *Annotation data table*

Description

A dataset containing basic information about 25,892 mouse mRNAs (Ensembl v81 transcript annotation).

Usage

```
mm81cdna
```

Format

A data table with 25,892 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,
  txt = FALSE,
  txt_file = NULL
)
```

Arguments

data	Either list of data tables or GRangesList object from `bamtolist`, `bedtolist`, `duplicates_filter` 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% of read lengths associated to the highest signals. Default is 99. This parameter is considered only if `plot` is TRUE.
txt	Logical value whether to write in a txt file the extremity used for the correction step and the best offset for each sample. Similar information are displayed by default in the console. Default is FALSE.
txt_file	Character string specifying the path, name and extension (e.g. "PATH/NAME.extension") of the plain text file where the extremity used for the correction step and the best offset for each sample shuold be written. If the specified folder doesn't exist, it is automatically created. If NULL (the default), the information are written in *"best_offset.txt"* and saved in the working directory. This parameter is considered only if `txt` 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")
```

---

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: 1) for transcripts not associated to any annotated CDS the P-site position with respect to the start and the stop codon is set to NA; 2) P-sites of short reads (<20 nts) might be located very close to the 5' or 3' extremity, with no biological meaning and causing potential downstream issues; for these reasons, all read lengths showing this feature will be removed. 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,
  refseq_sep = NULL,
  bsgenome = NULL,
  gtfpath = NULL,
  txdb = NULL,
  dataSource = NA,
  organism = NA,
  output_class = "datatable"
)
```

Arguments

data	Either list of data tables or GRangesList object from [bamtolist](), [bedtolist](), [duplicates_filter]() 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	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 genome asseblies (chromosome sequences) or of 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 genome asseblies (chromosome sequences). If TRUE (the default), an annotation object is required (see `gtfpath` and `txdb`). FALSE implies nucleotide sequences of transcripts are provided instead.

refseq_sep   Character specifying the separator between reference sequences' name and additional information to discard, stored in the headers of the FASTA file specified by `fastapath` (if any). It might be required for matching the reference sequences' identifiers reported in the input list of data tables. All characters before the first occurrence of the specified separator are kept. Default is NULL i.e. no string splitting is performed.

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 genome assemblies (chromosome sequences) 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 genome assemblies (chromosome sequences) 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.

output_class   Either "datatable" or "granges". It specifies the format of the output i.e. a list of data tables or a GRangesList object. Default is "datatable".

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. BAM based on transcript coordinates can be generated in two ways: i) aligning directly against transcript sequences; ii) aligning against sequences of genome

assemblies i.e. standard chromosome sequences, thus requiring the outputs to be translated in tran-
script 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 data table*

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 (intended as the width of the reference sequence region
covered by the RNA fragment. For further information about this choice please refer to section
Details of function bamtolist)

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 cor-
rection step)

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

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

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

sample Name of the sample

---

reads_list   *Reads information data table*

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 further information.

Usage

```
reads_list
```

Format

A list of data tables with 1 object (named *Samp1*) of 100,062 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 (intended as the width of the reference sequence region covered by the RNA fragment. For further information about this choice please refer to section Details of function bamtolist)

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

---

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. Multiple samples and replicates can be handled.

Usage

```
region_psite(
  data,
  annotation,
  sample,
  multisamples = "average",
  plot_style = "stack",
  transcripts = NULL,
```

```
  length_range = NULL,
  cl = 100,
  colour = c("gray70", "gray40", "gray10")
)
```

Arguments

data	Either list of data tables or GRangesList object from `psite_info`.
annotation	Data table as generated by `create_annotation`.
sample	Either character string, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled and visualised according to `multisamples` and `plot_style`.
multisamples	Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in `sample`:

- if `sample` is a character string vector and `multisamples` is set to "average" the elements of the vector are considered as replicates of one sample and a single bar plot is returned.
- if `sample` is a character string vector and `multisamples` is set to "independent", each element of the vector is analysed independently of the others.
- if `sample` is a list, `multisamples` must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. Note: when this parameter is set to "average" the bar plot associated with each sample displays the region-specific mean signal computed across the replicates and, if `plot_style` is set to "dodge", the corresponding standard error is also reported. Default is "average".

plot_style	Either "stack" or "dodge". It specifies how to organize the bars associated with the three regions of the transcript:

- "stack": bars are placed one on top of the other.
- "dodge": bars are placed one next to the other. In this case, the standard error obtained by merging multiple samples (if any, see `sample` and `multisamples`) is displayed. Default is "stack".

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.
length_range	Integer or integer vector for restricting the analysis to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used. If specified, this parameter prevails over `cl`.
cl	Integer value in [1,100] specifying a confidence level for restricting the plot to an automatically-defined range of read lengths. The new range is computed according to the most frequent read lengths, which accounts for the cl% of the sample and is defined by discarding the (100-cl)% of read lengths falling in the tails of the read lengths distribution. If multiple samples are analysed, a single range of read lengths is computed such that at least the cl% of all samples is represented. Default is 100.
colour	Character string vector of three elements specifying the colour of the bar associated with the 5' UTR, CDS and 3' UTR, respectively. Default is a grayscale.

Details

In the plot, "RNAs" reflects the expected read distribution from random fragmentation of all transcripts used in the analysis. It can be used as baseline to asses the enrichment of ribosomes (P-sites) mapping on the CDS with respect to the UTRs. The three bars are based on the cumulative nucleotide length of the 5' UTRs, CDSs and 3' UTRs, respectively, expressed as percentages.

Value

List containing: one ggplot object(s) and the data table with the corresponding x-, y-axis values and the z-values, defining the color of the bars ("plot_dt"); an additional data table with raw and scaled number of P-sites per frame for each sample ("count_dt").

Examples

```
## data(reads_list)
## data(mm81cdna)
##
## ## Generate fake samples and replicates
## for(i in 2:6){
##   samp_name <- paste0("Samp", i)
##   set.seed(i)
##   reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
## }
##
## ## Compute and add p-site details
## psite_offset <- psite(reads_list, flanking = 6, extremity = "auto")
## reads_psite_list <- psite_info(reads_list, psite_offset)
##
## ## Define the list of samples and replicate to use as input
## input_samples <- list("S1" = c("Samp1", "Samp2"),
##                        "S2" = c("Samp3", "Samp4", "Samp5"),
##                        "S3" = c("Samp6"))
##
## Generate bar plot
## example_psite_per_region <- region_psite(reads_psite_list, mm81cdna,
##                                          sample = input_samples,
##                                          multisamples = "average",
##                                          plot_style = "stack",
##                                          cl = 85,
##                                          colour = c("#333f50", "gray70", "#39827c"))
```

---

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. Multiple samples and replicates can be handled.

Usage

```
rends_heat(
  data,
  annotation,
  sample,
  multisamples = "average",
  plot_style = "split",
  scale_factors = "auto",
  transcripts = NULL,
  length_range = NULL,
  cl = 100,
  utr5l = 50,
  cdsl = 50,
  utr3l = 50,
  log_colour = F,
  colour = "black"
)
```

Arguments

data	Either list of data tables or GRangesList object from [bamtolist](#), [bedtolist](#), [length_filter](#) or [psite_info](#).
annotation	Data table as generated by [create_annotation](#).
sample	Either character string, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled and visualised according to multisamples and plot_style.
multisamples	Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in sample:
	• if sample is a character string vector and multisamples is set to "average" the elements of the vector are considered as replicates of one sample and a single heatmap is returned.
	• if sample is a character string vector and multisamples is set to "independent", each element of the vector is analysed independently of the others. The number of plots returned and their organization is specified by plot_style.
	• if sample is a list, multisamples must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. The number of plots returned and their organization is specified by plot_style. Note: when this parameter is set to "average" the heatmap associated with each sample displays the length- and position- specific mean signal computed across the replicates. Default is "average".
plot_style	Either "split" or "facet". It specifies how to organize and display multiple heatmaps:
	• "split": one heatmap for each sample is returned as an independent ggplot object.
	• "facet": the heatmaps are placed one below the other, in independent boxes. Default is "split".

scale_factors   Either "auto", a named numeric vector or "none". It specifies how heatmap values should be scaled before merging multiple replicates (if any):

- "auto": each heatmap is scaled so that the average of all values is 1.
- named numeric vector: scale_factors must be the same length of unlisted sample and each scale factor must be named after the corresponding string in unlisted sample. No specific order is required. Each heatmap value is multiplied by the matching scale factor.
- "none": no scaling is applied. Default is "auto".

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.

length_range   Integer or integer vector for restricting the plot to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used. If specified, this parameter prevails over cl.

cl   Integer value in 1,100 specifying a confidence level for restricting the plot to an automatically-defined range of read lengths. The new range is computed according to the most frequent read lengths, which accounts for the cl% of the sample and is defined by discarding the (100-cl)% of read lengths falling in the tails of the read lengths distribution. If multiple samples are analysed, a single range of read lengths is computed such that at least the cl% of all samples is represented. Default is 100.

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

List containing: one or more ggplot object(s) and the data table with the corresponding x- and y-axis values and the values defining the color of the tiles ("plot_dt"); an additional data table with raw and scaled number of read extremities mapping around the start and the stop codon, per length, for each sample ("count_dt").

Examples

```
data(reads_list)
data(mm81cdna)

## Generate fake samples and replicates
```

```
for(i in 2:6){
  samp_name <- paste0("Samp", i)
  set.seed(i)
  reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
}

## Define the list of samples and replicate to use as input
input_samples <- list("S1" = c("Samp1", "Samp2"),
                      "S2" = c("Samp3", "Samp4", "Samp5"),
                      "S3" = c("Samp6"))

## Generate metaheatmaps for a sub-range of read lengths:
example_ends_heatmap <- rends_heat(reads_list, mm81cdna,
                                   sample = input_samples,
                                   multisamples = "average",
                                   plot_style = "split",
                                   cl = 85,
                                   utr5l = 25, cdsl = 40, utr3l = 25)
```

---

rlength_distr   *Read length distributions.*

Description

This function generates read length distributions, displayed as bar plots. Multiple samples and
replicates can be handled..

Usage

```
rlength_distr(
  data,
  sample,
  multisamples = "average",
  plot_style = "split",
  scale_factors = "auto",
  transcripts = NULL,
  length_range = NULL,
  cl = 100,
  colour = NULL
)
```

Arguments

data	Either list of data tables or GRangesList object from [bamtolist](), [bedtolist](), [length_filter]() or [psite_info]().
sample	Either character string, character string vector or named list of character string(s)/character string vector(s) specifying the name of the sample(s) and replicate(s) of interest. If a list is provided, each element of the list is considered as an independent sample associated with one ore multiple replicates. Multiple samples and replicates are handled and visualised according to multisamples and plot_style.
multisamples	Either "average" or "independent". It specifies how to handle multiple samples and replicates stored in sample:

- if `sample` is a character string vector and `multisamples` is set to "average" the elements of the vector are considered as replicates of one sample and a single bar plot is returned.

- if `sample` is a character string vector and `multisamples` is set to "independent", each element of the vector is analysed independently of the others. The number of plots returned and their organization is specified by `plot_style`.

- if `sample` is a list, `multisamples` must be set to "average". Each element of the list is analysed independently of the others, its replicates averaged and its name reported in the plot. The number of plots returned and their organization is specified by `plot_style`. Note: when this parameter is set to "average" the bar plot associated with each sample displays the length-specific mean signal computed across the replicates and the corresponding standard error is also reported. Default is "average".

`plot_style`   Either "split", "facet", "dodge" or "mirror". It specifies how to organize and display multiple bar plots:

- "split": one bar plot for each sample is returned as an independent ggplot object;

- "facet": the bar plots are placed one next to the other, in independent boxes;

- "dodge": all bar plots are displayed in one box and, for each length, samples are placed side by side.

- "mirror": `sample` must be either a character string vector or a list of exactly two elements and the resulting bar plots are mirrored along the x axis. Default is "split".

`scale_factors`   Either "auto", a named numeric vector or "none". It specifies how read length distributions should be scaled before merging multiple samples (if any):

- "auto": each distribution is scaled so that the sum of all bars is 100.

- named numeric vector: `scale_factors` must be the same length of unlisted `sample` and each scale factor must be named after the corresponding string in unlisted `sample`. No specific order is required. Each distribution is multiplied by the matching scale factor.

- "none": no scaling is applied. Default is "auto".

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

`length_range`   Integer or integer vector for restricting the plot to a chosen range of read lengths. Default is NULL, i.e. all read lengths are used. If specified, this parameter prevails over `cl`.

`cl`   Integer value in 1,100 specifying a confidence level for restricting the plot to an automatically-defined range of read lengths. The new range is computed according to the most frequent read lengths, which accounts for the cl% of the sample and is defined by discarding the (100-cl)% of read lengths falling in the tails of the read lengths distribution. If multiple samples are analysed, a single range of read lengths is computed such that at least the cl% of all samples is represented. Default is 100.

`colour`   Character string or character string vector specifying the colour of the bar plot(s). If `plot_style` is set to either "dodge" or "mirror", a colour for each sample is required. Default is NULL, i.e. the default R colour palette is used.

Value

List containing: one or more ggplot object(s) and the data table with the corresponding x- and y-axis values ("plot_dt"); an additional data table with raw and scaled number of reads per length in each sample ("count_dt").

Examples

```
data(reads_list)

## Generate fake samples and replicates
for(i in 2:6){
  samp_name <- paste0("Samp", i)
  set.seed(i)
  reads_list[[samp_name]] <- reads_list[["Samp1"]][sample(.N, 5000)]
}

## Define the list of samples and replicate to use as input
input_samples <- list("S1" = c("Samp1", "Samp2"),
                      "S2" = c("Samp3", "Samp4", "Samp5"),
                      "S3" = c("Samp6"))

## Generate the length distribution for a sub-range of read lengths:
example_length_dist <- rlength_distr(reads_list,
                                     sample = input_samples,
                                     multisamples = "average",
                                     plot_style = "facet",
                                     cl = 99,
                                     colour = c("#333f50", "#39827c", "gray70"))
```

Index