Package ‘ribor’

May 25, 2024

Title An R Interface for Ribo Files

Version 1.16.0

Description The ribor package provides an R Interface for .ribo files. It provides functionality to read the .ribo file, which is of HDF5 format, and performs common analyses on its contents.

License GPL-3

Encoding UTF-8

LazyData false

Depends R (>= 3.6.0)

biocViews Software, Infrastructure

Imports dplyr, ggplot2, hash, methods, rhdf5, rlang, stats, S4Vectors, tidyr, tools, yaml

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.1.1

VignetteBuilder knitr

Collate 'annotation_functions.R' 'check_functions.R' 'coverage_functions.R' 'ribo_class.R' 'ribo_methods.R' 'create_ribo.R' 'helper_functions.R' 'info_functions.R' 'metagene_functions.R' 'region_count_functions.R' 'ribor.R' 'rnaseq_functions.R'

git_url https://git.bioconductor.org/packages/ribor

git_branch RELEASE_3_19

git_last_commit 80eac3e

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-24

Author Michael Geng [cre, aut],
Hakan Ozadam [aut],
Can Cenik [aut]

Maintainer Michael Geng <michaelgeng@utexas.edu>

Contents

get_coverage . . . 2
get_experiments . . . 4
get_info . . . 5
get_internal_region_coordinates . . . 6
get_internal_region_lengths . . . 7
get_length_distribution . . . 7
get_metadata . . . 9
get_metagene . . . 10
get_original_region_coordinates . . . 12
get_original_region_lengths . . . 13
get_reference_names . . . 13
get_region_coordinates . . . 14
get_region_counts . . . 15
get_region_lengths . . . 17
get_rnaseq . . . 18
get_tidy_metagene . . . 19
plot_length_distribution . . . 21
plot_metagene . . . 23
plot_region_counts . . . 24
rename_default . . . 26
rename_transcripts . . . 27
Ribo-class . . . 28
ribor . . . 30
set_aliases . . . 31

Index **32**

`get_coverage` *Retrieves the coverage data for a given transcript*

Description

The function `get_coverage` generates a DataFrame of coverage data over the length of a given transcript.

Usage

```
get_coverage(
  ribo.object,
  name,
  range.lower = length_min(ribo.object),
  range.upper = length_max(ribo.object),
  length = TRUE,
  tidy = FALSE,
  alias = FALSE,
  compact = TRUE,
```

```
  experiment = experiments(ribo.object)
)
```

Arguments

ribo.object	A 'Ribo' object
name	Transcript Name
range.lower	Lower bound of the read length, inclusive
range.upper	Upper bound of the read length, inclusive
length	Logical value that denotes if the coverage should be summed across read lengths
tidy	Logical value denoting whether or not the user wants a tidy format
alias	Option to accept the transcript input as aliases/nicknames
compact	Option to return a DataFrame with Rle and factor as opposed to a raw data.frame
experiment	List of experiments to obtain coverage information on

Details

The function get_coverage first checks the experiments in the 'experiments' parameter to see if they are present in the .ribo file. It will then check these experiments for coverage data which is an optional dataset. As a result, this function safe guards against experiments that do not have coverage data, but it also, by default, includes all of the experiments in a file in the experiments' parameter.

The function checks the coverage of one transcript at a time at each read length from 'range.lower' to 'range.upper', inclusive. However, the parameter 'length' allows the user to obtain the coverage information of a transcript across the range of read lengths indicated by 'range.lower' and 'range.upper'.

If the ribo.object is generated with aliases, the 'alias' parameter, if set to TRUE, allows the user to use the alias of the transcript as the 'name' parameter instead of the original transcript name.

Value

An annotated DataFrame or data.frame (if the compact parameter is set to FALSE) of the coverage information for the provided list of 'experiments' in the 'experiment' parameter. The returned object will have a length column when the 'length' parameter is set to FALSE, indicating that the user does not want to sum the count information across the range of read lengths. The returned data frame has the option of being tidy, and if the 'tidy' parameter is set to TRUE, a position column will be added. Finally, if the 'alias' parameter is set to TRUE, the alias transcript name must have been provided at the generation of the ribo object, and the function will accept this aliased name in the 'transcript' parameter.

See Also

Ribo to generate the necessary ribo.object parameter

Examples

```
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#get the experiments of interest that also contain coverage data
experiments <- c("Hela_1", "Hela_2", "Hela_3", "WT_1")

#the ribo file contains a transcript named 'MYC'
coverage.data <- get_coverage(ribo.object = sample,
                              name = "MYC",
                              range.lower = 2,
                              range.upper = 5,
                              length = TRUE,
                              experiment = experiments)
```

---

get_experiments *Provides a list of experiments from a .ribo file*

Description

The function get_experiments provides a list of experiment names in the .ribo file.

Usage

```
get_experiments(ribo.object)
```

Arguments

ribo.object	S4 object of class "Ribo"

Details

get_experiments returns a list of strings denoting the experiments. It obtains this by reading directly from the .ribo file through the path of the 'ribo.object' parameter. To generate the param 'ribo.object', call the Ribo function and provide the path to the .ribo file of interest.

The user can then choose to create a subset from this list for any specific experiments of interest for later function calls. Many functions that have the param 'experiment.list' call get_experiments to generate a default list of all experiments in the .ribo file.

Value

A list of the experiment names

See Also

Ribo to generate the necessary ribo.object parameter

Examples

```
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#get a list of the experiments
get_experiments(sample)
```

---

get_info *Get information about the .ribo file*

Description

The function get_info provides information on the attributes, metadata, and datasets of the ribo file.

Usage

```
get_info(ribo.object)
```

Arguments

ribo.object	ribo.object is an S4 object of class "Ribo"

Details

The get_info first provides information on the format version, left_span, right_span, longest read length, shortest read length, metagene_radius, and reference model. The last element of the returned list contains the information about the presence of coverage and RNA-seq data which are optional datasets to include in a .ribo file.

Value

Returns a list containing a nested list of file attributes, a logical value denoting whether the root file has additional metadata, and a data.frame of information on each experiment

See Also

Ribo to generate the necessary ribo.object parameter

Examples

```
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#retrieve information
get_info(sample)
```

get_internal_region_coordinates

Retrieves the region stop and start coordinates

Description

The function get_internal_region_coordinates retrieves the start and site positions for the UTR5, UTR5 Junction, CDS, UTR3 Junction, and UTR3 regions of every transcript.

Usage

```
get_internal_region_coordinates(ribo.object, alias = FALSE)
```

Arguments

ribo.object	A 'Ribo' object
alias	Option to return the transcript names as aliases

Details

To note, because of the R-specific 1-based indexing, the positions start at 1 instead of 0 in other programming languages. The positions provided in the returned data.frame will correspond to the positions in the output of get_coverage.

Additionally, within the transcripts, there are edge cases. NA values found in the returned data.frame means that the region has no start and stop position and a length of zero after computing the boundaries of the UTR5 and UTR3 junction.

Value

A data.frame of start and stop coordinates for every region

Examples

```
# generate a ribo object
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path, rename = rename_default)

# get the region coordinates
coord <- get_internal_region_coordinates(sample, alias = TRUE)
```

get_internal_region_lengths

Returns the overall length of each region with UTR Junctions

Description

The function get_internal_region_coordinates retrieves the lengths for the UTR5, UTR5 Junction, CDS, UTR3 Junction, and UTR3 regions of every transcript.

Usage

```
get_internal_region_lengths(ribo.object, alias = FALSE)
```

Arguments

ribo.object	A 'Ribo' object
alias	Option to return the transcript names as aliases

Value

A data.frame of the region lengths

Examples

```
# generate a ribo object
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path, rename = rename_default)

# get the region lengths
region_lengths <- get_internal_region_lengths(sample, alias = TRUE)
```

get_length_distribution

Retrieves the length distribution of a given region

Description

The function get_length_distribution retrieves the raw or normalized counts at each read length from 'range.lower' to 'range.upper'.

Usage

```
get_length_distribution(
  ribo.object,
  region,
  range.lower = length_min(ribo.object),
  range.upper = length_max(ribo.object),
  compact = TRUE,
  experiment = experiments(ribo.object)
)
```

Arguments

ribo.object	A 'Ribo' object
region	Specific region of interest
range.lower	Lower bound of the read length, inclusive
range.upper	Upper bound of the read length, inclusive
compact	Option to return a DataFrame with Rle and factor as opposed to a raw data.frame
experiment	List of experiment names

Details

This function is a wrapper function of get_region_counts, and the returned DataFrame is valid input for plot_length_distribution.

Value

An annotated DataFrame or data.frame (if the compact parameter is set to FALSE) of the read-length specific region count information for a single region specified in the 'region' parameter. The returned data frame will have a length column, and it will not contain a transcript column.

See Also

plot_length_distribution to plot the output of this function

Examples

```
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#specify the experiments of interest
experiments <- c("Hela_1", "Hela_2", "WT_1")

#gets the normalized length distribution from read length 2 to 5
length.dist <- get_length_distribution(ribo.object = sample,
                                       region = "CDS",
                                       range.lower = 2,
                                       range.upper = 5)
```

get_metadata *Retrieves the metadata of an experiment*

Description

get_metadata provides information on all of the user-inputted metadata of an experiment. If the experiment is not found, then the attributes of the root .ribo file is returned instead.

Usage

```
get_metadata(ribo.object, name = NULL, print = TRUE)
```

Arguments

ribo.object	object of class 'ribo'
name	The name of the experiment
print	Logical value indicating whether or not to neatly print the output

Value

If a valid experiment name is provided, a list of elements providing all of the metadata of the experiment is returned.

If the name is not provided and the root file has metadata, then a list of elements providing all of the metadata found in the root file is returnend.

See Also

Ribo to generate the necessary ribo.object parameter

Examples

```
#ribo object use case
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#the ribo file contains an experiment named 'Hela_1'
get_metadata(sample, "Hela_1")
```

get_metagene *Retrieves the metagene data from a .ribo file*

Description

The function get_metagene returns a data frame that provides the coverage at the positions surrounding the metagene start or stop site.

Usage

```
get_metagene(
  ribo.object,
  site,
  range.lower = length_min(ribo.object),
  range.upper = length_max(ribo.object),
  transcript = TRUE,
  length = TRUE,
  alias = FALSE,
  compact = TRUE,
  experiment = experiments(ribo.object)
)
```

Arguments

ribo.object	A 'Ribo' object
site	"start" or "stop" site coverage
range.lower	Lower bound of the read length, inclusive
range.upper	Upper bound of the read length, inclusive
transcript	Logical value that denotes if the metagene information should be summed across transcripts
length	Logical value that denotes if the metagene information should be summed across read lengths
alias	Option to report the transcripts as aliases/nicknames
compact	Option to return a DataFrame with Rle and factor as opposed to a raw data.frame
experiment	List of experiment names

Details

The dimensions of the returned data frame depend on the parameters range.lower, range.upper, length, and transcript.

The param 'length' condenses the read lengths together. When length is TRUE and transcript is FALSE, the data frame presents information for each transcript across all of the read lengths. That is, each transcript has a value that is the sum of all of the counts across every read length. As a result, information about the transcript at each specific read length is lost.

The param 'transcripts' condenses the transcripts together. When transcript is TRUE and length is FALSE, the data frame presents information at each read length between range.lower and range.upper inclusive. That is, each separate read length denotes the sum of counts from every transcript. As a result, information about the counts of each individual transcript is lost.

If both 'length' and 'transcript' are TRUE, then the resulting data frame prints out one row for each experiment. This provides the metagene information across all transcripts and all reads in a given experiment.

If both length' and 'transcript' are FALSE, no calculations are done to the data, all information is preserved for both the read length and the transcript. The data frame would just present the entire stored raw data from the read length 'range.lower' to the read length 'range.upper' which in most cases would result in a slow run time with a massive DataFrame returned.

When 'transcript' is set to FALSE, the 'alias' parameter specifies whether or not the returned DataFrame should present each transcript as an alias instead of the original name. If 'alias' is set to TRUE, then the returned data frame will contain the aliases rather than the original reference names of the .ribo file.

Value

An annotated DataFrame or data.frame (if the compact parameter is set to FALSE) of the metagene information for either the 'stop' or 'start' site provided in the 'site' parameter. The returned data frame will have a length column when the 'length' parameter is set to FALSE, indicating the returned data frame will have a transcript column whe the 'transcript' parameter is set to FALSE, indicating that the count information will not be summed across the transcripts. In the case that transcript parameter is 'FALSE', the returned data frame will present the transcripts according to the aliases specified at the creation of the ribo object if the 'alias' parameter is set to TRUE.

See Also

Ribo to generate the necessary 'Ribo' class object, plot_metagene to visualize the metagene data, get_tidy_metagene to obtain tidy metagene data under certain conditions

Examples

```
#generate the ribo object by providing the file.path to the ribo file
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)


#extract the total metagene information for all experiments
#across the read lengths and transcripts of the start site
#from read length 2 to 5
metagene_info <- get_metagene(ribo.object = sample,
                              site = "start",
                              range.lower = 2,
                              range.upper = 5,
                              length = TRUE,
                              transcript = TRUE,
                              experiment = experiments(sample))
```

```
#Note that length, transcript, and experiments in this case are the
#default values and can be left out. The following generates the same output.

metagene_info <- get_metagene(ribo.object = sample,
                              site = "start",
                              range.lower = 2,
                              range.upper = 5)
```

---

get_original_region_coordinates

Retrieves the region stop and start coordinates

Description

The function get_original_region_coordinates retrieves the start and site positions for the UTR5, UTR5 Junction, CDS, UTR3 Junction, and UTR3 regions of every transcript.

Usage

```
get_original_region_coordinates(ribo.object, alias = FALSE)
```

Arguments

ribo.object	A 'Ribo' object
alias	Option to return the transcript names as aliases

Details

To note, because of the R-specific 1-based indexing, the positions start at 1 instead of 0 in other programming languages. The positions provided in the returned data.frame will correspond to the positions in the output of get_coverage.

Additionally, within the transcripts, there are edge cases. NA values found in the returned data.frame means that the region has no start and stop position and a length of zero after computing the boundaries of the UTR5 and UTR3 junction.

Value

A data.frame of start and stop coordinates for every region

Examples

```
# generate a ribo object
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path, rename = rename_default)

# get the region coordinates
coord <- get_original_region_coordinates(sample, alias = TRUE)
```

get_original_region_lengths *Returns the overall length of each region*

Description

The function get_original_region_coordinates retrieves the lengths for the UTR5, CDS, and UTR3 regions of every transcript.

Usage

```
get_original_region_lengths(ribo.object, alias = FALSE)
```

Arguments

ribo.object	A 'Ribo' object
alias	Option to return the transcript names as aliases

Value

A data.frame of the region lengths

Examples

```
# generate a ribo object
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path, rename = rename_default)

# get the region coordinates
region_lengths <- get_original_region_lengths(sample, alias = TRUE)
```

get_reference_names *Retrieves a list of reference names*

Description

Gets a list of reference names by reading directly from the .ribo file

Usage

```
get_reference_names(ribo.object)
```

Arguments

ribo.object	A 'Ribo' object-=09

Value

a list of the reference names

Examples

```
#generate a ribo object with transcript nicknames/aliases
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path)

#get the reference names
names <- get_reference_names(sample)
```

---

get_region_coordinates

Retrieves the region stop and start coordinates

Description

The function get_region_coordinates retrieves the start and site positions for the UTR5, UTR5 Junction, CDS, UTR3 Junction, and UTR3 regions of every transcript.

Usage

```
get_region_coordinates(ribo.object, alias = FALSE)
```

Arguments

ribo.object	A 'Ribo' object
alias	Option to return the transcript names as aliases

Details

To note, because of the R-specific 1-based indexing, the positions start at 1 instead of 0 in other programming languages. The positions provided in the returned data.frame will correspond to the positions in the output of get_coverage.

Additionally, within the transcripts, there are edge cases. NA values found in the returned data.frame means that the region has no start and stop position and a length of zero after computing the boundaries of the UTR5 and UTR3 junction.

Value

A data.frame of start and stop coordinates for every region

get_region_counts *Retrieves the region counts from a .ribo file*

Description

get_region_counts will return the particular region counts of any subset of regions for a given set of experiments.

Usage

```
get_region_counts(
  ribo.object,
  range.lower = length_min(ribo.object),
  range.upper = length_max(ribo.object),
  length = TRUE,
  transcript = TRUE,
  tidy = TRUE,
  alias = FALSE,
  normalize = FALSE,
  region = c("UTR5", "UTR5J", "CDS", "UTR3J", "UTR3"),
  compact = TRUE,
  experiment = experiments(ribo.object)
)
```

Arguments

ribo.object	A 'Ribo' object
range.lower	Lower bound of the read length, inclusive
range.upper	Upper bound of the read length, inclusive
length	Logical value that denotes if the region count information should be summed across read lengths
transcript	Logical value that denotes if the region count information should be summed across transcripts
tidy	Option to return the data frame in a tidy format
alias	Option to report the transcripts as aliases/nicknames
normalize	Option to normalize the counts as counts per million reads
region	Specific region of interest
compact	Option to return a DataFrame with Rle and factor as opposed to a raw data.frame
experiment	List of experiment names

Details

This function will return a data frane of the counts at each specified region for each specified experiment. The region options are "UTR5", "UTR5J", "CDS", "UTR3J", and "UTR3". The user can specify any subset of regions in the form of a vector, a list, or a single string if only one region is desired.

The dimensions of the returned DataFrame depend on the parameters range.lower, range.upper, length, and transcript.

The param 'length' condenses the read lengths together. When length is TRUE and transcript is FALSE, the data frame presents information for each transcript across all of the read lengths. That is, each transcript has a value that is the sum of all of the counts across every read length. As a result, information about the transcript at each specific read length is lost.

The param 'transcript' condenses the transcripts together. When transcript is TRUE and length is FALSE data frame presents information at each read length between range.lower and range.upper inclusive. That is, each separate read length denotes the sum of counts from every transcript. As a result, information about the counts of each individual transcript is lost.

When 'transcript' is set to FALSE, the 'alias' parameter specifies whether or not the returned DataFrame should present each transcript as an alias instead of the original name. If 'alias' is set to TRUE, then the column of the transcript names will contain the aliases rather than the original reference names of the .ribo file.

If both 'length' and 'transcript' are TRUE, then the resulting DataFrame prints out one row for each experiment. This provides the metagene information across all transcripts and all reads in a given experiment.

If both length' and 'transcript' are FALSE, calculations are done to the data, all information is preserved for both the read length and the transcript. The DataFrame would just present the entire stored raw data from the read length 'range.lower' to the read length 'range.upper' which in most cases would result in a slow run time with a massive DataFrame returned.

When 'transcript' is set to FALSE, the 'alias' parameter specifies whether or not the returned DataFrame should present each transcript as an alias instead of the original name. If 'alias' is set to TRUE, then the column of the transcript names will contain the aliases rather than the original reference names of the .ribo file.

Value

An annotated DataFrame or data.frame (if the compact parameter is set to FALSE) of the region count information for the regions specified in the 'region' parameter. The returned data frame will have a length column when the 'length' parameter is set to FALSE, indicating that the count information will not be summed across the provided range of read lengths. Similarly, the returned data frame will have a transcript column when the 'transcript' parameter is set to FALSE, indicating that the count information will not be summed across the transcripts. In the case that transcript parameter is 'FALSE', the returned data frame will present the transcripts according to the aliases specified at the creation of the ribo object if the 'alias' parameter is set to TRUE.

Examples

```
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
```

```
sample <- Ribo(file.path)

#specify the regions and experiments of interest
regions <- c("UTR5", "UTR5J", "CDS", "UTR3J", "UTR3")
experiments <- c("Hela_1", "Hela_2", "WT_1")

#obtains the region counts at each individual read length, summed across every transcript
region.counts <- get_region_counts(ribo.object = sample,
                                   region = regions,
                                   range.lower = 2,
                                   range.upper = 5,
                                   length = FALSE,
                                   transcript = TRUE,
                                   tidy = FALSE,
                                   alias = FALSE,
                                   experiment = experiments)
```

`get_region_lengths` *Returns the overall length of each region with UTR Junctions*

Description

The function `get_region_lengths` retrieves the lengths for the UTR5, UTR5 Junction, CDS, UTR3 Junction, and UTR3 regions of every transcript.

Usage

```
get_region_lengths(ribo.object, alias = FALSE)
```

Arguments

`ribo.object`	A 'Ribo' object
`alias`	Option to return the transcript names as aliases

Details

This function is deprecated, and we recommend `get_internal_region_lengths`.

Value

A data.frame of the region lengths

get_rnaseq *Information on the RNA-Seq data of the experiments, if any*

Description

get_rnaseq returns a data frame containing information on the transcript name, experiment, and sequence abundance

Usage

```
get_rnaseq(
  ribo.object,
  tidy = TRUE,
  region = c("UTR5", "UTR5J", "CDS", "UTR3J", "UTR3"),
  experiment = experiments(ribo.object),
  compact = TRUE,
  alias = FALSE
)
```

Arguments

ribo.object	A 'Ribo' object
tidy	Option to return the data frame in a tidy format
region	Specific region(s) of interest
experiment	List of experiment names
compact	Option to return a DataFrame with Rle and factor as opposed to a raw data.frame
alias	Option to report the transcripts as aliases/nicknames

Details

As a default value, experiment.list is presumed to include all of the experiments within a ribo file. RNA-Seq data is an optional dataset to include in a .ribo file. The experiments in experiment.list are checked for experiment existence in the ribo file and then checked for RNA-seq data.

The returned DataFrame can either be in the tidy format for easier data cleaning or in a condensed non-tidy format. The data will present RNA-seq counts for each transcript in each valid experiment in experiment.list.

The 'alias' parameter specifies whether or not the returned DataFrame should present each transcript as an alias instead of the original name. If 'alias' is set to TRUE, then the column of the transcript names will contain the aliases rather than the original reference names of the .ribo file.

Value

An annotated data frame containing the RNA-Seq counts for the regions in specified in the 'region' parameter with the option of presenting the data in a tidy format. Additionally, the function returns a DataFrame with Rle and factor applied if the 'compact' parameter is set to TRUE and a data.frame without any Rle or factor if the 'compact' parameter is set to FALSE

See Also

Ribo to generate the necessary ribo.object parameter

Examples

```
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#list out the experiments of interest that have RNA-Seq data
experiments <- c("Hela_1", "Hela_2", "WT_1")
regions <- c("UTR5", "CDS", "UTR3")
rnaseq.data <- get_rnaseq(ribo.object = sample,
                          tidy = TRUE,
                          region = regions,
                          experiment = experiments)
```

---

get_tidy_metagene *Retrieves the metagene data in a tidy format*

Description

The function get_tidy_metagene provides the user with a tidy data format for easier data cleaning and manipulation. In providing this functionality while reducing the returned data frame size, the user must aggregate across the transcripts and is only provided the option to aggregate the read lengths together.

Usage

```
get_tidy_metagene(
  ribo.object,
  site,
  range.lower = length_min(ribo.object),
  range.upper = length_max(ribo.object),
  length = TRUE,
  compact = TRUE,
  experiment = experiments(ribo.object)
)
```

Arguments

ribo.object	A 'Ribo' object
site	"start" or "stop" site coverage
range.lower	Lower bound of the read length, inclusive
range.upper	Upper bound of the read length, inclusive

length	Logical value that denotes if the metagene information should be summed across read lengths
compact	Option to return a DataFrame with Rle and factor as opposed to a raw data.frame
experiment	List of experiment names

Details

The dimensions of the returned data frame depend on the parameters range.lower, range.upper, and length.

The param 'length' condenses the read lengths together. When length is TRUE, then the resulting data frame prints out one row for each experiment. This provides a tidy format of the metagene information across all transcripts and all read lengths in a given experiment. Each row in the data frame represents the total metagene coverage count of a given experiment at a given position.

When the param 'length' is FALSE, then the resulting data frame prints out the metagene coverage count at each position of the metagene radius for each read length. This provides a tidy format of the metagene information across the transcripts, preserving the metagene coverage count at each read length.

Value

An annotated, tidy DataFrame or data.frame (if the compact parameter is set to FALSE) of the metagene information for either the 'stop' or 'start' site provided in the 'site' parameter. The data frame, as a result of its tidy property, will have a position column. The returned data frame will have a length column when the 'length' parameter is set to FALSE, indicating will be automatically aggregated to keep the memory footprint of this function reasonable.

See Also

Ribo to generate the necessary 'Ribo' class object. plot_metagene to visualize the metagene data, get_metagene to obtain tidy metagene data under certain conditions

Examples

```
#generate the ribo object by loading in a ribo function and calling the \code{\link{Ribo}} function
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#extract the total metagene information in a tidy format
#for all experiments across the read lengths and transcripts
#of the start site from read length 2 to 5

metagene_info <- get_tidy_metagene(ribo.object = sample,
                                   site = "start",
                                   range.lower = 2,
                                   range.upper = 5,
                                   length = TRUE,
                                   experiment = experiments(sample))

#Note that length and experiments in this case are the
#default values and can be left out. The following generates the same output.
```

```
metagene_info <- get_tidy_metagene(ribo.object = sample,
                                   site = "start",
                                   range.lower = 2,
                                   range.upper = 5)
```

---

plot_length_distribution

Plots the length distribution

Description

The function plot_length_distribution can take either a DataFrame or a "Ribo" object to generate a line graph of the length distributions from range.lower to range.upper.

Usage

```
plot_length_distribution(
  x,
  region,
  experiment,
  range.lower,
  range.upper,
  fraction = FALSE,
  title = "Length Distribution"
)
```

Arguments

x	A 'Ribo' object or a DataFrame generated from get_region_counts
region	the region of interest
experiment	a list of experiment names
range.lower	a lower bounds for a read length range
range.upper	an upper bounds for a read length range
fraction	logical value that, if TRUE, presents the count as a fraction of the total reads in the given ranges
title	a title for the generated plot

Details

The param 'fraction' will plot the fractions of each length relative to the total sum of the read length range provided by param 'range.lower' and 'range.upper'. When fraction is set to FALSE, the total count of each read length is plotted.

When given a "Ribo" object, plot_length_distribution calls get_region_counts to retrieve the necessary information for plotting.

The user can instead provide a DataFrame with the same structure as the output of the get_region_counts function where the 'transcript' parameter is set to FALSE and 'length' parameters is the default value of TRUE. This also means that the many of the remaining parameters of the plot_length_distribution function are not necessary. The run time becomes substantially faster when plot_region_counts is given the direct DataFrame to plot. Note that there is no manipulation by this function on the DataFrame. This responsibility is given to the user and allows for more control.

Value

A 'ggplot' of the length distribution

See Also

get_region_counts to generate a DataFrame that can be provided as input, Ribo to create a ribo.object that can be provided as input

Examples

```
#ribo object use case

#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#specify experiments of interest
experiments <- c("Hela_1", "Hela_2", "WT_1")

plot_length_distribution(x = sample,
                         region = "CDS",
                         range.lower = 2,
                         range.upper = 5,
                         experiment = experiments,
                         fraction = TRUE)


#DataFrame use case
#obtains the region counts at each individual read length, summed across every transcript
region.counts <- get_length_distribution(ribo.object = sample,
                                         region      = "CDS",
                                         range.lower = 2,
                                         range.upper = 5,
                                         experiment  = experiments)

#the param 'length' must be set to FALSE and param 'transcript' must be set
#to TRUE to use a DataFrame
plot_length_distribution(region.counts)
```

plot_metagene *Plots the metagene coverage data*

Description

The function plot_metagene plots the metagene site coverage, separating by experiment.

Usage

```
plot_metagene(
  x,
  site,
  experiment,
  range.lower,
  range.upper,
  normalize = FALSE,
  title = "Metagene Site Coverage",
  tick = 10
)
```

Arguments

x	A 'Ribo' object or a data frame generated from get_metagene
site	"start" or "stop" site
experiment	list of experiments
range.lower	lower bound of the read length, inclusive
range.upper	upper bound of the read length, inclusive
normalize	When TRUE, normalizes the data by the total reads.
title	title of the generated plot
tick	x-axis labeling increment

Details

If a DataFrame is provided as param 'x', then the only additional parameter is the optional title' parameter for the generated plot. If a ribo.object is provided as param 'x', the rest of the parameters listed are necessary.

When given a ribo class object, the plot_metagene function generates a DataFrame by calling the get_tidy_metagene function, so the run times in this case will be mostly comprised of a call to the get_metagene function.

This function uses ggplot in its underlying implementation.

Value

A 'ggplot' of the metagene site coverage

Examples

```
#a potential use case is to directly pass in the ribo object file as param 'x'

#generate the ribo object to directly use
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#specify experiments of interest
experiments <- c("Hela_1", "Hela_2", "WT_1")

#plot the metagene start site coverage for all experiments in 'sample.ribo'
#from read length 2 to 5
plot_metagene(x = sample,
              site = "start",
              range.lower = 2,
              range.upper = 5,
              experiment = experiments)

#Note that the site, range.lower, range.upper, and experiment parameter are only
#necessary if a ribo object is being passed in as param 'x'. If a ribo
#object is passed in, then the param 'experiments' will be set to all of
#the experiments by default.

#If a DataFrame is passed in, then the plot_metagene function
#does not need any other information. All of the elements of the DataFrame
#will be used, assuming that it contains the same column names and number of
#columns as the output from get_tidy_metagene()

#gets the metagene start site coverage from read length 2 to 5
#note that the data must be summed across transcripts and read lengths
#for the plot_metagene function
data <- get_tidy_metagene(sample,
                          site = "start",
                          range.lower = 2,
                          range.upper = 5)

#plot the metagene data
plot_metagene(data)
```

`plot_region_counts` *Plots the region counts of UTR5, CDS, and UTR3*

Description

The function `plot_region_counts` can take either a DataFrame or a "Ribo" object to generate the a stacked bar plot of proportions that correspond to the "UTR5", "CDS", and "UTR3" regions.

Usage

```
plot_region_counts(
  x,
  experiment,
  range.lower,
  range.upper,
  title = "Region Counts"
)
```

Arguments

x	A 'Ribo' object or a DataFrame generated from get_region_counts
experiment	a list of experiment names
range.lower	a lower bounds for a read length range
range.upper	an upper bounds for a read length range
title	a title for the generated plot

Details

When given a 'Ribo' object, plot_region_counts calls get_region_counts to retrieve the necessary information for plotting. This option is in the case that a DataFrame of the region count information is not required.

The user can instead provide a DataFrame with the same structure as the output of the get_region_counts function where the 'transcript' and 'length' parameters are the default values of TRUE. This also means that the remaining parameters of the plot_region_counts function are not necessary. The run time becomes substantially faster when plot_region_counts is given the direct DataFrame to plot. However, the DataFrame needs to follow the format and types in the output of the reading functions

Value

A 'ggplot' of the region counts for each of the experiments

See Also

get_region_counts to generate a DataFrame that can be provided as input, Ribo to create a ribo.object that can be provided as input

Examples

```
#ribo object use case
#generate the ribo object
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

#specify the regions and experiments of interest
regions <- c("UTR5", "CDS", "UTR3")
experiments <- c("Hela_1", "Hela_2", "WT_1")
```

```
plot_region_counts(sample,
                   range.lower = 2,
                   range.upper = 5,
                   experiments)

#DataFrame use case
#obtains the region counts at each individual read length, summed across every transcript
region.counts <- get_region_counts(sample,
                                   region = regions,
                                   range.lower = 2,
                                   range.upper = 5,
                                   tidy = TRUE,
                                   length = TRUE,
                                   transcript = TRUE)

#the params 'length' and 'transcript' must be set to true to use a DataFrame
plot_region_counts(region.counts)
```

`rename_default` *Rename function for appris transcriptome naming convention*

Description

The function `rename_default` is the default renaming function for the appris human transcriptome. It takes one single transcript name and returns a simplified alias.

Usage

```
rename_default(x)
```

Arguments

x	Character denoting original name of the transcript

Value

Character denoting simplified name of the object

Examples

```
original <- paste("ENST00000613283.2|ENSG00000136997.17|",
                  "OTTHUMG00000128475.8|-|MYC-206|MYC|1365|protein_coding|",
                  sep = "")
alias <- rename_default(original)
```

rename_transcripts *Renames the transcripts*

Description

The function rename_transcripts strives to make the transcript names less cumbersome to write and easier to use.

Usage

```
rename_transcripts(ribo, rename)
```

Arguments

ribo	a path to the ribo file or a 'Ribo' object
rename	A function that renames the original transcript or an already generated character vector of aliases

Details

Transcript names found in a .ribo file can often be long and inconvenient to use. As a result, this function allows the user to rename the transcripts.

Often times, a short function can be used on the ribo file reference names to split and extract a more convenient name, and a function with a similar input and output to rename_default can be passed in.

However, if there is no simple function that takes the original name and renames it into a unique alias, then the user can provide a character vector of the same length as the number of transcripts in the ribo file. This character vector would provide aliases that match the order of the original reference names returned by the get_reference_names function.

Value

A character vector denoting the renamed transcript aliases

See Also

rename_default to view expected input and output of a 'rename' function Ribo to generate a ribo object

Examples

```
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path, rename = rename_default)

aliases <- rename_transcripts(sample, rename = rename_default)
```

Ribo-class *Ribo Class*

Description

The Ribo object serves as the main utility vehicle for the ribor package. Specifically, it allows the user to interface with a .ribo file in the R ribor rely on the Ribo object to read, visualize, and inspect the contents of the .ribo file. The information stored in this object include the .ribo file path, the list of experiments, the format version, the reference model, the minimum read length, maximum read length, the left span, the right span, and other transcript information.

Usage

```
## S4 method for signature 'Ribo'
show(object)

## S4 method for signature 'Ribo'
path(object)

## S4 method for signature 'Ribo'
experiments(object)

## S4 method for signature 'Ribo'
format_version(object)

## S4 method for signature 'Ribo'
reference(object)

## S4 method for signature 'Ribo'
length_min(object)

## S4 method for signature 'Ribo'
length_max(object)

## S4 method for signature 'Ribo'
left_span(object)

## S4 method for signature 'Ribo'
right_span(object)

## S4 method for signature 'Ribo'
metagene_radius(object)

## S4 method for signature 'Ribo'
length_offset(object)

## S4 method for signature 'Ribo'
```

```
has_metadata(object)

## S4 method for signature 'Ribo'
experiment_info(object)

## S4 method for signature 'Ribo'
transcript_info(object)

## S4 method for signature 'Ribo'
alias_hash(object)

## S4 method for signature 'Ribo'
original_hash(object)

Ribo(path, rename = NULL)
```

Arguments

object	Ribo object
path	The path to the .ribo file
rename	A function that renames the original transcript or an already generated character vector of aliases

Details

Note that the path parameter takes in a file path and stores it. While using the package, be sure to not to move or change the location of the .ribo file. The default names of the transcripts may be difficult to use depending on the settings used to generate the .ribo file. As a result, we have provided a rename parameter that integrates well with the Appris reference transcriptome. Users may also define a simple function that processes a given default transcript name in a one-to-one manner to another custom alias.

Value

Returns an S4 object of class "Ribo" containing a path to the HDF5 file, various attributes in the root folder, and information about the transcripts such as names and lengths

See Also

If a ribo object is already generated but aliases want to be added or updated, use the set_aliases function.

Examples

```
file.path <- system.file("extdata", "sample.ribo", package = "ribor")
sample <- Ribo(file.path)

show(sample)
```

```
#generate a ribo object with transcript nicknames/aliases
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path, rename = rename_default )
```

---

ribor *ribor: A package for reading .ribo files*

Description

The 'ribor' package offers a suite of reading functions for the datasets present in a .ribo file and also provides some rudimentary plotting functions.

Vignette

To get started with the ribor package, please see the vignette page at https://ribosomeprofiling.github.io/ribor/ribor.html.

Related Tools

The paper associated with the Ribo ecosystem can be found at https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/btaa028/5701654.

For more information on the preprocessing pipeline, please see the link to the source code at https://github.com/ribosomeprofiling/riboflow.

For more information on the .ribo file format, please see its documentation page at https://ribopy.readthedocs.io/en/latest/ribo_file_format.html.

For an alternative to ribor, please see a link to source code of ribopy, a python interface, at https://github.com/ribosomeprofiling/ribopy.

Package Content

Generating a ribo object: `Ribo` to get started

Length Distribution: `get_length_distribution` to get length distribution counts
`plot_length_distribution` to plot the length distribution

Region Counts: `get_region_counts` to get region counts
`plot_region_counts` to plot the region counts

Metagene Coverage: `get_metagene` to get metagene site coverage
`get_tidy_metagene` to get a tidy format of the metagene site coverage
`plot_metagene` to plot the metagene site coverage

set_aliases — *Set the aliases of a ribo object*

Description

The function set_aliases allows the user to add aliases to a valid ribo object.

Usage

```
set_aliases(ribo.object, rename)
```

Arguments

ribo.object	A 'ribo' object
rename	A function that renames original transcript name into an alias

Details

If there is a different naming convention from the default appris transcriptome, there may be no simple way to generate convenient aliases from the original reference names. As a result, the user can first generate the ribo object and get the reference names, use custom (and likely more intricate) functions to generate a list of aliases, and then pass in a character vector of these aliases. The character vector should match the order of and correspond to the list of reference names retrieved from get_reference_names

Value

A modified 'ribo' object that contains alias information

Examples

```
#generate a ribo object with transcript nicknames/aliases
file.path <- system.file("extdata", "HEK293_ingolia.ribo", package = "ribor")
sample <- Ribo(file.path)
sample <- set_aliases(ribo.object = sample,
                      rename = rename_default)
```

Index

alias_hash *(Ribo-class)*, 28
alias_hash,Ribo-method *(Ribo-class)*, 28

experiment_info *(Ribo-class)*, 28
experiment_info,Ribo-method *(Ribo-class)*, 28
experiments *(Ribo-class)*, 28
experiments,Ribo-method *(Ribo-class)*, 28

format_version *(Ribo-class)*, 28
format_version,Ribo-method *(Ribo-class)*, 28

get_coverage, *2*, 2, *3*, *6*, *12*, *14*
get_experiments, *4*, 4
get_info, *5*, 5
get_internal_region_coordinates, *6*, 6, *7*
get_internal_region_lengths, 7, *17*
get_length_distribution, *7*, 7, *30*
get_metadata, *9*, 9
get_metagene, *10*, 10, *20*, *23*, *30*
get_original_region_coordinates, *12*, 12, *13*
get_original_region_lengths, 13
get_reference_names, 13, *27*, *31*
get_region_coordinates, *14*, 14
get_region_counts, *8*, *15*, 15, *21*, *22*, *25*, *30*
get_region_lengths, *17*, 17
get_rnaseq, *18*, 18
get_tidy_metagene, *11*, *19*, 19, *23*, *30*

has_metadata *(Ribo-class)*, 28
has_metadata,Ribo-method *(Ribo-class)*, 28

left_span *(Ribo-class)*, 28
left_span,Ribo-method *(Ribo-class)*, 28
length_max *(Ribo-class)*, 28
length_max,Ribo-method *(Ribo-class)*, 28
length_min *(Ribo-class)*, 28
length_min,Ribo-method *(Ribo-class)*, 28
length_offset *(Ribo-class)*, 28
length_offset,Ribo-method *(Ribo-class)*, 28

metagene_radius *(Ribo-class)*, 28
metagene_radius,Ribo-method *(Ribo-class)*, 28

original_hash *(Ribo-class)*, 28
original_hash,Ribo-method *(Ribo-class)*, 28

path *(Ribo-class)*, 28
path,Ribo-method *(Ribo-class)*, 28
plot_length_distribution, *8*, *21*, 21, *22*, *30*
plot_metagene, *11*, *20*, *23*, 23, *30*
plot_region_counts, *22*, *24*, 24, *25*, *30*

reference *(Ribo-class)*, 28
reference,Ribo-method *(Ribo-class)*, 28
rename_default, *26*, 26, *27*
rename_transcripts, *27*, 27
Ribo, *3–5*, *9*, *11*, *19*, *20*, *22*, *25*, *27*, *30*
Ribo *(Ribo-class)*, 28
Ribo-class, 28
ribor, 30
right_span *(Ribo-class)*, 28
right_span,Ribo-method *(Ribo-class)*, 28

set_aliases, *29*, *31*, 31
show,Ribo-method *(Ribo-class)*, 28

transcript_info *(Ribo-class)*, 28
transcript_info,Ribo-method *(Ribo-class)*, 28