Package 'rifi'

October 11, 2022

Title 'rifi' anyalyses data from rifampicin time series ceated by microarray or RNAseq

Version 1.0.0

Description 'rifi' analyses data from rifampicin time series created by microarray or RNAseq.
'rifi' is a transcriptome data analysis tool for the holistic identification of transcription and decay associated processes.
The decay constants and the delay of the onset of decay is fitted for each probe/bin. Subsequently, probes/bins of equal properties
are combined into segments by dynamic programming, independent of a existing genome annotation. This allows to detect transcript segments
of different stability or transcriptional events within one annotated gene.
In addition to the classic decay constant/half-
life analysis, 'rifi' detects processing sites, transcription pausing sites, internal transcription start sites in operons, sites of partial transcription termination in operons, identifies areas of likely transcriptional interference by the collision mechanism
and gives an estimate of the transcription velocity.
All data are integrated to give an estimate of continous transcriptional units, i.e. operons.
Comprehensive output tables and visualizations of the full genome result and the individual fits for all probes/bins are produced.

Depends R (>= 4.1)

Imports car, cowplot, doMC, parallel, dplyr, egg, foreach, ggplot2, graphics, grDevices, grid, methods, nls2, nnet, rlang, S4Vectors, scales, stats, stringr, SummarizedExperiment, tibble, rtracklayer, utils

Suggests DescTools, knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

biocViews RNASeq, DifferentialExpression, GeneRegulation, Transcriptomics, Regression, Microarray, Software

BugReports https://github.com/CyanolabFreiburg/rifi

License GPL-3 + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.12

Language en-US

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

git_branch RELEASE_3_15

git_last_commit a495442

git_last_commit_date 2022-04-27

Date/Publication 2022-10-11

Author Jens Georg [aut, cre]

Maintainer Jens Georg <jens.georg@biologie.uni-freiburg.de>

R topics documented:

`apply_ancova` *apply_ancova: is a statistical test to check variances between 2 segments showing pausing site (ps) or internal starting site (ITSS) independently. apply_ancova: is a statistical test to check if fragments showing ps and ITSS events have significant slope using Ancova test. The function uses ancova test. Ancova is applied when the data contains independent variables, dependent variables and covariant variables. In this case, segments are independent variables, position is the dependent variable and the delay is the covariant. The dataframe is prepared as depicted below. The lm fit is applied and p_value is extracted delay position segment S1 S1 S1 S1 S2 S2 S2 S2*

Description

apply_ancova: is a statistical test to check variances between 2 segments showing pausing site (ps) or internal starting site (ITSS) independently. apply_ancova: is a statistical test to check if fragments showing ps and ITSS events have significant slope using Ancova test. The function uses ancova test. Ancova is applied when the data contains independent variables, dependent variables and covariant variables. In this case, segments are independent variables, position is the dependent variable and the delay is the covariant. The dataframe is prepared as depicted below. The lm fit is applied and p_value is extracted delay position segment S1 S1 S1 S1 S2 S2 S2 S2

Usage

```
apply_ancova(inp)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.

Value

the SummarizedExperiment with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

delay: The delay value of the bin/probe

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

Examples

```
data(stats_minimal)
apply_ancova(inp = stats_minimal)
```

`apply_event_position` *apply_event_position: is a short version of apply_Ttest_delay function to extract event time duration as pausing site or iTSS happens.*

Description

apply_event_position adds a new column with the duration.

Usage

```
apply_event_position(inp)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.

Value

the SummarizedExperiment with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

delay: The delay value of the bin/probe

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

event_position:

Examples

```
data(stats_minimal)
apply_event_position(inp = stats_minimal)
```

`apply_manova` *apply_manova: this function checks if the ratio of hl ratio and intensity ratio is statistically significant. apply_manova compares the variance between two fold-changes,HL and intensity within the same TU (half-life frgA/half-life frgB/ intensity frgA/intensity frgB). HL fragment could cover two intensity fragments therefore this function sets first fragments borders and uses manova_function. Manova checks the variance between 2 segments (independent variables) and two dependents variables (HL and intensity).*

Description

The functions used is: manova_function: applies Manova statistical test. The dataframe template is depicted below. The lm fit is applied and p_value is extracted. half_life intensity segment 1.10479637 1244.078 S1 1.19222097 1894.595 S1 1.16218422 1668.416 S1 1.08733743 1428.831 S1 0.72964160 1381.102 S2 0.06750874 2429.843 S2 0.61911329 1749.105 S2 0.51840661 1122.775 S2

Usage

```
apply_manova(inp)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.

Value

the probe data frame with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

half_life: The half-life of the bin/probe

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

event_duration:

event_position:

FC_HL:

FC_fragment_HL:

p_value_HL:

FC_intensity:

FC_fragment_intensity:

p_value_intensity:

FC_HL_intensity:

FC_HL_intensity_fragment:

FC_HL_adapted:

synthesis_ratio:

synthesis_ratio_event:

p_value_Manova:

Examples

```
data(stats_minimal)
apply_manova(inp = stats_minimal)
```

`apply_Ttest_delay` *apply_Ttest_delay: is a statistical test to check the significance of the point between 2 segments showing pausing site (ps) and internal starting site (ITSS) independently. apply_Ttest_delay uses t-test. The last point from the first segment and the first point from the second segment are selected and added to the residuals of each model. The sum is subjected to t-test.*

Description

apply_Ttest_delay: is a statistical test to check the significance of the point between 2 segments showing pausing site (ps) and internal starting site (ITSS) independently. apply_Ttest_delay uses t-test. The last point from the first segment and the first point from the second segment are selected and added to the residuals of each model. The sum is subjected to t-test.

Usage

```
apply_Ttest_delay(inp)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.

Value

the SummarizedExperiment with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

delay: The delay value of the bin/probe

delay_fragment: The delay fragment the bin belongs to

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

Examples

```
data(stats_minimal)
apply_Ttest_delay(inp = stats_minimal)
```

---

apply_t_test *apply_t_test: it uses the statistical t_test to check if the fold-change of half-life (HL) fragments and the fold-change intensity fragments respectively are significant.*

Description

apply_t_test compares the mean of two neighboring fragments within the same TU to check if the fold-change is significant. Fragments with distance above threshold are not subjected to t-test. Dataframes with less than 3 rows are excluded.

Usage

```
apply_t_test(inp, threshold = 300)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
threshold	integer: threshold.

Details

The functions used are:

1. fragment_function: checks number of fragments inside TU, less than 2 are excluded otherwise they are gathered for analysis.
2. t_test_function: exclude dataframes with less than 3 rows, makes fold-change and apply t-test, assign fragments names and ratio, add columns with the corresponding p_values.

Value

the SummarizedExperiment with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

half_life: The half-life of the bin/probe

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

event_duration:

event_position:

FC_HL:

FC_fragment_HL:

p_value_HL:

FC_intensity:

FC_fragment_intensity:

p_value_intensity:

Examples

```
data(stats_minimal)
apply_t_test(inp = stats_minimal, threshold = 300)
```

`apply_t_test_ti` *apply_t_test_ti: compares the mean of two neighboring TI fragments within the same TU. apply_t_test_ti: this function uses the statistical t_test to check if two neighboring TI fragments are significant.*

Description

apply_t_test_ti: compares the mean of two neighboring TI fragments within the same TU. apply_t_test_ti: this function uses the statistical t_test to check if two neighboring TI fragments are significant.

Usage

```
apply_t_test_ti(inp)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.

Value

the SummarizedExperiment with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

flag: Information on which fitting model is applied

position_segment: The position based segment

TI_termination_factor: The termination factor of the bin/probe

TU: The overarching transcription unit

TI_termination_fragment: The TI fragment the bin belongs to

TI_mean_termination_factor: The mean termination factor of the respective TI fragment

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

event_duration:

event_position:

FC_HL:

FC_fragment_HL:

p_value_HL:

FC_intensity:

FC_fragment_intensity:

p_value_intensity:

FC_HL_intensity:

FC_HL_intensity_fragment:

FC_HL_adapted:

synthesis_ratio:

synthesis_ratio_event:

p_value_Manova:

p_value_TI:

TI_fragments_p_value:

Examples

```
data(stats_minimal)
apply_t_test_ti(inp = stats_minimal)
```

`check_input` *check_input: reviews the input given by the user. 'check_input' stops the operation if the input data frame has severe faults. Less severe faults lead to the removal of wrong IDs and a warnings describing the problem. The Summarized Experiment colData must have the columns "timepoint" with the timepoints convertible to numeric and containing the timepoint 0. If replicates are used the column in colData must be called "replicate". The replicate must be convertible to numeric. In the RowRanges, optionally, IDs can be given as character (except ",","\|","_"),but need to refer to a unique position/strand combination. Strand information need to be given. The relative intensity in the assay must be numeric. The relative intensity for the first time point cannot be 0 or NA.*

Description

check_input: reviews the input given by the user. 'check_input' stops the operation if the input data frame has severe faults. Less severe faults lead to the removal of wrong IDs and a warnings describing the problem. The Summarized Experiment colData must have the columns "timepoint" with the timepoints convertible to numeric and containing the timepoint 0. If replicates are used the column in colData must be called "replicate". The replicate must be convertible to numeric. In the RowRanges, optionally, IDs can be given as character (except ",","|","_"),but need to refer to a unique position/strand combination. Strand information need to be given. The relative intensity in the assay must be numeric. The relative intensity for the first time point cannot be 0 or NA.

Usage

```
check_input(inp, thrsh = 0)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
thrsh	numeric: the minimal allowed intensity for time point "0".

Value

the SummarizedExperiment object: checked, and with position, ID and filtration added to the rowRanges.

Examples

```
data(example_input_minimal)
check_input(inp = example_input_minimal, thrsh = 0)
```

`dataframe_summary` *dataframe_summary: creates two tables relating gene annotation to fragments. dataframe_summary creates two tables summary of segments and their half-lives. The first output is bin/probe features and the second one is intensity fragment based. The dataframe_summary creates one table with feature_type, gene, locus_tag, position, strand, TU, delay_fragment, HL_fragment, half_life, intensity_fragment, intensity and velocity. The second table is similar to the first one but in compact form. It contains the same columns, the only difference is on position where a start and end position are indicated separately. Strand is indicated in case of stranded data to select the corresponding positions.*

Description

dataframe_summary: creates two tables relating gene annotation to fragments. dataframe_summary creates two tables summary of segments and their half-lives. The first output is bin/probe features and the second one is intensity fragment based. The dataframe_summary creates one table with feature_type, gene, locus_tag, position, strand, TU, delay_fragment, HL_fragment, half_life, intensity_fragment, intensity and velocity. The second table is similar to the first one but in compact form. It contains the same columns, the only difference is on position where a start and end position are indicated separately. Strand is indicated in case of stranded data to select the corresponding positions.

Usage

```
dataframe_summary(data, input)
```

Arguments

data	SummarizedExperiment: the input data frame with correct format.
input	dataframe: dataframe from event_dataframe function.

Value

bin_df: all information regarding bins:

feature_type:
gene:
locus_tag:
TU: The overarching transcription unit
delay_fragment: The delay fragment the bin belongs to
HL_fragment: The half-life fragment the bin belongs to
half_life: The half-life of the bin/probe
intensity_fragment: The intensity fragment the bin belongs to
intensity: The relative intensity at time point 0
velocity: The velocity value of the bin

frag_df: all information regarding fragments:

feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
delay_fragment: The delay fragment the bin belongs to
HL_fragment: The half-life fragment the bin belongs to
half_life: The half-life of the fragment
intensity_fragment: The intensity fragment the bin belongs to
intensity: The relative intensity at time point 0
velocity: The velocity value of the respective delay fragment

Examples

```
data(stats_minimal)
data(res_minimal)
dataframe_summary(data = stats_minimal, input = res_minimal)
```

`dataframe_summary_events`

dataframe_summary_events creates one table with all events between the segments. The dataframe_summary_events creates one table with the following columns: event, features, p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, velocity_ratio, FC_HL, FC_intensity, FC_HL/FC_intensity. The columns are:

1. *event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.*
2. *FC_HL: fold change between 2 half-life fragments.*
3. *FC_intensity: fold change between 2 intensity fragments.*
4. *FC_HL/FC_intensity: ratio of fold change between 2 half-life fragments and fold change between 2 intensity fragments.*
5. *velocity_ratio: ratio between any two fragment where the event happen.*
6. *p_value: depending on the event, t-test, manova test p_value is assigned.*
7. *feature_type: indicated on the output data frame as region, are the feature type covering the event.*
8. *gene: gene covering the event.*
9. *locus_tag: locus_tag covering the event.*
10. *strand: +/- indicated in case of stranded data.*
11. *TU: TU covering the event.*
12. *segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I. The rest of the events include HL fragment and intensity fragment.*
13. *segment_2: same description as segment_1 but is the second fragment of the event.*
14. *event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.*
15. *event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.*
16. *gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.*
17. *features: number of segment involved on the event.*

Description

dataframe_summary_events creates one table with all events between the segments. The dataframe_summary_events creates one table with the following columns: event, features, p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, velocity_ratio, FC_HL, FC_intensity, FC_HL/FC_intensity. The columns are:

1. event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.
2. FC_HL: fold change between 2 half-life fragments.
3. FC_intensity: fold change between 2 intensity fragments.
4. FC_HL/FC_intensity: ratio of fold change between 2 half-life fragments and fold change between 2 intensity fragments.
5. velocity_ratio: ratio between any two fragment where the event happen.
6. p_value: depending on the event, t-test, manova test p_value is assigned.
7. feature_type: indicated on the output data frame as region, are the feature type covering the event.
8. gene: gene covering the event.
9. locus_tag: locus_tag covering the event.
10. strand: +/- indicated in case of stranded data.
11. TU: TU covering the event.
12. segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I. The rest of the events include HL fragment and intensity fragment.
13. segment_2: same description as segment_1 but is the second fragment of the event.
14. event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.
15. event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.
16. gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.
17. features: number of segment involved on the event.

Usage

```
dataframe_summary_events(data, data_annotation)
```

Arguments

`data`	SummarizedExperiment: the input data frame with correct format.
`data_annotation`	dataframe: dataframe from processed gff3 file.

Value

event:

FC_HL:

FC_intensity:

FC_HL_FC_intensity:

p_adjusted:

velocity_ratio:

p_value:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_position:

event_duration:

gap_fragments:

features:

Examples

```
if(!require(SummarizedExperiment)){
suppressPackageStartupMessages(library(SummarizedExperiment))
}
data(stats_minimal)
dataframe_summary_events(data = stats_minimal,
data_annotation = metadata(stats_minimal)$annot[[1]])
```

dataframe_summary_events_HL_int

dataframe_summary_events_HL_int creates one table with all events between the segments. The dataframe_summary_events_HL_int creates one table with the following columns: event, features, p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, FC_HL, FC_intensity, FC_HL/FC_intensity. The columns are:

1. *event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.*
2. *FC_HL: fold change between 2 half-life fragments*
3. *FC_intensity: fold change between 2 intensity fragments*
4. *FC_HL/FC_intensity: ratio of fold change between 2 half-life fragments and fold change between 2 intensity fragments.*
5. *p_value: depending on the event, t-test, manova test p_value is assigned.*
6. *feature_type: indicated on the output data frame as region, are the feature type covering the event.*
7. *gene: gene covering the event.*
8. *locus_tag: locus_tag covering the event.*
9. *strand: +/- indicated in case of stranded data.*
10. *TU: TU covering the event.*
11. *segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I. The rest of the events include HL fragment and could be extended intensity fragment.*
12. *segment_2: same description as segment_1 but is the second fragment of the event.*
13. *event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.*
14. *event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.*
15. *gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.*
16. *features: number of segment involved on the event.*

Description

dataframe_summary_events_HL_int creates one table with all events between the segments. The dataframe_summary_events_HL_int creates one table with the following columns: event, features,

p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, FC_HL, FC_intensity, FC_HL/FC_intensity. The columns are:

1. event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.
2. FC_HL: fold change between 2 half-life fragments
3. FC_intensity: fold change between 2 intensity fragments
4. FC_HL/FC_intensity: ratio of fold change between 2 half-life fragments and fold change between 2 intensity fragments.
5. p_value: depending on the event, t-test, manova test p_value is assigned.
6. feature_type: indicated on the output data frame as region, are the feature type covering the event.
7. gene: gene covering the event.
8. locus_tag: locus_tag covering the event.
9. strand: +/- indicated in case of stranded data.
10. TU: TU covering the event.
11. segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I. The rest of the events include HL fragment and could be extended intensity fragment.
12. segment_2: same description as segment_1 but is the second fragment of the event.
13. event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.
14. event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.
15. gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.
16. features: number of segment involved on the event.

Usage

```
dataframe_summary_events_HL_int(data, data_annotation)
```

Arguments

`data`	SummarizedExperiment: the input data frame with correct format.
`data_annotation`	dataframe: dataframe from processed gff3 file.

Value

event:

p_value:

p_adjusted:

FC_HL:

FC_intensity:

FC_HL_adapted: Fold change of half-life/ fold change of intensity, position of the half-life fragment is adapted to intensity fragment

FC_HL_FC_intensity: Fold change of half-life/ fold change of intensity

event_position:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_duration:

gap_fragments:

features:

Examples

```
if(!require(SummarizedExperiment)){
suppressPackageStartupMessages(library(SummarizedExperiment))
}
data(stats_minimal)
dataframe_summary_events_HL_int(data = stats_minimal,
data_annotation = metadata(stats_minimal)$annot[[1]])
```

`dataframe_summary_events_ps_itss`

dataframe_summary_events_ps_itss creates one table with all events between the segments. The dataframe_summary_events_ps_itss creates one table with the following columns: event, features, p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, velocity_ratio. The columns are:

1. *event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.*
2. *velocity_ratio: ratio between any two fragment where the event happen.*
3. *p_value: depending on the event, t-test, manova test p_value is assigned.*
4. *feature_type: indicated on the output data frame as region, are the feature type covering the event.*
5. *gene: gene covering the event.*
6. *locus_tag: locus_tag covering the event.*
7. *strand: +/- indicated in case of stranded data.*
8. *TU: TU covering the event.*
9. *segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I.*
10. *segment_2: same description as segment_1 but is the second fragment of the event.*
11. *event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.*
12. *event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.*
13. *gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.*
14. *features: number of segment involved on the event.*

Description

dataframe_summary_events_ps_itss creates one table with all events between the segments. The dataframe_summary_events_ps_itss creates one table with the following columns: event, features, p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, velocity_ratio. The columns are:

1. event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.

2. velocity_ratio: ratio between any two fragment where the event happen.
3. p_value: depending on the event, t-test, manova test p_value is assigned.
4. feature_type: indicated on the output data frame as region, are the feature type covering the event.
5. gene: gene covering the event.
6. locus_tag: locus_tag covering the event.
7. strand: +/- indicated in case of stranded data.
8. TU: TU covering the event.
9. segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I.
10. segment_2: same description as segment_1 but is the second fragment of the event.
11. event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.
12. event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.
13. gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.
14. features: number of segment involved on the event.

Usage

```
dataframe_summary_events_ps_itss(data, data_annotation)
```

Arguments

`data`	SummarizedExperiment: the input data frame with correct format.
`data_annotation`	dataframe: dataframe from processed gff3 file.

Value

event:

p_value:

p_adjusted:

event_position:

velocity_ratio:

FC_HL_adapted:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_duration:

gap_fragments:

features:

Examples

```
data(stats_minimal)
if(!require(SummarizedExperiment)){
suppressPackageStartupMessages(library(SummarizedExperiment))
}
dataframe_summary_events_ps_itss(data = stats_minimal,
data_annotation = metadata(stats_minimal)$annot[[1]])
```

`dataframe_summary_events_velocity`

dataframe_summary_events_velocity creates one table with all events between the segments. The dataframe_summary_events_velocity creates one table with the following columns: event, features, p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, velocity_ratio. The columns are:

1. *event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.*
2. *velocity_ratio: ratio between any two fragment where the event happen.*
3. *p_value: depending on the event, t-test, manova test p_value is assigned.*
4. *feature_type: indicated on the output data frame as region, are the feature type covering the event.*
5. *gene: gene covering the event.*
6. *locus_tag: locus_tag covering the event.*
7. *strand: +/- indicated in case of stranded data.*
8. *TU: TU covering the event.*
9. *segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I. The rest of the events include HL fragment and could be extended intensity fragment.*
10. *segment_2: same description as segment_1 but is the second fragment of the event.*
11. *event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.*
12. *event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.*
13. *gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.*
14. *features: number of segment involved on the event.*

Description

dataframe_summary_events_velocity creates one table with all events between the segments. The dataframe_summary_events_velocity creates one table with the following columns: event, features, p_value, event_position, event_duration, position, region, gene, locus_tag, strand, TU, segment_1, segment_2, length, velocity_ratio. The columns are:

1. event: event type, pausing site, iTSS_I, iTSS_II, Termination, HL_event, Int_event, HL_Int_event and velocity_change.

2. velocity_ratio: ratio between any two fragment where the event happen.
3. p_value: depending on the event, t-test, manova test p_value is assigned.
4. feature_type: indicated on the output data frame as region, are the feature type covering the event.
5. gene: gene covering the event.
6. locus_tag: locus_tag covering the event.
7. strand: +/- indicated in case of stranded data.
8. TU: TU covering the event.
9. segment_1: the first segment of the event, includes the segment, TU, delay fragment in case of ps or iTSS_I. The rest of the events include HL fragment and could be extended intensity fragment.
10. segment_2: same description as segment_1 but is the second fragment of the event.
11. event_position: the position of event, calculated dividing the last position of the first fragment and the first position of the next fragment on 2.
12. event_duration: the difference (min) between 2 delay fragment when ps or iTSS_I happen.
13. gap_fragments: length in position (nt), calculated by the difference between the last position of the first fragment and the first position of the second fragment.
14. features: number of segment involved on the event.

Usage

```
dataframe_summary_events_velocity(data, data_annotation)
```

Arguments

`data`	SummarizedExperiment: the input data frame with correct format.
`data_annotation`	dataframe: dataframe from processed gff3 file.

Value

event:

p_value:

p_adjusted:

event_position:

velocity_ratio:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_duration:

gap_fragments:

features:

Examples

```
if(!require(SummarizedExperiment)){
suppressPackageStartupMessages(library(SummarizedExperiment))
}
data(stats_minimal)
dataframe_summary_events_velocity(data = stats_minimal,
data_annotation = metadata(stats_minimal)$annot[[1]])
```

`dataframe_summary_TI` *dataframe_summary_TI creates one table with all TI fragments, p_value and the coordinates. The dataframe_summary creates one table with the following columns: event, TI_fragment, TI_factor, TI_fragments_TU, p_value, feature_type, gene, locus_tag, strand, TU, features, event_position, position_1 and position_2. The columns are:*

1. *event: event type, transcription interference.*
2. *TI_fragment: Transcription interference fragment.*
3. *TI_factor: Transcription interference factor.*
4. *TI_fragments_TU: Transcription interference fragments included on the TU.*
5. *p_value: TI p_value between two successive fragments is assigned.*
6. *feature_type: indicated on the output data frame as region, are the feature type covering the TI.*
7. *gene: the genes covering the TI.*
8. *locus_tag: the locus_tags covering the TI.*
9. *strand: +/- indicated in case of stranded data.*
10. *TU: TU covering the TI.*
11. *features: number of segment TI involved on a TU.*
12. *event_position : position between two TI fragments.*
13. *position_1 : the first position of TI fragment, if 2 fragments, first position is from the first fragment.*
14. *position_2 : the last position of TI fragment, if 2 fragments, last position is from the second fragment.*

Description

dataframe_summary_TI creates one table with all TI fragments, p_value and the coordinates. The dataframe_summary creates one table with the following columns: event, TI_fragment, TI_factor, TI_fragments_TU, p_value, feature_type, gene, locus_tag, strand, TU, features, event_position, position_1 and position_2. The columns are:

1. event: event type, transcription interference.
2. TI_fragment: Transcription interference fragment.
3. TI_factor: Transcription interference factor.
4. TI_fragments_TU: Transcription interference fragments included on the TU.
5. p_value: TI p_value between two successive fragments is assigned.
6. feature_type: indicated on the output data frame as region, are the feature type covering the TI.
7. gene: the genes covering the TI.
8. locus_tag: the locus_tags covering the TI.
9. strand: +/- indicated in case of stranded data.
10. TU: TU covering the TI.
11. features: number of segment TI involved on a TU.
12. event_position : position between two TI fragments.
13. position_1 : the first position of TI fragment, if 2 fragments, first position is from the first fragment.
14. position_2 : the last position of TI fragment, if 2 fragments, last position is from the second fragment.

Usage

```
dataframe_summary_TI(data, input)
```

Arguments

data	SummarizedExperiment: the input data frame with correct format.
input	dataframe: dataframe from event_dataframe function.

Value

WIP

Examples

```
data(stats_minimal)
data(res_minimal)
dataframe_summary_TI(data = stats_minimal, input = res_minimal)
```

event_dataframe	*event_dataframe: creates a dataframe only with events for segments and genes. The function used are: position_function: adds the specific position of ps or iTSS event annotation_function_event: adds the events to the annotated genes. gff3 file has to be supplied. Strand is indicated in case of stranded data The event_dataframe selects columns with statistical features. ID, position, strand and TU columns are required. Two major dataframe are generated, df gathers t-test and Manova test and df1 gathers ps and ITSS with the corresponding features. df selects only unique intensity fragments since they are the lowest on the hierarchy. One new column is added to df, "synthesis_ratio_event", it corresponds to the FC-HL/FC-intensity and assignment of an event to the synthesis ratio respectively. df adds a new column to indicate the position of the ps or ITSS event.*

Description

event_dataframe: creates a dataframe only with events for segments and genes. The function used are: position_function: adds the specific position of ps or iTSS event annotation_function_event: adds the events to the annotated genes. gff3 file has to be supplied. Strand is indicated in case of stranded data The event_dataframe selects columns with statistical features. ID, position, strand and TU columns are required. Two major dataframe are generated, df gathers t-test and Manova test and df1 gathers ps and ITSS with the corresponding features. df selects only unique intensity fragments since they are the lowest on the hierarchy. One new column is added to df, "synthesis_ratio_event", it corresponds to the FC-HL/FC-intensity and assignment of an event to the synthesis ratio respectively. df adds a new column to indicate the position of the ps or ITSS event.

Usage

```
event_dataframe(data, data_annotation)
```

Arguments

data	dataframe: the probe based data frame.
data_annotation	dataframe: the coordinates are extracted from gff3

Value

WIP

Examples

```
if(!require(SummarizedExperiment)){
suppressPackageStartupMessages(library(SummarizedExperiment))
}
data(stats_minimal)
event_dataframe(data = stats_minimal,
```

```
data_annotation = metadata(stats_minimal)$annot[[1]])
```

`example_input_e_coli` *An example SummarizedExperiment from E. coli An example SummarizedExperiment from RNA-seq containing information about the intensities at all time points (assay). Seqnames, IRanges and strand columns (rowRanges)and colData with time point series and replicates.*

Description

An example SummarizedExperiment from E. coli An example SummarizedExperiment from RNA-seq containing information about the intensities at all time points (assay). Seqnames, IRanges and strand columns (rowRanges)and colData with time point series and replicates.

Usage

```
data(example_input_e_coli)
```

Format

A assay:

0: relative intensities at 0 min

1: relative intensities at 1 min

10: relative intensities at 10 min

15: relative intensities at 15 min

2: relative intensities at 2 min

20: relative intensities at 20 min

3: relative intensities at 3 min

4: relative intensities at 4 min

5: relative intensities at 5 min

6: relative intensities at 6 min

8: relative intensities at 8 min

Source

https://github.com/CyanolabFreiburg/rifi

example_input_minimal *An artificial example SummarizedExperiment An example SummarizedExperiment containing information about the intensities at all time points (assay). Seqnames, IRanges and strand columns (rowRanges) and colData with time point series and replicates.*

Description

An artificial example SummarizedExperiment An example SummarizedExperiment containing information about the intensities at all time points (assay). Seqnames, IRanges and strand columns (rowRanges) and colData with time point series and replicates.

Usage

```
data(example_input_minimal)
```

Format

An object of class `RangedSummarizedExperiment` with 4 rows and 33 columns.

Source

https://github.com/CyanolabFreiburg/rifi

example_input_synechocystis_6803

An example input data frame from Synechocystis PCC 6803 A SummarizedExperiment from microarrays data containing information about the intensities at all time points (assay), Seqnames, IRanges and strand columns (rowRanges) and colData with time point series and averaged replicates.

Description

An example input data frame from Synechocystis PCC 6803 A SummarizedExperiment from microarrays data containing information about the intensities at all time points (assay), Seqnames, IRanges and strand columns (rowRanges) and colData with time point series and averaged replicates.

Usage

```
data(example_input_synechocystis_6803)
```

Format

Assay with 3000 rows and 10 variables:

0: relative intensities at 0 min

2: relative intensities at 2 min

4: relative intensities at 4 min

8: relative intensities at 8 min

16: relative intensities at 16 min

32: relative intensities at 32 min

64: relative intensities at 64 min

Source

https://github.com/CyanolabFreiburg/rifi

`finding_PDD` *finding_PDD: flags potential candidates for post transcription decay. 'finding_PDD' uses 'score_fun_linear_PDD' to make groups by the difference to the slope. Then the slope is checked for steepness to decide for PDD. '*PDD*' is added to the 'flag' column. Post transcription decay is characterized by a strong decrease of intensity by position. The rowRanges need to contain at least 'ID', 'intensity', 'position' and 'position_segment'!*

Description

finding_PDD: flags potential candidates for post transcription decay. 'finding_PDD' uses 'score_fun_linear_PDD' to make groups by the difference to the slope. Then the slope is checked for steepness to decide for PDD. '*PDD*' is added to the 'flag' column. Post transcription decay is characterized by a strong decrease of intensity by position. The rowRanges need to contain at least 'ID', 'intensity', 'position' and 'position_segment'!

Usage

```
finding_PDD(inp, cores = 1, pen = 2, pen_out = 1, thrsh = 0.001)
```

Arguments

`inp`	SummarizedExperiment: the input.
`cores`	integer: the number of assigned cores for the task
`pen`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Advised to be kept at 2. Default is 2.
`pen_out`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer possible outliers. Advised to be kept at 1. Default is 1.
`thrsh`	numeric: an internal parameter that allows fragments with slopes steeper than the thrsh to be flagged with '*PDD*'. Higher values result in fewer candidates. Advised to be kept at 0.001. Default is 0.001.

Value

the SummarizedExperiment object: with "*PDD*" added to the flag column.

Examples

```
data(preprocess_minimal)
finding_PDD(inp = preprocess_minimal, cores = 2, pen = 2,
pen_out = 1, thrsh = 0.001)
```

`finding_TI` *finding_TI: flags potential candidates for transcription interference. 'finding_TI' uses 'score_fun_ave' to make groups by the mean of "probe_TI". "TI" is added to the "flag" column. TI is characterized by relative intensities at time points later than "0". The rowRanges need to contain at least "ID", "probe_TI" and "position_segment"!*

Description

finding_TI: flags potential candidates for transcription interference. 'finding_TI' uses 'score_fun_ave' to make groups by the mean of "probe_TI". "TI" is added to the "flag" column. TI is characterized by relative intensities at time points later than "0". The rowRanges need to contain at least "ID", "probe_TI" and "position_segment"!

Usage

```
finding_TI(inp, cores, pen = 10, thrsh = 0.5, add = 1000)
```

Arguments

`inp`	SummarizedExperiment: the input.
`cores`	integer: the number of assigned cores for the task
`pen`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Advised to be kept at 10. Default is 10.
`thrsh`	numeric: an internal parameter that allows fragments with a certain amount of IDs with higher relative intensities at time points later than "0" to be flagged as "*TI*". Higher values result in fewer candidates. -0.5 is 25 %, 0 is 50%, 0.5 is 75%. Advised to be kept at 0.5. Default is 0.5.
`add`	integer: range of nucleotides before and after a potential TI event wherein IDs are fitted with the TI fit.

Value

the SummarizedExperiment object: with "*TI*" added to the flag column.

Examples

```
data(preprocess_minimal)
finding_TI(inp = preprocess_minimal, cores = 2, pen = 10, thrsh = 0.5,
add = 1000)
```

`fit_e_coli` *The result of rifi_fit for E.coli example data A SummarizedExperiment containing the output from rifi_fit as an extension of rowRanges and metadata.*

Description

The result of rifi_fit for E.coli example data A SummarizedExperiment containing the output from rifi_fit as an extension of rowRanges and metadata.

Usage

```
data(fit_e_coli)
```

Format

Three data frames with 290 rows and 10 variables, 155 rows and 5 variables, and 135 rows and 9 variables are generated. The columns of the first data frame are added to the rowRanges and the rest are added as metadata.

inp: The SummarizedExperiment:

- **ID:** The bin/probe specific ID
- **position:** The bin/probe specific position
- **intensity:** The relative intensity at time point 0
- **probe_TI:** An internal value to determine which fitting model is applied
- **flag:** Information on which fitting model is applied
- **postion_segment:** The position based segment
- **delay:** The delay value of the bin/probe
- **half_life:** The half-life of the bin/probe
- **TI_termination_factor:** The termination factor of the bin/probe

fit_obj_STD: the fit object for the standard fit:

- **ID:** The bin/probe specific ID
- **delay:** The delay value of the bin/probe
- **half_life:** The half-life of the bin/probe
- **inty_S0:** The relative intensity at time point 0
- **intyf:** The background value of the fit

fit_obj_TI: the fit object for the TI fit:

- **delay:** The delay value of the bin/probe

ti_delay: The ti-delay value of the bin/probe

half_life: The half-life of the bin/probe

ti_value: The ti-value of the bin/probe

TI_termination_factor: The termination factor of the bin/probe

synthesis_rate: The synthesis rate of the bin/probe

TI_background: The background value of the fit

position: The bin/probe specific position

ID: The bin/probe specific ID

Source

https://github.com/CyanolabFreiburg/rifi

`fit_minimal` *The artificial result of rifi_fit for artificial example data A SummarizedExperiment containing the output from rifi_fit.*

Description

The artificial result of rifi_fit for artificial example data A SummarizedExperiment containing the output from rifi_fit.

Usage

```
data(fit_minimal)
```

Format

An object of class `RangedSummarizedExperiment` with 4 rows and 33 columns.

Source

https://github.com/CyanolabFreiburg/rifi

`fit_synechocystis_6803` *The result of rifi_fit for Synechocystis 6803 example data A SummarizedExperiment containing the output from rifi_fit as an extension of rowRanges and metadata.*

Description

The result of rifi_fit for Synechocystis 6803 example data A SummarizedExperiment containing the output from rifi_fit as an extension of rowRanges and metadata.

Usage

```
data(fit_synechocystis_6803)
```

Format

Three data frames with 3000 rows and 10 variables, 2811 rows and 5 variables, and 189 rows and 9 variable are generated. The columns of the first data frame are added to the rowRanges and the rest are added as metadata.

inp: the SummarizedExperiment:

- **ID:** The bin/probe specific ID
- **position:** The bin/probe specific position
- **strand:** The bin/probe specific strand
- **intensity:** The relative intensity at time point 0
- **probe_TI:** An internal value to determine which fitting model is applied
- **flag:** Information on which fitting model is applied
- **postion_segment:** The position based segment
- **delay:** The delay value of the bin/probe
- **half_life:** The half-life of the bin/probe
- **TI_termination_factor:** The termination factor of the bin/probe

fit_obj_STD: the fit object for the standard fit:

- **ID:** The bin/probe specific ID
- **delay:** The delay value of the bin/probe
- **half_life:** The half-life of the bin/probe
- **inty_S0:** The relative intensity at time point 0
- **intyf:** The background value of the fit

fit_obj_TI: the fit object for the TI fit:

- **delay:** The delay value of the bin/probe
- **ti_delay:** The ti-delay value of the bin/probe
- **half_life:** The half-life of the bin/probe
- **ti_value:** The ti-value of the bin/probe

TI_termination_factor: The termination factor of the bin/probe
synthesis_rate: The synthesis rate of the bin/probe
TI_background: The background value of the fit
position: The bin/probe specific position
ID: The bin/probe specific ID

Source

https://github.com/CyanolabFreiburg/rifi

`fold_change` *fold_change: it sets a fold-change ratio between the neighboring fragments of Half-life (HL) and intensity. fold_change sets fold change on intensity and fold change HL fragments of two successive fragments. Two intensity fragments could belong to one HL fragment. This function sets first the borders using the position and applies the fold change ratio between the neighboring fragments of HL and those from intensity (intensity frgA/intensity frgB/half-life frgA/half-life frgB). All grepped fragments are from the same TU excluding outliers.*

Description

The function used is: synthesis_r_Function: assigns events depending on the ratio between HL and intensity of two consecutive fragments. intensity(int) = synthesis rate(k)/decay(deg) (steady state), int1/int2 = k1/deg1*deg2/k2 int1 * (deg1/int2) * deg2 = k1/k2 => synthesis ratio. In case of synthesis ratio is: synthesis ratio > 1 -> New start synthesis ratio < 1 -> Termination

Usage

```
fold_change(inp)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.

Value

the SummarizedExperiment with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

half_life: The half-life of the bin/probe

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

event_duration:

event_position:

FC_HL:

FC_fragment_HL:

p_value_HL:

FC_intensity:

FC_fragment_intensity:

p_value_intensity:

FC_HL_intensity:

FC_HL_intensity_fragment:

FC_HL_adapted:

Examples

```
data(stats_minimal)
fold_change(inp = stats_minimal)
```

---

fragmentation_e_coli *The result of rifi_fragmentation for E.coli example data A SummarizedExperiment containing the output from rifi_fragmentation as an extension of rowRanges*

Description

The result of rifi_fragmentation for E.coli example data A SummarizedExperiment containing the output from rifi_fragmentation as an extension of rowRanges

Usage

```
data(fragmentation_e_coli)
```

Format

rowRanges of the SummarizedExperiment with 290 rows and 22 variables:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

probe_TI: An internal value to determine which fitting model is applied

flag: Information on which fitting model is applied

position_segment: The position based segment

delay: The delay value of the bin/probe

half_life: The half-life of the bin/probe

TI_termination_factor: The termination factor of the bin/probe

delay_fragment: The delay fragment the bin belongs to

velocity_fragment: The velocity value of the respective delay fragment

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

TI_termination_fragment: The TI fragment the bin belongs to

TI_mean_termination_factor: The mean termination factor of the respective TI fragment

seg_ID: The combined ID of the fragment

Source

https://github.com/CyanolabFreiburg/rifi

`fragmentation_minimal` *The result of rifi_fragmentation for artificial example data A SummarizedExperiment containing the output from rifi_fragmentation as an extension of rowRanges and metadata.*

Description

The result of rifi_fragmentation for artificial example data A SummarizedExperiment containing the output from rifi_fragmentation as an extension of rowRanges and metadata.

Usage

```
data(fragmentation_minimal)
```

Format

An object of class `RangedSummarizedExperiment` with 24 rows and 33 columns.

Source

https://github.com/CyanolabFreiburg/rifi

`fragmentation_synechocystis_6803`
The result of rifi_fragmentation for Synechocystis 6803 example data A SummarizedExperiment containing the output from rifi_fragmentation as an extension fo rowRanges

---

Description

The result of rifi_fragmentation for Synechocystis 6803 example data A SummarizedExperiment containing the output from rifi_fragmentation as an extension fo rowRanges

Usage

```
data(fragmentation_synechocystis_6803)
```

Format

rowRanges of the SummarizedExperiment:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

probe_TI: An internal value to determine which fitting model is applied

flag: Information on which fitting model is applied

position_segment: The position based segment

delay: The delay value of the bin/probe

half_life: The half-life of the bin/probe

TI_termination_factor: The termination factor of the bin/probe

delay_fragment: The delay fragment the bin belongs to

velocity_fragment: The velocity value of the respective delay fragment

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

TI_termination_fragment: The TI fragment the bin belongs to

TI_mean_termination_factor: The mean termination factor of the respective TI fragment

seg_ID: The combined ID of the fragment

Source

https://github.com/CyanolabFreiburg/rifi

`fragment_delay` *fragment_delay: performs the delay fragmentation. fragment_delay makes delay_fragments based on position_segments and assigns all gathered information to the SummarizedExperiment object. The columns "delay_fragment", "velocity_fragment", "intercept" and "slope" are added. fragment_delay makes delay_fragments, assigns slopes, which are 1/velocity at the same time, and intercepts for the TU calculation. The function used is: score_fun_linear the input is the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.*

Description

fragment_delay: performs the delay fragmentation. fragment_delay makes delay_fragments based on position_segments and assigns all gathered information to the SummarizedExperiment object. The columns "delay_fragment", "velocity_fragment", "intercept" and "slope" are added. fragment_delay makes delay_fragments, assigns slopes, which are 1/velocity at the same time, and intercepts for the TU calculation. The function used is: score_fun_linear the input is the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.

Usage

```
fragment_delay(inp, cores = 1, pen, pen_out)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.
`cores`	cores: integer: the number of assigned cores for the task.
`pen`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
`pen_out`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.

Value

the SummarizedExperiment object: with delay_fragment, velocity_fragment, intercept and slope added to the rowRanges.

Examples

```
data(fragmentation_minimal)
fragment_delay(inp = fragmentation_minimal, cores = 2, pen = 2, pen_out = 1)
```

`fragment_HL` *fragment_HL: performs the half_life fragmentation fragment_HL makes HL_fragments based on delay_fragments and assigns all gathered information to the SummarizedExperiment object. The columns "HL_fragment" and "HL_mean_fragment" are added. fragment_HL makes half-life_fragments and assigns the mean of each fragment. The function used is: .score_fun_ave. The input the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.*

Description

fragment_HL: performs the half_life fragmentation fragment_HL makes HL_fragments based on delay_fragments and assigns all gathered information to the SummarizedExperiment object. The columns "HL_fragment" and "HL_mean_fragment" are added. fragment_HL makes half-life_fragments and assigns the mean of each fragment. The function used is: .score_fun_ave. The input the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.

Usage

```
fragment_HL(inp, cores = 1, pen, pen_out)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.
`cores`	integer: the number of assigned cores for the task.
`pen`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
`pen_out`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.

Value

the SummarizedExperiment object: with HL_fragment and HL_mean_fragment added to the rowRanges.

Examples

```
data(fragmentation_minimal)
fragment_HL(inp = fragmentation_minimal, cores = 2, pen = 2, pen_out = 1)
```

`fragment_inty` *fragment_inty: performs the intensity fragmentation fragment_inty makes intensity_fragments based on HL_fragments and assigns all gathered information to the SummarizedExperiment object. The columns "intensity_fragment" and "intensity_mean_fragment" are added. fragment_inty makes intensity_fragments and assigns the mean of each fragment. The function used is: .score_fun_ave. The input is the the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.*

Description

fragment_inty: performs the intensity fragmentation fragment_inty makes intensity_fragments based on HL_fragments and assigns all gathered information to the SummarizedExperiment object. The columns "intensity_fragment" and "intensity_mean_fragment" are added. fragment_inty makes intensity_fragments and assigns the mean of each fragment. The function used is: .score_fun_ave. The input is the the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.

Usage

```
fragment_inty(inp, cores = 1, pen, pen_out)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.
`cores`	cores: integer: the number of assigned cores for the task.
`pen`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
`pen_out`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.

Value

the SummarizedExperiment object: with intensity_fragment and intensity_mean_fragment added to the rowRanges.

Examples

```
data(fragmentation_minimal)
fragment_inty(inp = fragmentation_minimal, cores = 2, pen = 2, pen_out = 1)
```

fragment_TI	*fragment_TI: performs the TI fragmentation. fragment_TI makes TI_fragments based on TUs and assigns all gathered information to the SummarizedExperiment object. The columns "TI_termination_fragment" and the TI_mean_termination_factor are added. The function used is: .score_fun_ave. The input is the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.*

Description

fragment_TI: performs the TI fragmentation. fragment_TI makes TI_fragments based on TUs and assigns all gathered information to the SummarizedExperiment object. The columns "TI_termination_fragment" and the TI_mean_termination_factor are added. The function used is: .score_fun_ave. The input is the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming, pen_out is the outlier penalty.

Usage

```
fragment_TI(inp, cores = 1, pen, pen_out)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
cores	cores: integer: the number of assigned cores for the task.
pen	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
pen_out	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.

Value

the SummarizedExperiment object: with TI_termination_fragment and TI_termination_mean_fragment added to the rowRanges.

Examples

```
data(fragmentation_minimal)
fragment_TI(inp = fragmentation_minimal, cores = 2, pen = 2, pen_out = 1)
```

gff3_preprocess	*gff3_preprocess: process gff3 file from database for multiple usage. gff3_preprocess processes the gff3 file extracting gene names and locus_tag from all coding regions (CDS), UTRs/ncRNA/asRNA are also extracted if available. The resulting dataframe contains region, positions, strand, gene and locus_tag.*

Description

gff3_preprocess: process gff3 file from database for multiple usage. gff3_preprocess processes the gff3 file extracting gene names and locus_tag from all coding regions (CDS), UTRs/ncRNA/asRNA are also extracted if available. The resulting dataframe contains region, positions, strand, gene and locus_tag.

Usage

```
gff3_preprocess(path)
```

Arguments

path	path: path to the directory containing the gff3 file.

Value

A list with 2 items:

data annotation: **region:** the region from the gff file

start: the start of the annotation

end: the end of the annotation

strand: the strand of the annotation

gene: the annotated gene name

locus_tag: the annotated locus tag

genome length: a numeric vector containing the length of the genome

Examples

```
gff3_preprocess(
path = gzfile(system.file("extdata", "gff_e_coli.gff3.gz", package = "rifi"))
)
```

make_df	*make_df: adds important columns to the SummarizedExperiment object. 'make_df' adds to the SummarizedExperiment object with the columns: "intensity", "probe_TI" and "flag". The replicates are collapsed into their respective means. "intensity" is the mean intensity from time point 0. "probe_TI" is a value needed for the distribution for the different fitting models. "flag" contains information or the distribution for the different fitting models. Here, probes that don't reach the background level expression are flagged as "*ABG*" ("above background"). This is only needed for microarray data and is controlled by the bg parameter. The default for bg = 0, resulting in all probes to be above background (0 is advised for RNAseq data). Probes where all replicates were filtered in the optional filtration step can be fully removed by rm_FLT = TRUE! If you wish to keep all information in the assay set to FALSE!*

Description

make_df: adds important columns to the SummarizedExperiment object. 'make_df' adds to the SummarizedExperiment object with the columns: "intensity", "probe_TI" and "flag". The replicates are collapsed into their respective means. "intensity" is the mean intensity from time point 0. "probe_TI" is a value needed for the distribution for the different fitting models. "flag" contains information or the distribution for the different fitting models. Here, probes that don't reach the background level expression are flagged as "*ABG*" ("above background"). This is only needed for microarray data and is controlled by the bg parameter. The default for bg = 0, resulting in all probes to be above background (0 is advised for RNAseq data). Probes where all replicates were filtered in the optional filtration step can be fully removed by rm_FLT = TRUE! If you wish to keep all information in the assay set to FALSE!

Usage

```
make_df(inp, cores = 1, bg = 0, rm_FLT = TRUE)
```

Arguments

inp	SummarizedExperiment: the (checked) input.
cores	integer: the number of assigned cores for the task.
bg	numeric: threshold over which the last timepoint has to be fitted with the above background mode.
rm_FLT	logical: remove IDs where all replicates are marked as filtered. Default is FALSE.

Value

the SummarizedExperiment object: with intensity, probe_TI and flag added to the rowRanges.

Examples

```
data(preprocess_minimal)
make_df(inp = preprocess_minimal, cores = 2, bg = 0, rm_FLT = TRUE)
```

---

make_pen

make_pen: automatically assigns a penalties. 'make_pen' calls one of four available penalty functions to automatically assign penalties for the dynamic programming. The four functions to be called are:

1. *fragment_delay_pen*
2. *fragment_HL_pen*
3. *fragment_inty_pen*
4. *fragment_TI_pen. These functions return the amount of statistically correct and statistically wrong splits at a specific pair of penalties. 'make_pen' iterates over many penalty pairs and picks the most suitable pair based on the difference between wrong and correct splits. The sample size, penalty range and resolution as well as the number of cycles can be customized. The primary start parameters create a matrix with n = rez_pen rows and n = rez_pen_out columns with values between sta_pen/sta_pen_out and end_pen/end_pen_out. The best penalty pair is picked. If dept is bigger than 1 the same process is repeated with a new matrix of the same size based on the result of the previous cycle. Only position segments with length within the sample size range are considered for the penalties to increase run time. Returns a penalty object (list of 4 objects) the first being the logbook.*

Description

make_pen: automatically assigns a penalties. 'make_pen' calls one of four available penalty functions to automatically assign penalties for the dynamic programming. The four functions to be called are:

1. fragment_delay_pen
2. fragment_HL_pen
3. fragment_inty_pen
4. fragment_TI_pen. These functions return the amount of statistically correct and statistically wrong splits at a specific pair of penalties. 'make_pen' iterates over many penalty pairs and picks the most suitable pair based on the difference between wrong and correct splits. The sample size, penalty range and resolution as well as the number of cycles can be customized. The primary start parameters create a matrix with n = rez_pen rows and n = rez_pen_out columns with values between sta_pen/sta_pen_out and end_pen/end_pen_out. The best penalty pair is picked. If dept is bigger than 1 the same process is repeated with a new matrix of the same size based on the result of the previous cycle. Only position segments with length within the sample size range are considered for the penalties to increase run time. Returns a penalty object (list of 4 objects) the first being the logbook.

Usage

```
make_pen(
  inp,
  FUN,
  cores = 1,
  logs,
  dpt = 1,
  smpl_min = 10,
  smpl_max = 100,
  sta_pen = 0.5,
  end_pen = 4.5,
  rez_pen = 9,
  sta_pen_out = 0.5,
  end_pen_out = 3.5,
  rez_pen_out = 7
)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
FUN	function: one of the four bottom level functions (see details)
cores	integer: the number of assigned cores for the task
logs	numeric vector: the logbook vector.
dpt	integer: the number of times a full iteration cycle is repeated with a more narrow range based on the previous cycle. Default is 2.
smpl_min	integer: the smaller end of the sampling size. Default is 10.
smpl_max	integer: the larger end of the sampling size. Default is 100.
sta_pen	numeric: the lower starting penalty. Default is 0.5.
end_pen	numeric: the higher starting penalty. Default is 4.5.
rez_pen	numeric: the number of penalties iterated within the penalty range. Default is 9.
sta_pen_out	numeric: the lower starting outlier penalty. Default is 0.5.
end_pen_out	numeric: the higher starting outlier penalty. Default is 3.5.
rez_pen_out	numeric: the number of outlier penalties iterated within the outlier penalty range. Default is 7.

Value

A list with 4 items:

logbook: The logbook vector containing all penalty information

penalties: a vetor with the respective penalty and outlier penalty

correct: a matrix of the correct splits

wrong: a matrix of the incorrect splits

Examples

```
data(fit_minimal)
make_pen(
  inp = fit_minimal, FUN = rifi:::fragment_HL_pen, cores = 2,
  logs = as.numeric(rep(NA, 8)), dpt = 1, smpl_min = 10, smpl_max = 50,
  sta_pen = 0.5, end_pen = 4.5, rez_pen = 9, sta_pen_out = 0.5,
  end_pen_out = 3.5, rez_pen_out = 7
)
```

---

nls2_fit

nls2_fit: estimates decay for each probe or bin. nls2_fit uses nls2 function to fit a probe or bin using intensities of the time series data from different time point. nls2 uses different starting values through expand grid and selects the best fit. Different filters could be applied prior fitting to the model. To apply nls2_fit function, prior filtration could applied.

1. *generic_filter_BG: filter probes with intensities below background using threshold. Those probes are filtered.*
2. *filtration_below_backg: additional functions exclusive to microarrays could be applied. Its very strict to the background (not recommended in usual case).*
3. *filtration_above_backg: selects probes with a very high intensity and above the background (recommended for special transcripts). Probes are flagged with "*ABG*". Those transcripts are usually related to a specific function in bacteria. This filter selects all probes with the same ID, the mean is applied, the last time point is selected and compared to the threshold.*

Description

the model used estimates the delay, decay, intensity of the first time point (synthesis rate/decay) and the background. The coefficients are gathered in vectors with the corresponding IDs. Absence of the fit or a very bad fit are assigned with NA. In case of probes with very high intensities and above the background, the model used makes abstinence of background coefficient. The output of all coefficients is saved in the metadata. The fits are plotted using the function_plot_fit.r through rifi_fit.

Usage

```
nls2_fit(
  inp,
  cores = 1,
  decay = seq(0.08, 0.11, 0.02),
  delay = seq(0, 10, 0.1),
  k = seq(0.1, 1, 0.2),
```

```
  bg = 0.2
)
```

Arguments

inp	SummarizedExperiment: the input with correct format.
cores	integer: the number of assigned cores for the task.
decay	numeric vector: A sequence of starting values for the decay. Default is seq(.08, 0.11, by=.02)
delay	numeric vector: A sequence of starting values for the delay. Default is seq(0,10, by=.1)
k	numeric vector: A sequence of starting values for the synthesis rate. Default is seq(0.1,1,0.2)
bg	numeric vector: A sequence of starting values. Default is 0.2.

Value

the SummarizedExperiment object: with delay and decay added to the rowRanges. The full fit data is saved in the metadata as "fit_STD".

Examples

```
data(preprocess_minimal)
nls2_fit(inp = preprocess_minimal, cores = 2)
```

---

penalties_e_coli

The result of rifi_penalties for E.coli example data. A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata.

Description

The result of rifi_penalties for E.coli example data. A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata.

Usage

```
data(penalties_e_coli)
```

Format

A list with 5 items:

logbook: The logbook vector containing all penalty information

pen_obj_delay: A list with 4 items:

- **logbook:** The logbook vector containing all penalty information
- **delay_penalties:** a vetor with the delay penalty and delay outlier penalty
- **correct:** a matrix of the correct splits
- **wrong:** a matrix of the incorrect splits

pen_obj_HL: A list with 4 items:

- **logbook:** The logbook vector containing all penalty information
- **HL_penalties:** a vetor with the half-life penalty and half-life outlier penalty
- **correct:** a matrix of the correct splits
- **wrong:** a matrix of the incorrect splits

pen_obj_inty: A list with 4 items:

- **logbook:** The logbook vector containing all penalty information
- **inty_penalties:** a vetor with the intensity penalty and intensity outlier penalty
- **correct:** a matrix of the correct splits
- **wrong:** a matrix of the incorrect splits

pen_obj_TI: A list with 4 items:

- **logbook:** The logbook vector containing all penalty information
- **TI_penalties:** a vetor with the TI penalty and TI outlier penalty
- **correct:** a matrix of the correct splits
- **wrong:** a matrix of the incorrect splits

Source

https://github.com/CyanolabFreiburg/rifi

---

penalties_minimal

The result of rifi_penalties for artificial example data A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata.

Description

The result of rifi_penalties for artificial example data A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata.

Usage

```
data(penalties_minimal)
```

Format

An object of class `RangedSummarizedExperiment` with 24 rows and 33 columns.

Source

https://github.com/CyanolabFreiburg/rifi

`penalties_synechocystis_6803`

The result of rifi_penalties for Synechocystis 6803 example data. A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata.

Description

The result of rifi_penalties for Synechocystis 6803 example data. A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata.

Usage

```
data(penalties_synechocystis_6803)
```

Format

A list with 5 items:

logbook: The logbook vector containing all penalty information

pen_obj_delay: A list with 4 items:

- **logbook:** The logbook vector containing all penalty information
- **delay_penalties:** a vetor with the delay penalty and delay outlier penalty
- **correct:** a matrix of the correct splits
- **wrong:** a matrix of the incorrect splits

pen_obj_HL: A list with 4 items:

- **logbook:** The logbook vector containing all penalty information
- **HL_penalties:** a vetor with the half-life penalty and half-life outlier penalty
- **correct:** a matrix of the correct splits
- **wrong:** a matrix of the incorrect splits

pen_obj_inty: A list with 4 items:

- **logbook:** The logbook vector containing all penalty information
- **inty_penalties:** a vetor with the intensity penalty and intensity outlier penalty
- **correct:** a matrix of the correct splits
- **wrong:** a matrix of the incorrect splits

pen_obj_TI: A list with 4 items:

logbook: The logbook vector containing all penalty information
TI_penalties: a vetor with the TI penalty and TI outlier penalty
correct: a matrix of the correct splits
wrong: a matrix of the incorrect splits

Source

https://github.com/CyanolabFreiburg/rifi

`predict_ps_itss` *predict_ps_itss: predicts pausing sites (ps) and internal starting sites (ITSS) between delay fragments. predict_ps_itss predicts ps and ITSS within the same TU. Neighboring delay segments are compared to each other by positioning the intercept of the second segment into the first segment using slope and intercept coefficients.#' predict_ps_itss uses 3 steps to identify ps and ITSS:*

1. *select unique TU.*
2. *select from the input dataframe the columns: ID, position, strand, delay, delay fragment, TU and slope coordinates, velocity_fragment and intercept.*
3. *select delay segments in the TU.*
4. *loop into all delay segments and estimate the coordinates of the last point of the first segment using the coefficients of the second segment and vice versa. We get two predicted positions, the difference between them is compared to the threshold. In case the strand is "-", additional steps are added: The positions of both segments are ordered from the last position to the first one. all positions are merged in one column and subtracted from the maximum position. the column is split in 2. The first and second correspond to the positions of the first and second segments respectively. Both segments are subjected to lm fit and the positions predicted are used on the same way as the opposite strand. if the difference between the positions predicted is lower than negative threshold, ps is assigned otherwise, and if the difference is higher than the positive threshold, ITSS is assigned.*

Description

predict_ps_itss: predicts pausing sites (ps) and internal starting sites (ITSS) between delay fragments. predict_ps_itss predicts ps and ITSS within the same TU. Neighboring delay segments are compared to each other by positioning the intercept of the second segment into the first segment using slope and intercept coefficients.#' predict_ps_itss uses 3 steps to identify ps and ITSS:

1. select unique TU.

2. select from the input dataframe the columns: ID, position, strand, delay, delay fragment, TU and slope coordinates, velocity_fragment and intercept.
3. select delay segments in the TU.
4. loop into all delay segments and estimate the coordinates of the last point of the first segment using the coefficients of the second segment and vice versa. We get two predicted positions, the difference between them is compared to the threshold. In case the strand is "-", additional steps are added: The positions of both segments are ordered from the last position to the first one. all positions are merged in one column and subtracted from the maximum position. the column is split in 2. The first and second correspond to the positions of the first and second segments respectively. Both segments are subjected to lm fit and the positions predicted are used on the same way as the opposite strand. if the difference between the positions predicted is lower than negative threshold, ps is assigned otherwise, and if the difference is higher than the positive threshold, ITSS is assigned.

Usage

```
predict_ps_itss(inp, maxDis = 300)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
maxDis	integer: the maximal distance allowed between two successive fragments.

Value

the SummarizedExperiment with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

delay: The delay value of the bin/probe

delay_fragment: The delay fragment the bin belongs to

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

TU: The overarching transcription unit

pausing_site:

iTSS_I:

ps_ts_fragment:

event_duration:

Examples

```
data(fragmentation_minimal)
predict_ps_itss(inp = fragmentation_minimal, maxDis = 300)
```

preprocess_e_coli	*The result of rifi_preprocess for E.coli example data A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata. A list containing the output from rifi_preprocess, including the inp and the modified input_df.*

Description

The result of rifi_preprocess for E.coli example data A SummarizedExperiment containing the output from rifi_penalties including the logbook and the four penalty objects as metadata. A list containing the output from rifi_preprocess, including the inp and the modified input_df.

Usage

```
data(preprocess_e_coli)
```

Format

A SummarizedExperiment:

inp: the SummarizedExperiment:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

probe_TI: An internal value to determine which fitting model is applied

flag: Information on which fitting model is applied

postion_segment: The position based segment

fit_obj_TI: the fit object for the TI fit:

0: relative intensities at 0 min

1: relative intensities at 1 min

10: relative intensities at 10 min

15: relative intensities at 15 min

2: relative intensities at 2 min

20: relative intensities at 20 min

3: relative intensities at 3 min

4: relative intensities at 4 min

5: relative intensities at 5 min

6: relative intensities at 6 min

8: relative intensities at 8 min

ID: unique IDs

position: genome positions

filtration: indicator wether the replicate is filtered or not

Source

https://github.com/CyanolabFreiburg/rifi

`preprocess_minimal` *The result of rifi_preprocess for artificial example data A SummarizedExperiment containing the output from rifi_preprocess*

Description

The result of rifi_preprocess for artificial example data A SummarizedExperiment containing the output from rifi_preprocess

Usage

```
data(preprocess_minimal)
```

Format

An object of class `RangedSummarizedExperiment` with 4 rows and 33 columns.

Source

https://github.com/CyanolabFreiburg/rifi

`preprocess_synechocystis_6803`
The result of rifi_preprocess for Synechocystis 6803 example data is a A SummarizedExperiment containing the output of rifi_preprocess as an extention to rowRanges

---

Description

The result of rifi_preprocess for Synechocystis 6803 example data is a A SummarizedExperiment containing the output of rifi_preprocess as an extention to rowRanges

Usage

```
data(preprocess_synechocystis_6803)
```

Format

A SummarizedExperiment:

inp: the SummarizedExperiment:

ID: The bin/probe specific ID
position: The bin/probe specific position
strand: The bin/probe specific strand
intensity: The relative intensity at time point 0
probe_TI: An internal value to determine which fitting model is applied
flag: Information on which fitting model is applied
postion_segment: The position based segment

fit_obj_TI: the fit object for the TI fit:

0: relative intensities at 0 min
2: relative intensities at 2 min
4: relative intensities at 4 min
8: relative intensities at 8 min
16: relative intensities at 16 min
32: relative intensities at 32 min
64: relative intensities at 64 min
ID: unique IDs
position: genome positions
filtration: indicator wether the replicate is filtered or not

Source

https://github.com/CyanolabFreiburg/rifi

`res_minimal` *The result of event_dataframe for E.coli artificial example. A data frame combining the processed genome annotation and a SummarizedExperiment data from rifi_stats. The dataframe is*

Description

The result of event_dataframe for E.coli artificial example. A data frame combining the processed genome annotation and a SummarizedExperiment data from rifi_stats. The dataframe is

Usage

```
data(res_minimal)
```

Format

A list with 2 items:

region: the region from the gff file

gene: the annotated gene name

locus_tag: the annotated locus tag

strand: the strand of the annotation

TU: The overarching transcription unit

position: The bin/probe specific position

FC_fragment_intensity:

FC_intensity:

p_value_intensity:

FC_fragment_HL:

FC_HL:

p_value_HL:

FC_HL_intensity_fragment:

FC_HL_intensity:

FC_HL_adapted:

p_value_Manova:

synthesis_ratio:

synthesis_ratio_event:

pausing_site:

iTSS_I:

event_ps_itss_p_value_Ttest:

ps_ts_fragment:

event_position:

event_duration:

delay_frg_slope:

p_value_slope:

delay:

half_life:

intensity:

Source

https://github.com/CyanolabFreiburg/rifi

rifi_fit *rifi_fit: conveniently wraps all fitting steps*

Description

Wraps the functions: nls2_fit, TI_fit, plot_nls2_function and plot_singleProbe_function.

Usage

```
rifi_fit(
  inp,
  cores = 1,
  viz = FALSE,
  restr = 0.2,
  decay = seq(0.08, 0.11, by = 0.02),
  delay = seq(0, 10, by = 0.1),
  k = seq(0.1, 1, 0.2),
  bg = 0.2,
  TI_k = seq(0, 1, by = 0.5),
  TI_decay = c(0.05, 0.1, 0.2, 0.5, 0.6),
  TI = seq(0, 1, by = 0.5),
  TI_delay = seq(0, 2, by = 0.5),
  TI_rest_delay = seq(0, 2, by = 0.5),
  TI_bg = 0
)
```

Arguments

inp	SummarizedExperiment: the input with correct format.
cores	integer: the number of assigned cores for the task.
viz	logical: whether to visualize the output.
restr	numeric: a parameter that restricts the freedom of the fit to avoid wrong TI-term_factors, ranges from 0 to 0.2
decay	numeric vector: A sequence of starting values for the decay. Default is seq(.08, 0.11, by=.02)
delay	numeric vector: A sequence of starting values for the delay. Default is seq(0,10, by=.1)
k	numeric vector: A sequence of starting values for the synthesis rate.Default is seq(0.1,1,0.2)
bg	numeric vector: A sequence of starting values. Default is 0.2.
TI_k	numeric vector: A sequence of starting values for the synthesis rate. Default is seq(0, 1, by = 0.5).
TI_decay	numeric vector: A sequence of starting values for the decay. Default is c(0.05, 0.1, 0.2, 0.5, 0.6).

TI	numeric vector: A sequence of starting values for the TI. Default is seq(0, 1, by = 0.5).
TI_delay	numeric vector: A sequence of starting values for the delay. Default is seq(0, 2, by = 0.5).
TI_rest_delay	numeric vector: A sequence of starting values. Default is seq(0, 2, by = 0.5).
TI_bg	numeric vector: A sequence of starting values. Default is 0.

Value

the SummarizedExperiment object: with delay, decay and TI_termination_factor added to the rowRanges. The full fit data is saved in the metadata as "fit_STD" and "fit_TI". A plot is given if viz = TRUE.

See Also

`nls2_fit`

`TI_fit`

`plot_nls2`

`plot_singleProbe`

Examples

```
data(preprocess_minimal)
rifi_fit(
  inp = preprocess_minimal,
  cores = 1, viz = FALSE, restr = 0.1,
  decay = seq(.08, 0.11, by = .02),
  delay = seq(0, 10, by = .1), k = seq(0.1, 1, 0.2), bg = 0.2,
  TI_k = seq(0, 1, by = 0.5), TI_decay = c(0.05, 0.1, 0.2, 0.5, 0.6),
  TI = seq(0, 1, by = 0.5), TI_delay = seq(0, 2, by = 0.5),
  TI_rest_delay = seq(0, 2, by = 0.5), TI_bg = 0
)
```

---

rifi_fragmentation *rifi_fragmentation: conveniently wraps all fragmentation steps*

Description

rifi_fragmentation wraps the following functions:

1. fragment_delay
2. fragment_HL
3. fragment_inty
4. TUgether
5. fragment_TI

Usage

```
rifi_fragmentation(
  inp,
  cores = 1,
  pen_delay = NULL,
  pen_out_delay = NULL,
  pen_HL = NULL,
  pen_out_HL = NULL,
  pen_inty = NULL,
  pen_out_inty = NULL,
  pen_TU = NULL,
  pen_TI = NULL,
  pen_out_TI = NULL
)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
cores	integer: the number of assigned cores for the task.
pen_delay	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
pen_out_delay	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.
pen_HL	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
pen_out_HL	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.
pen_inty	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
pen_out_inty	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.
pen_TU	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default -0.75.
pen_TI	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default is the auto generated value.
pen_out_TI	numeric: an internal parameter for the dynamic programming. Higher values result in fewer allowed outliers. Default is the auto generated value.

Value

the SummarizedExperiment object: with delay_fragment, HL_fragment, intensity_fragment, TI_termination_fragment and TU, and the respective values added to the rowRanges.

See Also

fragment_delay

fragment_HL

fragment_inty

TUgether

fragment_TI

Examples

```
data(penalties_minimal)
rifi_fragmentation(inp = penalties_minimal, cores = 2)
```

---

rifi_penalties *rifi_penalties: conveniently wraps all penalty steps*

Description

wraps the functions: make_pen and viz_pen_obj.

Usage

```
rifi_penalties(
  inp,
  details = FALSE,
  viz = FALSE,
  top_i = 25,
  cores = 1,
  dpt = 1,
  smpl_min = 10,
  smpl_max = 100,
  sta_pen = 0.5,
  end_pen = 4.5,
  rez_pen = 9,
  sta_pen_out = 0.5,
  end_pen_out = 4.5,
  rez_pen_out = 9
)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
details	logical: whether to return the penalty objects or just the logbook.
viz	logical: whether to visualize the output or not. Default is FALSE
top_i	integer: the number of top results visualized. Default is all.

cores	integer: the number of assigned cores for the task.
dpt	integer: the number of times a full iteration cycle is repeated with a more narrow range based on the previous cycle. Default is 2.
smpl_min	integer: the smaller end of the sampling size. Default is 10.
smpl_max	integer: the larger end of the sampling size. Default is 100.
sta_pen	numeric: the lower starting penalty. Default is 0.5.
end_pen	numeric: the higher starting penalty. Default is 4.5.
rez_pen	numeric: the number of penalties iterated within the penalty range. Default is 9.
sta_pen_out	numeric: the lower starting outlier penalty. Default is 0.5.
end_pen_out	numeric: the higher starting outlier penalty. Default is 3.5.
rez_pen_out	numeric: the number of outlier penalties iterated within the outlier penalty range. Default is 7.

Value

the SummarizedExperiment object: with the penalties in the logbook added to the metadata. Also adds logbook_details if details is TRUE, and plots the penalties if viz is TRUE.

See Also

make_pen

viz_pen_obj

Examples

```
data(fit_minimal)
rifi_penalties(
  inp = fit_minimal, details = FALSE, viz = FALSE,
  top_i = 25, cores = 2, dpt = 1, smpl_min = 10, smpl_max = 100,
  sta_pen = 0.5, end_pen = 4.5, rez_pen = 9, sta_pen_out = 0.5,
  end_pen_out = 4.5, rez_pen_out = 9
)
```

---

rifi_preprocess *rifi_preprocess: conveniently wraps all pre-processing steps. Wraps the functions:*

1. *check_input*
2. *make_df*
3. *function_seg*
4. *finding_PDD*
5. *finding_TI Allows for the optional integration of filter functions. Filter functions mark replicates with TRUE. Those are then not considered in the fit! FUN_filter is a general filter usually to exclude probes with low expression or "bad" patterns.*

Description

rifi_preprocess: conveniently wraps all pre-processing steps. Wraps the functions:

1. check_input
2. make_df
3. function_seg
4. finding_PDD
5. finding_TI Allows for the optional integration of filter functions. Filter functions mark replicates with TRUE. Those are then not considered in the fit! FUN_filter is a general filter usually to exclude probes with low expression or "bad" patterns.

Usage

```
rifi_preprocess(
  inp,
  cores,
  FUN_filter = function(x) {     FALSE },
  bg = 0,
  rm_FLT = FALSE,
  thrsh_check = 0,
  dista = 300,
  run_PDD = FALSE,
  pen_PDD = 2,
  pen_out_PDD = 1,
  thrsh_PDD = 0.001,
  pen_TI = 10,
  thrsh_TI = 0.5,
  add = 1000
)
```

Arguments

inp	SummarizedExperiment: the input.
cores	integer: the number of assigned cores for the task.
FUN_filter	function: A function of x, returning a logical. x is the numeric vector of the intensity from all time points for a specific replicate.
bg	numeric: threshold over which the last time point has to be to be fitted with the above background mode.
rm_FLT	logical: remove IDs where all replicates are marked as filtered by the background check. Default is FALSE.
thrsh_check	numeric: the minimal allowed intensity for time point "0". Advised to be kept at 0! Default is 0.
dista	integer: the amount of nucleotides defining the gap. Default is 300.
run_PDD	logical: running the PDD flag function

pen_PDD	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Advised to be kept at 2. Default is 2.
pen_out_PDD	numeric: an internal parameter for the dynamic programming. Higher values result in fewer possible outliers. Advised to be kept at 1. Default is 1.
thrsh_PDD	numeric: an internal parameter that allows fragments with slopes steeper than the threshold to be flagged with "*PDD*". Higher values result in fewer candidates . Advised to be kept at 0.001. Default is 0.001.
pen_TI	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Advised to be kept at 10. Default is 10.
thrsh_TI	numeric: an internal parameter that allows fragments with a certain amount of IDs with higher relative intensities at time points later than "0" to be flagged as "*TI*". Higher values result in fewer candidates. -0.5 is 25 %, 0 is 50%, 0.5 is 75%. Advised to be kept at 0.5. Default is 0.5.
add	integer: range of nucleotides before a potential TI event where in IDs are fitted with the TI fit.

Value

the SummarizedExperiment object: checked, and with position, ID, intensity, probe_TI, position_segment, flag and filtration added to the rowRanges.

See Also

`check_input`

`make_df`

`segment_pos`

`finding_PDD`

`finding_TI`

Examples

```
data(example_input_minimal)
rifi_preprocess(
  inp = example_input_minimal, cores = 2, bg = 100, rm_FLT = FALSE,
  thrsh_check = 0, dista = 300, run_PDD = FALSE
  )
```

`rifi_stats` *rifi_stats: conveniently wraps all statistical prediction steps. Wraps the functions: predict_ps_itss, apply_Ttest_delay, apply_ancova, apply_event_position, apply_t_test, fold_change, apply_manova, apply_t_test_ti and gff3_preprocess.*

Description

rifi_stats: conveniently wraps all statistical prediction steps. Wraps the functions: predict_ps_itss, apply_Ttest_delay, apply_ancova, apply_event_position, apply_t_test, fold_change, apply_manova, apply_t_test_ti and gff3_preprocess.

Usage

```
rifi_stats(inp, dista = 300, path)
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
dista	integer: the maximal distance allowed between two successive fragments. Default is the auto generated value.
path	path: to the directory containing the gff3 file.

Value

the probe data frame with the columns regarding statistics:

ID: The bin/probe specific ID

position: The bin/probe specific position

strand: The bin/probe specific strand

intensity: The relative intensity at time point 0

probe_TI: An internal value to determine which fitting model is applied

flag: Information on which fitting model is applied

position_segment: The position based segment

delay: The delay value of the bin/probe

half_life: The half-life of the bin/probe

TI_termination_factor: The termination factor of the bin/probe

delay_fragment: The delay fragment the bin belongs to

velocity_fragment: The velocity value of the respective delay fragment

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

TI_termination_fragment: The TI fragment the bin belongs to

TI_mean_termination_factor: The mean termination factor of the respective TI fragment

seg_ID: The combined ID of the fragment

pausing_site:
iTSS_I:
ps_ts_fragment:
event_ps_itss_p_value_Ttest:
p_value_slope:
delay_frg_slope:
velocity_ratio:
event_duration:
event_position:
FC_HL:
FC_fragment_HL:
p_value_HL:
FC_intensity:
FC_fragment_intensity:
p_value_intensity:
FC_HL_intensity:
FC_HL_intensity_fragment:
FC_HL_adapted:
synthesis_ratio:
synthesis_ratio_event:
p_value_Manova:
p_value_TI:
TI_fragments_p_value:

See Also

predict_ps_itss
apply_Ttest_delay
apply_ancova
apply_event_position
apply_t_test
fold_change
apply_manova
apply_t_test_ti
gff3_preprocess

Examples

```
data(fragmentation_minimal)
rifi_stats(inp = fragmentation_minimal, dista = 300,
path = gzfile(system.file("extdata", "gff_e_coli.gff3.gz",
package = "rifi")))
```

rifi_summary	*rifi_summary: conveniently wraps and summarize all rifi outputs. Wraps the functions: event_dataframe, dataframe_summary, dataframe_summary_events, dataframe_summary_events_HL_int, dataframe_summary_events_ps_itss, dataframe_summary_events_velocity and dataframe_summary_TI.*

Description

rifi_summary: conveniently wraps and summarize all rifi outputs. Wraps the functions: event_dataframe, dataframe_summary, dataframe_summary_events, dataframe_summary_events_HL_int, dataframe_summary_events_ps_itss dataframe_summary_events_velocity and dataframe_summary_TI.

Usage

```
rifi_summary(inp, data_annotation = metadata(inp)$annot[[1]])
```

Arguments

inp	SummarizedExperiment: the input data frame with correct format.
data_annotation	dataframe: gff3 dataframe after processing.

Value

WIP

See Also

event_dataframe

dataframe_summary

dataframe_summary_events

dataframe_summary_events_HL_int

dataframe_summary_events_ps_itss

dataframe_summary_events_velocity

dataframe_summary_TI

Examples

```
data(stats_minimal)
if(!require(SummarizedExperiment)){
suppressPackageStartupMessages(library(SummarizedExperiment))
}
rifi_summary(inp = stats_minimal, data_annotation =
metadata(stats_minimal)$annot[[1]])
```

rifi_visualization *rifi_visualization: plots all the data with fragments and events from both strands. rifi_visualization: plots the whole genome with genes, transcription units (TUs), delay, half-life (HL), intensity fragments features, events, velocity, annotation, coverage if available. rifi_visualization uses several functions to plot the genes including as-RNA and ncRNA and TUs as segments. The function plots delay, HL and intensity fragments with statistical t-test between the neighboring fragment, significant t-test is assigned with ''.* t-test and Manova statistical test are also depicted as ''. *The functions used are: annotation_plot: plots the corresponding annotation positive_strand_function: plots delay, HL, intensity and events of positive strand negative_strand_function: plots delay, HL, intensity and events of negative strand empty_data_positive: plots empty boxes in case no data is available for positive strand empty_data_negative: plots empty boxes in case no data is available for negative strand strand_selection: check if data is stranded and arrange by position. splitGenome_function: splits the genome into fragments indice_function: assign a new column to the data to distinguish between fragments, outliers from delay or HL or intensity. TU_annotation: designs the segments border for the genes and TUs annotation gene_annot_function: it requires gff3 file, returns a dataframe adjusting each fragment according to its annotation. It allows as well the plot of genes and TUs shared into two pages label_log2_function: used to add log scale to intensity values. label_square_function: used to add square scale to coverage values. coverage_function: this function is used only in case of coverage is available. secondaryAxis: adjusts the half-life or delay to 20 in case of the dataframe row numbers is equal to 1 and the half-life or delay exceed the limit, they are plotted with different shape and color. add_genomeBorders: when the annotated genes are on the borders, they can not be plotted, therefore the region was split in 2 adding the row corresponding to the split part to the next annotation (i + 1) except for the first page. my_arrow: creates an arrow for the annotation. arrange_byGroup: selects the last row for each segment and add 40 nucleotides in case of negative strand for a nice plot. regr: plots the predicted delay from linear regression if the data is on negative strand meanPosition: assign a mean position for the plot. delay_mean: adds a column in case of velocity is NA or equal to 60. The mean of the delay is calculated outliers. my_segment_T: plots terminals and pausing sites labels. my_segment_NS: plots internal starting sites 'iTSS'. min_value: returns minimum value for event plots in intensity plot. velocity_fun: function for velocity plot limit_function: for values above 10 or 20 in delay and hl. Limit of the axis is set differently. y-axis limit is applied only if we have more than 3 values above 10 and lower or equal to 20. An exception is added in case a dataframe has less than 3 rows and 1 or more values are above 10, the rest of the values above 20 are adjusted to 20 on "secondaryAxis" function. empty_boxes: used only in case the dataframe from the positive strand is not empty, the TU are annotated. function_TU_arrow: used to avoid plotting arrows when a TU is split into two pages. terminal_plot_lm: draws a linear regression line when terminal outliers have an intensity above a certain threshold and are consecutive. Usually are smallRNA (ncRNA, asRNA). slope_function: replaces slope lower than 0.0009 to 0. velo_function: replaces infinite velocity with NA. plot the coverage of RNA_seq in exponential phase growth*

Description

rifi_visualization: plots all the data with fragments and events from both strands. rifi_visualization: plots the whole genome with genes, transcription units (TUs), delay, half-life (HL), intensity fragments features, events, velocity, annotation, coverage if available. rifi_visualization uses several functions to plot the genes including as-RNA and ncRNA and TUs as segments. The function plots delay, HL and intensity fragments with statistical t-test between the neighboring fragment, significant t-test is assigned with ''. *t-test and Manova statistical test are also depicted as ''*. The functions used are: annotation_plot: plots the corresponding annotation positive_strand_function: plots delay, HL, intensity and events of positive strand negative_strand_function: plots delay, HL, intensity and events of negative strand empty_data_positive: plots empty boxes in case no data is available for positive strand empty_data_negative: plots empty boxes in case no data is available for negative strand strand_selection: check if data is stranded and arrange by position. splitGenome_function: splits the genome into fragments indice_function: assign a new column to the data to distinguish between fragments, outliers from delay or HL or intensity. TU_annotation: designs the segments border for the genes and TUs annotation gene_annot_function: it requires gff3 file, returns a dataframe adjusting each fragment according to its annotation. It allows as well the plot of genes and TUs shared into two pages label_log2_function: used to add log scale to intensity values. label_square_function: used to add square scale to coverage values. coverage_function: this function is used only in case of coverage is available. secondaryAxis: adjusts the half-life or delay to 20 in case of the dataframe row numbers is equal to 1 and the half-life or delay exceed the limit, they are plotted with different shape and color. add_genomeBorders: when the annotated genes are on the borders, they can not be plotted, therefore the region was split in 2 adding the row corresponding to the split part to the next annotation (i + 1) except for the first page. my_arrow: creates an arrow for the annotation. arrange_byGroup: selects the last row for each segment and add 40 nucleotides in case of negative strand for a nice plot. regr: plots the predicted delay from linear regression if the data is on negative strand meanPosition: assign a mean position for the plot. delay_mean: adds a column in case of velocity is NA or equal to 60. The mean of the delay is calculated outliers. my_segment_T: plots terminals and pausing sites labels. my_segment_NS: plots internal starting sites 'iTSS'. min_value: returns minimum value for event plots in intensity plot. velocity_fun: function for velocity plot limit_function: for values above 10 or 20 in delay and hl. Limit of the axis is set differently. y-axis limit is applied only if we have more than 3 values above 10 and lower or equal to 20. An exception is added in case a dataframe has less than 3 rows and 1 or more values are above 10, the rest of the values above 20 are adjusted to 20 on "secondaryAxis" function. empty_boxes: used only in case the dataframe from the positive strand is not empty, the TU are annotated. function_TU_arrow: used to avoid plotting arrows when a TU is split into two pages. terminal_plot_lm: draws a linear regression line when terminal outliers have an intensity above a certain threshold and are consecutive. Usually are smallRNA (ncRNA, asRNA). slope_function: replaces slope lower than 0.0009 to 0. velo_function: replaces infinite velocity with NA. plot the coverage of RNA_seq in exponential phase growth

Usage

```
rifi_visualization(
  data,
  genomeLength,
  annot,
  coverage = 0,
  chr_fwd = NA,
```

```
  chr_rev = NA,
  region = c("CDS", "asRNA", "5'UTR", "ncRNA", "3'UTR", "tRNA"),
  color_region = c("grey0", "red", "blue", "orange", "yellow", "green", "white",
    "darkseagreen1", "grey50", "black"),
  color_text.1 = "grey0",
  color_text.2 = "black",
  color_TU = "blue",
  Alpha = 0.5,
  size_tu = 1.6,
  size_locusTag = 1.6,
  size_gene = 1.6,
  Limit = 10,
  shape = 22,
  col_outiler = "grey50",
  col_coverage = "grey",
  shape_outlier = 13,
  limit_intensity = NA,
  face = "bold",
  tick_length = 0.3,
  arrow.color = "darkseagreen1",
  minVelocity = 3000,
  medianVelocity = 6000,
  col_above20 = "#00FFFF",
  fontface = "plain",
  shape_above20 = 14,
  axis_text_y_size = 3,
  axis_title_y_size = 6,
  TI_threshold = 1.1,
  termination_threshold = 0.8,
  iTSS_threshold = 1.2,
  p_value_int = 0.05,
  p_value_event = 0.05,
  p_value_hl = 0.05,
  p_value_TI = 0.05,
  p_value_manova = 0.05,
  event_duration_ps = 1,
  event_duration_itss = -1,
  HL_threshold_1 = log2(1.5),
  HL_threshold_2 = -log2(1.5),
  vel_threshold = 200,
  HL_threshold_color = "black",
  vel_threshold_color = "grey52",
  ps_color = "orange",
  iTSS_I_color = "blue"
)
```

Arguments

data	SummarizedExperiment: the input data frame with correct format.

`genomeLength` integer: genome length output of gff3_preprocess function and element of metadata of SummarizedExperiment.

`annot` dataframe: the annotation file, output of gff3_preprocess function and element of metadata of SummarizedExperiment.

`coverage` integer: in case the coverage is available.

`chr_fwd` string object: coverage of the forward strand.

`chr_rev` string object: coverage of the reverse strand.

`region` dataframe: gff3 features of the genome.

`color_region` string vector: vector of colors.

`color_text.1` string: TU color text

`color_text.2` string: genes color text

`color_TU` string. TU color

`Alpha` integer: color transparency degree.

`size_tu` integer: TU size

`size_locusTag` integer: locus_tag size

`size_gene` integer: font size for gene annotation.

`Limit` integer: value for y-axis limit.

`shape` integer: value for shape.

`col_outiler` string: outlier color.

`col_coverage` integer: color for coverage plot.

`shape_outlier` integer: value for outlier shape.

`limit_intensity` integer: intensity limit if applicable.

`face` string: label font.

`tick_length` integer: value for ticks.

`arrow.color` string: arrows color.

`minVelocity` integer: threshold to fix the minimum of velocity.

`medianVelocity` integer: threshold to fix the maximum of velocity.

`col_above20` string: color for probes/bin above value 20.

`fontface` integer: font type

`shape_above20` integer: shape for probes/bins above value 20.

`axis_text_y_size` integer: text size for y-axis.

`axis_title_y_size` integer: title size for y-axis.

`TI_threshold` integer: threshold for TI between two fragments in case the TI termination factor drops from the first segment to the second, default 1.1.

`termination_threshold` integer: threshold for termination to plot, default .8.

`iTSS_threshold` integer: threshold for iTSS_II selected to plot, default 1.2.

`p_value_int` integer: p_value of intensity fragments fold-change to plot, default 0.05.

`p_value_event` integer: p_value of t-test from pausing site and iTSS_I events to plot, default 0.05.

`p_value_hl` integer: p_value of half_life fragments fold-change to plot, default 0.05.

`p_value_TI` integer: p_value of TI fragments selected to be plotted, default 0.05.

`p_value_manova` integer: p_value of manova test fragments to plot, default 0.05.

`event_duration_ps` integer: threshold for pausing sites selected to plot, default -2.

`event_duration_itss` integer: threshold for iTSS_I selected to plot, default 2.

`HL_threshold_1` integer: threshold for log2FC(HL) selected to plot, default log2(1.5). log2FC(HL) >= log2(1.5) are indicated by black color. If p_value <= p_value_hl (default 0.05), log2FC(HL) is indicated by HL* otherwise HL.

`HL_threshold_2` integer: threshold for log2FC(HL) selected to plot, default -log2(1.5). log2FC(HL) <= -log2(1.5) are indicated by green color. If p_value <= p_value_hl (default 0.05), log2FC(HL) is indicated by HL* otherwise HL. In case of p_value is significant and the log2FC(HL) is between -log2FC(1.5) and log2FC(1.5), FC is assigned by green color and HL*.

`vel_threshold` integer: threshold for velocity ratio selected to plot, default 200.

`HL_threshold_color` string: color for HL fold change plot.

`vel_threshold_color` string: color for velocity ratio plot.

`ps_color` string: color for pausing site plot.

`iTSS_I_color` string: color for iTSS_I plot.

Value

The visualization plot.

Examples

```
data(stats_minimal)
if(!require(SummarizedExperiment)){
suppressPackageStartupMessages(library(SummarizedExperiment))
}
rifi_visualization(data = stats_minimal,
genomeLength = metadata(stats_minimal)$annot[[2]],
annot = metadata(stats_minimal)$annot[[1]])
```

`rifi_wrapper` *rifi_wrapper: conveniently wraps all functions included on rifi workflow. Wraps the functions: rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats, rifi_summary and rifi_visualization.*

Description

rifi_wrapper: conveniently wraps all functions included on rifi workflow. Wraps the functions: rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats, rifi_summary and rifi_visualization.

Usage

```
rifi_wrapper(inp, cores, path, bg, restr)
```

Arguments

`inp`	data frame: the input data frame with correct format.
`cores`	integer: the number of assigned cores for the task.
`path`	path: path to an annotation file in gff format.
`bg`	numeric: threshold over which the last time point has to be to be fitted with the above background mode.
`restr`	numeric: a parameter that restricts the freedom of the fit to avoid wrong TI-term_factors, ranges from 0 to 0.2

Value

All intermediate objects

See Also

`rifi_preprocess`

`rifi_fit`

`rifi_penalties`

`rifi_fragmentation`

`rifi_stats`

`rifi_summary`

`rifi_visualization`

Examples

```
data(example_input_minimal)
rifi_wrapper(inp = example_input_minimal, cores = 2, path =
gzfile(system.file("extdata", "gff_e_coli.gff3.gz", package = "rifi")),
bg = 0, restr = 0.01)
```

segment_pos *segment_pos: divides all IDs by position into position_segments. segment_pos adds the column "position_segment" to the rowRanges. To reduce run time, the data is divided by regions of no expression larger than "dist" nucleotides.*

Description

segment_pos: divides all IDs by position into position_segments. segment_pos adds the column "position_segment" to the rowRanges. To reduce run time, the data is divided by regions of no expression larger than "dist" nucleotides.

Usage

```
segment_pos(inp, dista = 300)
```

Arguments

inp	SummarizedExperiment: the input.
dista	integer: the amount of nucleotides defining the gap. Default is 300.

Value

the SummarizedExperiment object: with position_segment added to the rowRanges.

Examples

```
data(preprocess_minimal)
segment_pos(inp = preprocess_minimal, dista = 300)
```

stats_e_coli *The result of rifi_stats for E.coli example data A SummarizedExperiment containing the output from rifi_stats*

Description

The result of rifi_stats for E.coli example data A SummarizedExperiment containing the output from rifi_stats

Usage

```
data(stats_e_coli)
```

Format

A SummarizedExperiment:

ID: The bin/probe specific ID
position: The bin/probe specific position
strand: The bin/probe specific strand
intensity: The relative intensity at time point 0
probe_TI: An internal value to determine which fitting model is applied
flag: Information on which fitting model is applied
position_segment: The position based segment
delay: The delay value of the bin/probe
half_life: The half-life of the bin/probe
TI_termination_factor: The termination factor of the bin/probe
delay_fragment: The delay fragment the bin belongs to
velocity_fragment: The velocity value of the respective delay fragment
intercept: The vintercept of fit through the respective delay fragment
slope: The slope of the fit through the respective delay fragment
HL_fragment: The half-life fragment the bin belongs to
HL_mean_fragment: The mean half-life value of the respective half-life fragment
intensity_fragment: The intensity fragment the bin belongs to
intensity_mean_fragment: The mean intensity value of the respective intensity fragment
TU: The overarching transcription unit
TI_termination_fragment: The TI fragment the bin belongs to
TI_mean_termination_factor: The mean termination factor of the respective TI fragment
seg_ID: The combined ID of the fragment
pausing_site:
iTSS_I:
ps_ts_fragment:
event_ps_itss_p_value_Ttest:
p_value_slope:
delay_frg_slope:
velocity_ratio:
event_duration:
event_position:
FC_HL:
FC_fragment_HL:
p_value_HL:
FC_intensity:

FC_fragment_intensity:

p_value_intensity:

FC_HL_intensity:

FC_HL_intensity_fragment:

FC_HL_adapted:

synthesis_ratio:

synthesis_ratio_event:

p_value_Manova:

p_value_TI:

TI_fragments_p_value:

Source

https://github.com/CyanolabFreiburg/rifi

`stats_minimal` *The result of rifi_stats for artificial example data A SummarizedExperiment containing the output of rifi_stats as an extention to rowRanges and metadata (gff file processed, see gff file documentation)*

Description

The result of rifi_stats for artificial example data A SummarizedExperiment containing the output of rifi_stats as an extention to rowRanges and metadata (gff file processed, see gff file documentation)

Usage

```
data(stats_minimal)
```

Format

A rowRanges of SummarizedExperiment with 24 rows and 45 variables:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

probe_TI: An internal value to determine which fitting model is applied

flag: Information on which fitting model is applied

position_segment: The position based segment

delay: The delay value of the bin/probe

half_life: The half-life of the bin/probe

TI_termination_factor: The termination factor of the bin/probe

delay_fragment: The delay fragment the bin belongs to

velocity_fragment: The velocity value of the respective delay fragment

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

TI_termination_fragment: The TI fragment the bin belongs to

TI_mean_termination_factor: The mean termination factor of the respective TI fragment

seg_ID: The combined ID of the fragment

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

event_duration:

event_position:

FC_HL:

FC_fragment_HL:

p_value_HL:

FC_intensity:

FC_fragment_intensity:

p_value_intensity:

FC_HL_intensity:

FC_HL_intensity_fragment:

FC_HL_adapted:

synthesis_ratio:

synthesis_ratio_event:

p_value_Manova:

p_value_TI:

TI_fragments_p_value:

Source

https://github.com/CyanolabFreiburg/rifi

`stats_synechocystis_6803` *The result of rifi_stats for Synechocystis 6803 example data A SummarizedExperiment containing the output of rifi_stats as an extention to rowRanges*

Description

The result of rifi_stats for Synechocystis 6803 example data A SummarizedExperiment containing the output of rifi_stats as an extention to rowRanges

Usage

```
data(stats_synechocystis_6803)
```

Format

The rowRanges of SummarizedExperiment:

ID: The bin/probe specific ID

position: The bin/probe specific position

intensity: The relative intensity at time point 0

probe_TI: An internal value to determine which fitting model is applied

flag: Information on which fitting model is applied

position_segment: The position based segment

delay: The delay value of the bin/probe

half_life: The half-life of the bin/probe

TI_termination_factor: The termination factor of the bin/probe

delay_fragment: The delay fragment the bin belongs to

velocity_fragment: The velocity value of the respective delay fragment

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

HL_fragment: The half-life fragment the bin belongs to

HL_mean_fragment: The mean half-life value of the respective half-life fragment

intensity_fragment: The intensity fragment the bin belongs to

intensity_mean_fragment: The mean intensity value of the respective intensity fragment

TU: The overarching transcription unit

TI_termination_fragment: The TI fragment the bin belongs to

TI_mean_termination_factor: The mean termination factor of the respective TI fragment

seg_ID: The combined ID of the fragment

pausing_site:

iTSS_I:

ps_ts_fragment:

event_ps_itss_p_value_Ttest:

p_value_slope:

delay_frg_slope:

velocity_ratio:

event_duration:

event_position:

FC_HL:

FC_fragment_HL:

p_value_HL:

FC_intensity:

FC_fragment_intensity:

p_value_intensity:

FC_HL_intensity:

FC_HL_intensity_fragment:

FC_HL_adapted:

synthesis_ratio:

synthesis_ratio_event:

p_value_Manova:

p_value_TI:

TI_fragments_p_value:

Source

https://github.com/CyanolabFreiburg/rifi

`summary_e_coli` *The result of rifi_summary for E.coli example data A SummarizedExperiment containing the output of rifi_stats as an extention to rowRanges*

Description

The result of rifi_summary for E.coli example data A SummarizedExperiment containing the output of rifi_stats as an extention to rowRanges

Usage

```
data(summary_e_coli)
```

Format

The rowRanges of SummarizedExperiment:

bin_df: all information regarding bins:

ID:
feature_type:
gene:
locus_tag:
position:
strand: The bin/probe specific strand
segment: The segment the bin/probe belongs to
TU: The overarching transcription unit
delay_fragment: The delay fragment the bin/probe belongs to
delay: The delay of the bin/probe
HL_fragment: The half-life fragment the bin/probe belongs to
half_life: The half-life of the bin/probe
intensity_fragment: The intensity fragment the bin/probe belongs to
intensity: The relative intensity at time point 0
flag: The flag of the bin/probe(TI, PDD)
TI_termination_factor: The TI_termination_factor of the bin/probe (in case TI is detected)

frag_df: all information regarding fragments:

feature_type:
gene:
locus_tag:
first_position_frg: The first position of the fragment on the genome
last_position_frg: The last position of the fragment on the genome
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment: The segment the fragment belongs to
delay_fragment: The delay fragment of the fragment
HL_fragment: The half-life fragment of the fragment
half_life: The half-life mean of the fragment
HL_SD: The half-life standard deviation of the fragment
HL_SE: The half-life standard error of the fragment
intensity_fragment: The intensity_fragment of the fragment
intensity: The relative intensity at time point 0
intensity_SD: The intensity standard deviation of the fragment
intensity_SE: The intensity standard error of the fragment
velocity: The velocity value of the respective delay fragment

event_df: all information regarding events:

event:
p_value:

p_adjusted:
FC_HL: Fold change of half-life
FC_intensity: Fold change of intensity
FC_HL_adapted: Fold change of half-life/ fold change of intensity, position of the half-life fragment is adapted to intensity fragment
FC_HL_FC_intensity: Fold change of half-life/ fold change of intensity
event_position:
velocity_ratio:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

events_HL_int_df: all information regarding events related to half-life and intensity:

event:
p_value:
p_adjusted:
FC_HL:
FC_intensity:
FC_HL_adapted: Fold change of half-life/ fold change of intensity, position of the half-life fragment is adapted to intensity fragment
FC_HL_FC_intensity: Fold change of half-life/ fold change of intensity
event_position:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

events_ps_itss_df: all information regarding events related to pausing sites and iTSS_I:

event:
p_value:
p_adjusted:

event_position:
velocity_ratio:
FC_HL_adapted:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

events_velocity_df: all information regarding events related to velocity:

event:
p_value:
p_adjusted:
event_position:
velocity_ratio:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

TI_df: all information regarding TI:

event:
TI_fragment:
TI_termination_factor:
p_value:
p_adjusted:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
features:
event_position:
position_1:
position_2:

Source

https://github.com/CyanolabFreiburg/rifi

---

summary_minimal *The result of rifi_summary for artificial example data A SummarizedExperiment with the output from rifi_summary as metadata*

Description

The result of rifi_summary for artificial example data A SummarizedExperiment with the output from rifi_summary as metadata

Usage

```
data(summary_minimal)
```

Format

A list of 7 data frames with 290 rows and 11 variables, 36 rows and 11 variables, 57 rows and 18 variables, and 8 rows and 14 variables:

bin_df: all information regarding bins:

- **ID:**
- **feature_type:**
- **gene:**
- **locus_tag:**
- **position:**
- **strand:** The bin/probe specific strand
- **segment:** The segment the bin/probe belongs to
- **TU:** The overarching transcription unit
- **delay_fragment:** The delay fragment the bin/probe belongs to
- **delay:** The delay of the bin/probe
- **HL_fragment:** The half-life fragment the bin/probe belongs to
- **half_life:** The half-life of the bin/probe
- **intensity_fragment:** The intensity fragment the bin/probe belongs to
- **intensity:** The relative intensity at time point 0
- **flag:** The flag of the bin/probe(TI, PDD)
- **TI_termination_factor:** The TI_termination_factor of the bin/probe (in case TI is detected)

frag_df: all information regarding fragments:

- **feature_type:**
- **gene:**
- **locus_tag:**
- **first_position_frg:** The first position of the fragment on the genome

last_position_frg: The last position of the fragment on the genome

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment: The segment the fragment belongs to

delay_fragment: The delay fragment of the fragment

HL_fragment: The half-life fragment of the fragment

half_life: The half-life mean of the fragment

HL_SD: The half-life standard deviation of the fragment

HL_SE: The half-life standard error of the fragment

intensity_fragment: The intensity_fragment of the fragment

intensity: The relative intensity at time point 0

intensity_SD: The intensity standard deviation of the fragment

intensity_SE: The intensity standard error of the fragment

velocity: The velocity value of the respective delay fragment

event_df: all information regarding events:

event:

p_value:

p_adjusted:

FC_HL: Fold change of half-life

FC_intensity: Fold change of intensity

FC_HL_adapted: Fold change of half-life/ fold change of intensity, position of the half-life fragment is adapted to intensity fragment

FC_HL_FC_intensity: Fold change of half-life/ fold change of intensity

event_position:

velocity_ratio:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_duration:

gap_fragments:

features:

events_HL_int_df: all information regarding events related to half-life and intensity:

event:

p_value:

p_adjusted:

FC_HL:

FC_intensity:

FC_HL_adapted: Fold change of half-life/ fold change of intensity, position of the half-life fragment is adapted to intensity fragment

FC_HL_FC_intensity: Fold change of half-life/ fold change of intensity

event_position:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_duration:

gap_fragments:

features:

events_ps_itss_df: all information regarding events related to pausing sites and iTSS_I:

event:

p_value:

p_adjusted:

event_position:

velocity_ratio:

FC_HL_adapted:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_duration:

gap_fragments:

features:

events_velocity_df: all information regarding events related to velocity:

event:

p_value:

p_adjusted:

event_position:

velocity_ratio:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

segment_1:

segment_2:

event_duration:

gap_fragments:

features:

TI_df: all information regarding TI:

event:

TI_fragment:

TI_termination_factor:

p_value:

p_adjusted:

feature_type:

gene:

locus_tag:

strand: The bin/probe specific strand

TU: The overarching transcription unit

features:

event_position:

position_1:

position_2:

Source

https://github.com/CyanolabFreiburg/rifi

`summary_synechocystis_6803`

The result of rifi_summary for Synechocystis 6803 example data A list containing the output from rifi_summary, including the fragment based data frame, bin based data frame, event data frame and the TI dataframe.

Description

The result of rifi_summary for Synechocystis 6803 example data A list containing the output from rifi_summary, including the fragment based data frame, bin based data frame, event data frame and the TI dataframe.

Usage

```
data(summary_synechocystis_6803)
```

Format

A list of 4 data frames with 3000 rows and 11 variables, 297 rows and 11 variables, 486 rows and 18 variables, and 10 rows and 14 variables:

bin_df: all information regarding bins:

- **ID:**
- **feature_type:**
- **gene:**
- **locus_tag:**
- **position:**
- **strand:** The bin/probe specific strand
- **segment:** The segment the bin/probe belongs to
- **TU:** The overarching transcription unit
- **delay_fragment:** The delay fragment the bin/probe belongs to
- **delay:** The delay of the bin/probe
- **HL_fragment:** The half-life fragment the bin/probe belongs to
- **half_life:** The half-life of the bin/probe
- **intensity_fragment:** The intensity fragment the bin/probe belongs to
- **intensity:** The relative intensity at time point 0
- **flag:** The flag of the bin/probe(TI, PDD)
- **TI_termination_factor:** The TI_termination_factor of the bin/probe (in case TI is detected)

frag_df: all information regarding fragments:

- **feature_type:**
- **gene:**
- **locus_tag:**
- **first_position_frg:** The first position of the fragment on the genome
- **last_position_frg:** The last position of the fragment on the genome
- **strand:** The bin/probe specific strand
- **TU:** The overarching transcription unit
- **segment:** The segment the fragment belongs to
- **delay_fragment:** The delay fragment of the fragment
- **HL_fragment:** The half-life fragment of the fragment
- **half_life:** The half-life mean of the fragment
- **HL_SD:** The half-life standard deviation of the fragment
- **HL_SE:** The half-life standard error of the fragment
- **intensity_fragment:** The intensity_fragment of the fragment
- **intensity:** The relative intensity at time point 0
- **intensity_SD:** The intensity standard deviation of the fragment
- **intensity_SE:** The intensity standard error of the fragment
- **velocity:** The velocity value of the respective delay fragment

event_df: all information regarding events:

- **event:**

p_value:
p_adjusted:
FC_HL: Fold change of half-life
FC_intensity: Fold change of intensity
FC_HL_adapted: Fold change of half-life/ fold change of intensity, position of the half-life fragment is adapted to intensity fragment
FC_HL_FC_intensity: Fold change of half-life/ fold change of intensity
event_position:
velocity_ratio:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

events_HL_int_df: all information regarding events related to half-life and intensity:

event:
p_value:
p_adjusted:
FC_HL:
FC_intensity:
FC_HL_adapted: Fold change of half-life/ fold change of intensity, position of the half-life fragment is adapted to intensity fragment
FC_HL_FC_intensity: Fold change of half-life/ fold change of intensity
event_position:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

events_ps_itss_df: all information regarding events related to pausing sites and iTSS_I:

event:
p_value:

p_adjusted:
event_position:
velocity_ratio:
FC_HL_adapted:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

events_velocity_df: all information regarding events related to velocity:

event:
p_value:
p_adjusted:
event_position:
velocity_ratio:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit
segment_1:
segment_2:
event_duration:
gap_fragments:
features:

TI_df: all information regarding TI:

event:
TI_fragment:
TI_termination_factor:
p_value:
p_adjusted:
feature_type:
gene:
locus_tag:
strand: The bin/probe specific strand
TU: The overarching transcription unit

features:
event_position:
position_1:
position_2:

Source

https://github.com/CyanolabFreiburg/rifi

TI_fit

TI_fit: estimates transcription interference and termination factor using nls function for probe or bin flagged as "TI". TI_fit uses nls2 function to fit the flagged probes or bins with "TI" found using finding_TI.r. It estimates the transcription interference level (referred later to TI) as well as the transcription factor fitting the probes/bins with nls function looping into several starting values. To determine TI and termination factor, TI_fit function is applied to the flagged probes and to the probes localized 1000 nucleotides upstream. Before applying TI_fit function, some probes/bins are filtered out if they are below the background using generic_filter_BG. The model loops into a dataframe containing sequences of starting values and the coefficients are extracted from the fit with the lowest residuals. When many residuals are equal to 0, the lowest residual can not be determined and the coefficients extracted could be wrong. Therefore, a second filter was developed. First we loop into all starting values, we collect nls objects and the corresponding residuals. They are sorted and residuals non equal to 0 are collected in a vector. If the first residuals are not equal to 0, 20 % of the best residuals are collected in tmp_r_min vector and the minimum termination factor is selected. In case the first residuals are equal to 0 then values between 0 to 20% of the values collected in tmp_r_min vector are gathered. The minimum termination factor coefficient is determined and saved. The coefficients are gathered in res vector and saved as an object.

Description

TI_fit: estimates transcription interference and termination factor using nls function for probe or bin flagged as "TI". TI_fit uses nls2 function to fit the flagged probes or bins with "TI" found using finding_TI.r. It estimates the transcription interference level (referred later to TI) as well as the transcription factor fitting the probes/bins with nls function looping into several starting values. To determine TI and termination factor, TI_fit function is applied to the flagged probes and to the probes localized 1000 nucleotides upstream. Before applying TI_fit function, some probes/bins are filtered out if they are below the background using generic_filter_BG. The model loops into a dataframe containing sequences of starting values and the coefficients are extracted from the fit with the lowest residuals. When many residuals are equal to 0, the lowest residual can not be determined and the coefficients extracted could be wrong. Therefore, a second filter was developed. First we loop into all starting values, we collect nls objects and the corresponding residuals. They

are sorted and residuals non equal to 0 are collected in a vector. If the first residuals are not equal to 0, 20 % of the best residuals are collected in tmp_r_min vector and the minimum termination factor is selected. In case the first residuals are equal to 0 then values between 0 to 20% of the values collected in tmp_r_min vector are gathered. The minimum termination factor coefficient is determined and saved. The coefficients are gathered in res vector and saved as an object.

Usage

```
TI_fit(
  inp,
  cores = 1,
  restr = 0.2,
  k = seq(0, 1, by = 0.5),
  decay = c(0.05, 0.1, 0.2, 0.5, 0.6),
  ti = seq(0, 1, by = 0.5),
  ti_delay = seq(0, 2, by = 0.5),
  rest_delay = seq(0, 2, by = 0.5),
  bg = 0
)
```

Arguments

inp	SummarizedExperiment: the input with correct format.
cores	integer: the number of assigned cores for the task.
restr	numeric: a parameter that restricts the freedom of the fit to avoid wrong TI-term_factors, ranges from 0 to 0.2.
k	numeric vector: A sequence of starting values for the synthesis rate. Default is seq(0, 1, by = 0.5).
decay	numeric vector: A sequence of starting values for the decay Default is c(0.05, 0.1, 0.2, 0.5, 0.6).
ti	numeric vector: A sequence of starting values for the delay. Default is seq(0, 1, by = 0.5).
ti_delay	numeric vector: A sequence of starting values for the delay. Default is seq(0, 2, by = 0.5).
rest_delay	numeric vector: A sequence of starting values. Default is seq(0, 2, by = 0.5).
bg	numeric vector: A sequence of starting values. Default is 0.

Value

the SummarizedExperiment object: with delay, decay and TI_termination_factor added to the rowRanges. The full fit data is saved in the metadata as "fit_TI".

Examples

```
data(preprocess_minimal)
TI_fit(inp = preprocess_minimal, cores=2, restr=0.01)
```

`TUgether` *TUgether: combines delay fragments into TUs.*

Description

TUgether combines delay fragments into TUs. The column "TU" is added.

Usage

```
TUgether(inp, cores = 1, pen = -0.75)
```

Arguments

`inp`	SummarizedExperiment: the input data frame with correct format.
`cores`	cores: integer: the number of assigned cores for the task.
`pen`	numeric: an internal parameter for the dynamic programming. Higher values result in fewer fragments. Default -0.75.

Details

TUgether combines delay fragments into TUs. It uses score fun_increasing on the start and end points of delay_fragments. The function used is: .score_fun_increasing The input is the SummarizedExperiment object. pen is the penalty for new fragments in the dynamic programming. Since high scores are aimed, pen is negative.

Value

the SummarizedExperiment with the columns regarding the TU:

ID: The bin/inp specific ID

position: The bin/inp specific position

position_segment: The position based segment

delay_fragment: The delay fragment the bin belongs to

intercept: The vintercept of fit through the respective delay fragment

slope: The slope of the fit through the respective delay fragment

TU: The overarching transcription unit

Examples

```
data(fragmentation_minimal)
TUgether(inp = fragmentation_minimal, cores = 2, pen = -0.75)
```

viz_pen_obj *viz_pen_obj: visualizes penalty objects. An optional visualization of any penalty object created by make_pen. Can be customized to show only the n = top_i top results.*

Description

viz_pen_obj: visualizes penalty objects. An optional visualization of any penalty object created by make_pen. Can be customized to show only the n = top_i top results.

Usage

```
viz_pen_obj(obj, top_i = nrow(obj[[3]][[1]]) * ncol(obj[[3]][[1]]))
```

Arguments

obj	object: penalty object(make_pen output)
top_i	integer: the number of top results visualized. Default is all.

Value

A visualization of the penalty object

Examples

```
data(penalties_e_coli)
viz_pen_obj(penalties_e_coli$pen_obj_delay,25)
```

wrapper_e_coli *The result of rifi_wrapper for E.coli example data A list of SummarizedExperiment containing the output of rifi_wrapper. The list contains 6 elements of SummarizedExperiment output of rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats and rifi_summary. The plot is generated from rifi_visualization. for more detail, please refer to each function separately.*

Description

The result of rifi_wrapper for E.coli example data A list of SummarizedExperiment containing the output of rifi_wrapper. The list contains 6 elements of SummarizedExperiment output of rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats and rifi_summary. The plot is generated from rifi_visualization. for more detail, please refer to each function separately.

Usage

```
data(wrapper_e_coli)
```

Format

An object of class `list` of length 6.

Source

https://github.com/CyanolabFreiburg/rifi

`wrapper_minimal` *The result of rifi_wrapper for E.coli artificial example. A list of SummarizedExperiment containing the output of rifi_wrapper. The list contains 6 elements of SummarizedExperiment output of rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats and rifi_summary. The plot is generated from rifi_visualization. for more detail, please refer to each function separately.*

Description

The result of rifi_wrapper for E.coli artificial example. A list of SummarizedExperiment containing the output of rifi_wrapper. The list contains 6 elements of SummarizedExperiment output of rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats and rifi_summary. The plot is generated from rifi_visualization. for more detail, please refer to each function separately.

Usage

```
data(wrapper_minimal)
```

Format

An object of class `list` of length 6.

Source

https://github.com/CyanolabFreiburg/rifi

wrapper_summary_synechocystis_6803

The result of rifi_wrapper for summary_synechocystis_6803 example data A list of SummarizedExperiment containing the output of rifi_wrapper. The list contains 6 elements of SummarizedExperiment output of rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats and rifi_summary. The plot is generated from rifi_visualization. for more detail, please refer to each function separately.

Description

The result of rifi_wrapper for summary_synechocystis_6803 example data A list of SummarizedExperiment containing the output of rifi_wrapper. The list contains 6 elements of SummarizedExperiment output of rifi_preprocess, rifi_fit, rifi_penalties, rifi_fragmentation, rifi_stats and rifi_summary. The plot is generated from rifi_visualization. for more detail, please refer to each function separately.

Usage

```
data(wrapper_summary_synechocystis_6803)
```

Format

An object of class `list` of length 6.

Source

https://github.com/CyanolabFreiburg/rifi

Index