Package ‘transite’

January 9, 2024

Title RNA-binding protein motif analysis

Version 1.20.0

Maintainer Konstantin Krismer <krismer@mit.edu>

Description transite is a computational method that allows comprehensive analysis of the regulatory role of RNA-binding proteins in various cellular processes by leveraging preexisting gene expression data and current knowledge of binding preferences of RNA-binding proteins.

License MIT + file LICENSE

URL https://transite.mit.edu

Depends R (>= 3.5)

Imports BiocGenerics (>= 0.26.0), Biostrings (>= 2.48.0), dplyr (>= 0.7.6), GenomicRanges (>= 1.32.6), ggplot2 (>= 3.0.0), ggseqlogo (>= 0.1), grDevices, gridExtra (>= 2.3), methods, parallel, Rcpp (>= 1.0.4.8), stats, TFMPvalue (>= 0.0.8), utils

Suggests knitr (>= 1.20), markdown (>= 1.10), roxygen2 (>= 6.1.0), testthat (>= 2.1.0)

LinkingTo Rcpp (>= 1.0.4.8)

VignetteBuilder knitr

biocViews GeneExpression, Transcription, DifferentialExpression, Microarray, mRNAMicroarray, Genetics, GeneSetEnrichment

Encoding UTF-8

RoxygenNote 7.1.1

SystemRequirements C++11

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

git_branch RELEASE_3_18

git_last_commit 7cc2cb6

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18
Date/Publication 2024-01-08

Author Konstantin Krismer [aut, cre, cph] (https://orcid.org/0000-0001-8994-3416), Anna Gattinger [aut] (https://orcid.org/0000-0001-7094-9279), Michael Yaffe [ths, cph] (https://orcid.org/0000-0002-9547-3251), Ian Cannell [ths] (https://orcid.org/0000-0001-5832-9210)

R topics documented:

calculate_kmer_enrichment .................................................. 3
calculate_local_consistency ................................................. 4
calculate_motif_enrichment .................................................. 5
calculate_transcript_mc ....................................................... 7
check_kmers ................................................................. 8
classify_spectrum ........................................................... 9
compute_kmer_enrichment .................................................... 11
count_homopolymer_corrected_kmers .................................... 13
create_kmer_motif .......................................................... 14
create_matrix_motif ........................................................ 14
draw_volcano_plot .......................................................... 15
estimate_significance ....................................................... 17
estimate_significance_core ................................................ 18
ge ................................................................. 19
generate_iupac_by_kmers .................................................. 19
generate_iupac_by_matrix ................................................ 20
generate_kmers ............................................................ 22
generate_kmers_from_iupac .............................................. 23
generate_permuted_enrichments ....................................... 24
geometric_mean ............................................................ 25
get_motifs ............................................................... 25
get_motifs_meta_info ..................................................... 26
get_motif_by_id .......................................................... 27
get_motif_by_rbp ........................................................ 27
get_ppm ................................................................. 28
init_iupac_lookup_table ................................................... 29
kmers_enrichment ........................................................ 30
motifs ................................................................. 30
p_combine ............................................................... 31
RBPotif-class ............................................................ 33
run_kmer_spma .......................................................... 35
run_kmer_tisma ........................................................ 38
run_matrix_spma ......................................................... 40
run_matrix_tisma ........................................................ 44
score_sequences .......................................................... 48
score_spectrum .......................................................... 48
score_transcripts ......................................................... 52
score_transcripts_single_motif ....................................... 54
**calculate_kmer_enrichment**

*k-mer Enrichment between Foreground and Background Sets*

**Description**

Calls `compute_kmer_enrichment` to compute k-mer enrichment values for multiple foregrounds. Calculates enrichment for foreground sets in parallel.

**Usage**

```r
calculate_kmer_enrichment(
  foreground_sets,
  background_set,
  k,
  permutation = FALSE,
  chisq_p_value_threshold = 0.05,
  p_adjust_method = "BH",
  n_cores = 4
)
```

**Arguments**

- `foreground_sets`: list of foreground sets; a foreground set is a character vector of DNA or RNA sequences (not both) and a strict subset of the `background_set`
- `background_set`: character vector of DNA or RNA sequences that constitute the background set
- `k`: length of k-mer, either 6 for hexamers or 7 for heptamers
- `permutation`: if TRUE, only the enrichment value is returned (efficiency mode used for permutation testing)
- `chisq_p_value_threshold`: threshold below which Fisher's exact test is used instead of Pearson's chi-squared test
- `p_adjust_method`: see `p.adjust`
- `n_cores`: number of computing cores to use
**Value**

A list with two entries:

- dfs: a list of data frames with results from `compute_kmer_enrichment` for each of the foreground sets
- kmers: a character vector of all k-mers

**See Also**

Other k-mer functions: `check_kmers()`, `compute_kmer_enrichment()`, `count_homopolymer_corrected_kmers()`, `draw_volcano_plot()`, `estimate_significance_core()`, `estimate_significance()`, `generate_kmers()`, `generate_permuted_enrichments()`, `run_kmer_spma()`, `run_kmer_tsma()`

**Examples**

```r
# define simple sequence sets for foreground and background
foreground_set2 <- c("UUAUUUA", "AUCCUUUACA", "UUUUAUCA", "UUUAAACU")
foreground_sets <- list(foreground_set1, foreground_set2)
background_set <- c(foreground_set1, foreground_set2, "CCACACAC", "CUCAUUGGAG", "ACUUGGACA", "CAGGUCAGC")

# single-threaded
kmer_enrichment_values_st <- calculate_kmer_enrichment(foreground_sets, background_set, 6, n_cores = 1)
```

```r
# multi-threaded
kmer_enrichment_values_mt <- calculate_kmer_enrichment(foreground_sets, background_set, 6)
```

---

**calculate_local_consistency**

*Local Consistency Score*

**Description**

C++ implementation of Local Consistency Score algorithm.

**Usage**

`calculate_local_consistency(x, numPermutations, minPermutations, e)`
**Arguments**

- **x** numeric vector that contains values for shuffling
- **numPermutations** maximum number of permutations performed in Monte Carlo test for consistency score
- **minPermutations** minimum number of permutations performed in Monte Carlo test for consistency score
- **e** stop criterion for consistency score Monte Carlo test: aborting permutation process after observing e random consistency values with more extreme values than the actual consistency value

**Value**

List with score, p_value, and n components, where score is the raw local consistency score (usually not used), p_value is the associated p-value for that score, obtained by Monte Carlo testing, and n is the number of permutations performed in the Monte Carlo test (the higher, the more significant)

**Examples**

```r
poor_enrichment_spectrum <- c(0.1, 0.5, 0.6, 0.4, 0.7, 0.6, 1.2, 1.1, 1.8, 1.6)
local_consistency <- calculate_local_consistency(poor_enrichment_spectrum, 1000000, 1000, 5)

enrichment_spectrum <- c(0.1, 0.3, 0.6, 0.7, 0.8, 0.9, 1.2, 1.4, 1.6, 1.4)
local_consistency <- calculate_local_consistency(enrichment_spectrum, 1000000, 1000, 5)
```

**Description**

This function is used to calculate binding site enrichment / depletion scores between predefined foreground and background sequence sets. Significance levels of enrichment values are obtained by Monte Carlo tests.

**Usage**

```r
calculate_motif_enrichment(
    foreground_scores_df,
    background_scores_df,
    background_total_sites,
    background_absolute_hits,
)```
calculate_motif_enrichment

n_transcripts_foreground,
max_fg_permutations = 1e+06,
min_fg_permutations = 1000,
e = 5,
p_adjust_method = "BH"
)

Arguments

foreground_scores_df
result of score_transcripts on foreground sequence set (foreground sequence sets must be a subset of the background sequence set)

background_scores_df
result of score_transcripts on background sequence set

background_total_sites
number of potential binding sites per sequence (returned by score_transcripts)

background_absolute_hits
number of putative binding sites per sequence (returned by score_transcripts)

n_transcripts_foreground
number of sequences in the foreground set

max_fg_permutations
maximum number of foreground permutations performed in Monte Carlo test for enrichment score

min_fg_permutations
minimum number of foreground permutations performed in Monte Carlo test for enrichment score

e
integer-valued stop criterion for enrichment score Monte Carlo test: aborting permutation process after observing e random enrichment values with more extreme values than the actual enrichment value

p_adjust_method
adjustment of p-values from Monte Carlo tests to avoid alpha error accumulation, see p.adjust

Value

A data frame with the following columns:

- motif_id: the motif identifier that is used in the original motif library
- motif_rbps: the gene symbol of the RNA-binding protein(s)
- enrichment: binding site enrichment between foreground and background sequences
- p_value: unadjusted p-value from Monte Carlo test
- p_value_n: number of Monte Carlo test permutations
- adj_p_value: adjusted p-value from Monte Carlo test (usually FDR)

See Also

Other matrix functions: run_matrix_spma(), run_matrix_tsma(), score_transcripts_single_motif(), score_transcripts()
**Examples**

```r
foreground_seqs <- c("CAGUCAGACUCC", "AAUUGGUGUCGGAUACUUCCCUGUACAU", "AGAU", "CCAGUA\")
background_seqs <- c(foreground_seqs, "CAACAGCCUUAAUU", "CUUUGGGGAAU", "UCAUUUUAUUAA", "AUCAAAUUA", "GAACUUAAAGAUC\", "UAGCAUAAACUUAUG", "AUGGA", "GAAGAGGUCUA", "AUAGAC", "AGUUC")
foreground_scores <- score_transcripts(foreground_seqs, cache = FALSE)
background_scores <- score_transcripts(background_seqs, cache = FALSE)
enrichments_df <- calculate_motif_enrichment(foreground_scores$df, background_scores$df, background_scores$total_sites, background_scores$absolute_hits, length(foreground_seqs), max_fg_permutations = 1000)
```

---

**calculate_transcript_mc**

*Motif Enrichment calculation*

**Description**

C++ implementation of Motif Enrichment calculation

**Usage**

```r
calculate_transcript_mc(
  absoluteHits, 
  totalSites, 
  relHitsForeground, 
  n, 
  maxPermutations, 
  minPermutations, 
  e
)
```

**Arguments**

- **absoluteHits**: number of putative binding sites per sequence (returned by `score_transcripts`)
- **totalSites**: number of potential binding sites per sequence (returned by `score_transcripts`)
- **relHitsForeground**: relative number of hits in foreground set
- **n**: number of sequences in the foreground set
- **maxPermutations**: maximum number of foreground permutations performed in Monte Carlo test for enrichment score
### check_kmers

**Description**

Checks if the provided set of k-mers is valid. A valid set of k-mers is (1) non-empty, (2) contains either only hexamers or only heptamers, and (3) contains only characters from the RNA alphabet (A, C, G, U)

**Usage**

```r
check_kmers(kmers)
```

**Arguments**

- `kmers`: set of k-mers

**Value**

`TRUE` if set of k-mers is valid
classify_spectrum

See Also

Other k-mer functions: calculate_kmer_enrichment(), compute_kmer_enrichment(), count_homopolymer_corrected_kmers(), draw_volcano_plot(), estimate_significance_core(), estimate_significance(), generate_kmers(), generate_permuted_enrichments(), run_kmer_spma(), run_kmer_tsma()

Examples

# valid set
check_kmers(c("ACGCUC", "AAACCC", "UUUACA"))

# invalid set (contains hexamers and heptamers)
check_kmers(c("ACGCUC", "AAACCC", "UUUACAA"))

classify_spectrum  Simple spectrum classifier based on empirical thresholds

Description

Spectra can be classified based on the aggregate spectrum classifier score. If \( \text{sum}(\text{score}) = 3 \), spectrum considered non-random, random otherwise.

Usage

classify_spectrum(
  adj_r_squared,
  degree,
  slope,
  consistency_score_n,
  n_significant,
  n_bins
)

Arguments

adj_r_squared adjusted \( R^2 \) of polynomial model, returned by score_spectrum
degree degree of polynomial, returned by score_spectrum
slope coefficient of the linear term of the polynomial model (spectrum "direction"), returned by score_spectrum
consistency_score_n number of performed permutations before early stopping, returned by score_spectrum
n_significant number of bins with statistically significant enrichment
n_bins number of bins
classify_spectrum

Value

a three-dimensional binary vector with the following components:

- coordinate 1  \( \text{adj_r_squared} \geq 0.4 \)
- coordinate 2  \( \text{consistency_score_n} > 1000000 \)
- coordinate 3  \( \text{n_significant} \geq \text{floor}(n_{\text{bins}} / 10) \)

See Also

Other SPMA functions: `run_kmer_spma()`, `run_matrix_spma()`, `score_spectrum()`, `subdivide_data()`

Examples

```r
n_bins <- 40

# random spectrum
random_sp <- score_spectrum(runif(n = n_bins, min = -1, max = 1),
                            max_model_degree = 1)
score <- classify_spectrum(
    get_adj_r_squared(random_sp), get_model_degree(random_sp),
    get_model_slope(random_sp), get_consistency_score_n(random_sp), 0, n_bins)
sum(score)

# non-random linear spectrum with strong noise component
signal <- seq(-1, 0.99, 2 / 40)
noise <- rnorm(n = 40, mean = 0, sd = 0.5)
linear_sp <- score_spectrum(signal + noise, max_model_degree = 1,
                            max_cs_permutations = 100000)
score <- classify_spectrum(
    get_adj_r_squared(linear_sp), get_model_degree(linear_sp),
    get_model_slope(linear_sp), get_consistency_score_n(linear_sp), 10, n_bins)
sum(score)
## Not run:
# non-random linear spectrum with weak noise component
signal <- seq(-1, 0.99, 2 / 40)
noise <- rnorm(n = 40, mean = 0, sd = 0.2)
linear_sp <- score_spectrum(signal + noise, max_model_degree = 1,
                            max_cs_permutations = 100000)
score <- classify_spectrum(
    get_adj_r_squared(linear_sp), get_model_degree(linear_sp),
    get_model_slope(linear_sp), get_consistency_score_n(linear_sp), 10, n_bins)
sum(score)
## End(Not run)

# non-random quadratic spectrum with strong noise component
signal <- seq(-1, 0.99, 2 / 40)^2 - 0.5
noise <- rnorm(n = 40, mean = 0, sd = 0.2)
quadratic_sp <- score_spectrum(signal + noise, max_model_degree = 2,
                              max_cs_permutations = 100000)
```

```
compute_kmer_enrichment

k-mer Enrichment between Foreground and Background Sets

Description

Compares foreground sequence set to background sequence set and computes enrichment values for each possible k-mer.

Usage

compute_kmer_enrichment(
  foreground_kmers,
  background_kmers,
  permutation = FALSE,
  chisq_p_value_threshold = 0.05,
  p_adjust_method = "BH"
)

Arguments

foreground_kmers
  k-mer counts of the foreground set (generated by generate_kmers)

background_kmers
  k-mer counts of the background set (generated by generate_kmers)

permutation
  if TRUE, only the enrichment value is returned (efficiency mode used for permutation testing)
compute_kmer_enrichment

chisq_p_value_threshold
threshold below which Fisher's exact test is used instead of Pearson's chi-squared test

p_adjust_method
see p.adjust

Details
Usually uses Pearson's chi-squared test, but recalculates p-values with Fisher's exact test for Pearson's chi-squared test p-values \leq chisq_p_value_threshold. The reason this is done is computational efficiency. Fisher's exact tests are computationally demanding and are only performed in situations, where exact p-values are preferred, e.g., if expected \(< 5 \) or significant p-values.

Value
enrichment of k-mers in specified foreground sequences. A data frame with the following columns is returned:

foreground_count  foreground counts for each k-mer
background_count  background counts for each k-mer
enrichment  k-mer enrichment
p_value  p-value of k-mer enrichment (either from Fisher's exact test or Pearson's chi-squared test)
adj_p_value  multiple testing corrected p-value

See Also
Other k-mer functions: calculate_kmer_enrichment(), check_kmers(), count_homopolymer_corrected_kmers(), draw_volcano_plot(), estimate_significance_core(), estimate_significance(), generate_kmers(), generate_permuted_enrichments(), run_kmer_spma(), run_kmer_tsma()

Examples

# define simple sequence sets for foreground and background
)
)
foreground_kmers <- generate_kmers(foreground_set, 6)
count_homopolymer_corrected_kmers

Correction for Homopolymeric Stretches

Description

Counts all non-overlapping instances of \( k \)-mers in a given set of sequences.

Usage

```r
count_homopolymer_corrected_kmers(sequences, k, kmers, is_rna = FALSE)
```

Arguments

- `sequences`: character vector of DNA or RNA sequences
- `k`: length of \( k \)-mer, either 6 for hexamers or 7 for heptamers
- `kmers`: column sums of return value of `Biostrings::oligonucleotideFrequency(sequences)`
- `is_rna`: if sequences are RNA sequences, this flag needs to be set

Value

Returns a named numeric vector, where the elements are \( k \)-mer counts and the names are \( k \)-mers.

See Also

Other \( k \)-mer functions: `calculate_kmer_enrichment()`, `check_kmers()`, `compute_kmer_enrichment()`, `draw_volcano_plot()`, `estimate_significance_core()`, `estimate_significance()`, `generate_kmers()`, `generate_permuted_enrichments()`, `run_kmer_spma()`,
create_kmer_motif  
*Creates Transite motif object from character vector of k-mers*

**Description**
Takes a position weight matrix (PWM) and meta info and returns an object of class RBPMotif.

**Usage**
```r
create_kmer_motif(id, rbps, kmers, type, species, src)
```

**Arguments**
- `id`: motif id (character vector of length 1)
- `rbps`: character vector of names of RNA-binding proteins associated with this motif
- `kmers`: character vector of k-mers that are associated with the motif, set of k-mers is valid if (1) all k-mers must have the same length, (2) only hexamers or heptamers allowed, (3) allowed characters are A, C, G, U
- `type`: type of motif (e.g., 'HITS-CLIP', 'EMSA', 'SELEX', etc.)
- `species`: species where motif was discovered (e.g., 'Homo sapiens')
- `src`: source of motif (e.g., 'RBPDB v1.3.1')

**Value**
object of class RBPMotif

**Examples**
```r
custom_motif <- create_kmer_motif(
  "custom_motif", "RBP1",
  c("AAAAAAA", "CAAAAAA"), "HITS-CLIP",
  "Homo sapiens", "user"
)
```

create_matrix_motif  
*Creates Transite motif object from position weight matrix*

**Description**
Takes a position weight matrix (PWM) and meta info and returns an object of class RBPMotif.

**Usage**
```r
create_matrix_motif(id, rbps, matrix, type, species, src)
```
**Arguments**

- **id**
  - motif id (character vector of length 1)
- **rbps**
  - character vector of names of RNA-binding proteins associated with this motif
- **matrix**
  - data frame with four columns (A, C, G, U) and 6 - 15 rows (positions), where cell (i, j) contains weight of nucleotide j on position i
- **type**
  - type of motif (e.g., 'HITS-CLIP', 'EMSA', 'SELEX', etc.)
- **species**
  - species where motif was discovered (e.g., 'Homo sapiens')
- **src**
  - source of motif (e.g., 'RBPDB v1.3.1')

**Value**

- object of class RBPMotif

**Examples**

```r
custom_motif <- create_matrix_motif(
  "custom_motif", "RBP1",
  transite::toy_motif_matrix, "HITS-CLIP",
  "Homo sapiens", "user"
)
```

---

**Description**

Uses a volcano plot to visualize k-mer enrichment. X-axis is log2 enrichment value, y-axis is log10 significance, i.e., multiple testing corrected p-value from Fisher’s exact test or Pearson’s chi-squared test.

**Usage**

```r
draw_volcano_plot( 
  kmers, 
  motif_kmers, 
  motif_rbps, 
  significance_threshold = 0.01, 
  show_legend = TRUE
)
```
Arguments

- **kmers**: data frame with the following columns: kmer, adj_p_value, enrichment
- **motif_kmers**: set of k-mers that are associated with a certain motif, will be highlighted in volcano plot
- **motif_rbps**: name of RNA-binding proteins associated with highlighted k-mers (character vector of length 1)
- **significance_threshold**: p-value threshold for significance, e.g., .05 or .01
- **show_legend**: whether or not a legend should be shown

Value

volcano plot

See Also

Other TSMA functions: run_kmer_tsma(), run_matrix_tsma()

Other k-mer functions: calculate_kmer_enrichment(), check_kmers(), compute_kmer_enrichment(), count_homopolymer_corrected_kmers(), estimate_significance_core(), estimate_significance(), generate_kmers(), generate_permuted_enrichments(), run_kmer_spma(), run_kmer_tsma()

Examples

```r
motif <- get_motif_by_id("951_12324455")
draw_volcano_plot(transite:::kmers_enrichment, get_hexamers(motif[[1]]),
                   get_rbps(motif[[1]]))
## Not run:
foreground_set <- c("UGUGGG", "GUGGGG", "GUGUGG", "UGUGGU")
background_set <- unique(c(foreground_set, c("CAACGCUUUAUU", "CAGUCAAGACUCC", "CUUUGGGAAU",
                                             "UCAUUUUAUJAA", "AAAUGUGUCUGGUAAUCUCCUGUACAU",
                                             "AUCAAAUUA", "AGAU", "GACACUUAAAGAAUCCU",
                                             "UAGCAUUAACUAUG", "AUGG", "GAAGAGUCUCA",
                                             "AUAGAC", "AGUUC", "CCAGUAA",
                                             "CCACACAC", "CUCAUUGGAG", "ACUUCCCAAC", "CAGGCACAGCA",
                                             "CCACACACAG", "CCCAACACUCAG", "CACACACUC", "CACCCCCCACAGC"
                                             )))

motif <- get_motif_by_id("M178_0.6")
results <- run_kmer_tsma(list(foreground_set), background_set,
                          motifs = motif)
draw_volcano_plot(results[[1]]$motif_kmers_dfs[[1]],
                   get_hexamers(motif[[1]]), "test RBP")
## End(Not run)
```
estimate_significance  Permutation Test Based Significance of Observed Mean

Description

estimate_significance returns an estimate of the significance of the observed mean, given a set of random permutations of the data.

Usage

estimate_significance(
  actual_mean,
  motif_kmers,
  random_permutations,
  alternative = c("two-sided", "less", "greater"),
  conf_level = 0.95,
  produce_plot = TRUE
)

Arguments

- actual_mean: observed mean
- motif_kmers: set of k-mers that were used to compute the actual_mean
- random_permutations: a set of random permutations of the original data, used to generate an empirical null distribution.
- alternative: side of the test, one of the following: "two-sided", "less", "greater"
- conf_level: confidence level for the returned confidence interval
- produce_plot: if distribution plot should be part of the returned list

Value

A list with the following components:

- p_value_estimate: the estimated p-value of the observed mean
- conf_int: the confidence interval around that estimate
- plot: plot of the empirical distribution of geometric means of the enrichment values

See Also

Other k-mer functions: calculate_kmer_enrichment(), check_kmers(), compute_kmer_enrichment(), count_homopolymer_corrected_kmers(), draw_volcano_plot(), estimate_significance_core(), generate_kmers(), generate_permuted_enrichments(), run_kmer_spma(), run_kmer_tsma()
estimate_significance_core

Significance of Observed Mean

Description

estimate_significance_core returns an estimate of the significance of the observed mean, given a vector of means based on random permutations of the data.

Usage

estimate_significance_core(
    random_means,
    actual_mean,
    alternative = c("two_sided", "less", "greater"),
    conf_level = 0.95
)

Arguments

random_means numeric vector of means based on random permutations of the data (empirical null distribution)
actual_mean observed mean
alternative side of the test, one of the following: "two_sided", "less", "greater"
conf_level confidence level for the returned confidence interval

Value

A list with the following components:

p_value_estimate the estimated p-value of the observed mean
conf_int the confidence interval around that estimate

See Also

Other k-mer functions: calculate_kmer_enrichment(), check_kmers(), compute_kmer_enrichment(), count_homopolymer_corrected_kmers(), draw_volcano_plot(), estimate_significance(), generate_kmers(), generate_permuted_enrichments(), run_kmer_spma(), run_kmer_tsma()

Examples

test_sd <- 1.0
test_null_distribution <- rnorm(n = 10000, mean = 1.0, sd = test_sd)
estimate_significance_core(test_null_distribution, test_sd * 2, "greater")
Description

This object contains a toy data set based on gene expression measurements and 3’-UTR sequences of 1000 genes. It comprises three data frames with RefSeq identifiers, log fold change values, and 3’-UTR sequences of genes, which are either upregulated or downregulated after some hypothetical treatment, as well as all measured genes. The actual values are not important. This data set merely serves as an example input for various functions.

Usage

data(ge)

Format

A list with the following components:

- foreground1_df: data frame that contains down-regulated genes after treatment
- foreground2_df: data frame that contains up-regulated genes after treatment
- background_df: data frame that contains all genes measured

generate_iupac_by_kmers

Generates IUPAC code for a character vector of k-mers

Description

Generates a compact logo of a motif based on IUPAC codes given by a character vector of k-mers

Usage

generate_iupac_by_kmers(kmers, code = NULL)

Arguments

- kmers: character vector of k-mers
- code: if IUPAC code table has already been initialized by init_iupac_lookup_table, it can be specified here
Details

IUPAC RNA nucleotide code:

- A: Adenine
- C: Cytosine
- G: Guanine
- U: Uracil
- R: A or G
- Y: C or U
- S: G or C
- W: A or U
- K: G or U
- M: A or C
- B: C or G or U
- D: A or G or U
- H: A or C or U
- V: A or C or G
- N: any base

Value

the IUPAC string of the binding site

References

http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html

See Also

Other motif functions: \texttt{generate_iupac_by_matrix()}, \texttt{generate_kmers_from_iupac()}, \texttt{get_motif_by_id()}, \texttt{get_motif_by_rbp()}, \texttt{get_motifs_meta_info()}, \texttt{get_motifs()}, \texttt{get_ppm()}, \texttt{init_iupac_lookup_table()}, \texttt{set_motifs()}

Examples

\begin{verbatim}
generate_iupac_by_kmers(c("AACCAA", "AACCGG", "CACCGA"))
\end{verbatim}

\begin{verbatim}
generate_iupac_by_matrix(matrix, threshold = 0.215, code = NULL)
\end{verbatim}

Description

Generates a compact logo of a motif based on IUPAC codes given by a position weight matrix

Usage

\texttt{generate_iupac_by_matrix(matrix, threshold = 0.215, code = NULL)}
Arguments

- **matrix**: the position probability matrix of an RNA-binding protein
- **threshold**: the threshold probability (nucleotides with lower probabilities are ignored)
- **code**: if IUPAC code table has already been initialized by `init_iupac_lookup_table`, it can be specified here

Details

IUPAC RNA nucleotide code:

- A: Adenine
- C: Cytosine
- G: Guanine
- U: Uracil
- R: A or G
- Y: C or U
- S: G or C
- W: A or U
- K: G or U
- M: A or C
- B: C or G or U
- D: A or G or U
- H: A or C or U
- V: A or C or G
- N: any base

Value

the IUPAC string of the binding site

References

[http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html](http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html)

See Also

Other motif functions: `generate_iupac_by_kmers()`, `generate_kmers_from_iupac()`, `get_motif_by_id()`, `get_motif_by_rbp()`, `get_motifs_meta_info()`, `get_motifs()`, `get_ppm()`, `init_iupac_lookup_table()`, `set_motifs()`

Examples

```r
generate_iupac_by_matrix(get_motif_matrix(get_motif_by_id("M178_0.6")[[1]]))
```
generate_kmers

k-mer Counts for Sequence Set

Description
Counts occurrences of k-mers of length k in the given set of sequences. Corrects for homopolymeric stretches.

Usage
generate_kmers(sequences, k)

Arguments
sequences character vector of DNA or RNA sequences
k length of k-mer, either 6 for hexamers or 7 for heptamers

Value
Returns a named numeric vector, where the elements are k-mer counts and the names are DNA k-mers.

Warning
generate_kmers always returns DNA k-mers, even if sequences contains RNA sequences. RNA sequences are internally converted to DNA sequences. It is not allowed to mix DNA and RNA sequences.

See Also
Other k-mer functions: calculate_kmer_enrichment(), check_kmers(), compute_kmer_enrichment(), count_homopolymer_corrected_kmers(), draw_volcano_plot(), estimate_significance_core(), estimate_significance(), generate_permuted_enrichments(), run_kmer_spma(), run_kmer_tsma()

Examples
# count hexamers in set of RNA sequences
rna_sequences <- c(
)
hexamer_counts <- generate_kmers(rna_sequences, 6)
# count heptamers in set of DNA sequences
dna_sequences <- c(
    "CAACAGCCTAATT", "CAGTCAAGACTCC", "CTTTGGGAAT",
    "TCATTATTTAAA", "AATGGGTCTGGATACCTCCTGATACAT",
    "ATCAATTA", "AGAT", "GACACTTAAGATCCT",
    "TAGCATTAACTTAATG", "ATGGA", "GAAGAGTCTCA",
    "ATAGAC", "AGTTC", "CCAGTAA",
    "TTATTTA", "ATCCCTTACA", "TTTTTT", "TTTCATCATT",
    "CCACACAC", "CTCATTGGAG", "ACTTTGGGACA", "CAGGTCAGCA"
)

hexamer_counts <- generate_kmers(dna_sequences, 7)

---

**generate_kmers_from_iupac**

*Generates all k-mers for IUPAC string*

**Description**

Generates all possible k-mers for a given IUPAC string.

**Usage**

```
generate_kmers_from_iupac(iupac, k)
```

**Arguments**

- `iupac` : IUPAC string
- `k` : length of k-mer, 6 (hexamers) or 7 (heptamers)

**Details**

IUPAC RNA nucleotide code:

- **A** : Adenine
- **C** : Cytosine
- **G** : Guanine
- **U** : Uracil
- **R** : A or G
- **Y** : C or U
- **S** : G or C
- **W** : A or U
- **K** : G or U
- **M** : A or C
- **B** : C or G or U
- **D** : A or G or U
- **H** : A or C or U
- **V** : A or C or G
generate_permuted_enrichments


N any base

Value
list of k-mers

References
http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html

See Also
Other motif functions: generate_iupac_by_kmers(), generate_iupac_by_matrix(), get_motif_by_id(),
gen_motif_by_rbp(), get_motifs_meta_info(), get_motifs(), get_ppm(), init_iupac_lookup_table(),
set_motifs() 

Examples

generate_kmers_from_iupac(get_iupac(get_motif_by_id("M178_0.6")[[1]]), k = 6)

generate_permuted_enrichments

Generate Random Permutations of the Enrichment Data

Description
Calculates k-mer enrichment values for randomly sampled (without replacement) foreground sets.

Usage

generate_permuted_enrichments(
  n_transcripts_foreground, 
  background_set, 
  k, 
  n_permutations = 1000, 
  n_cores = 4 
)

Arguments

n_transcripts_foreground
  number of transcripts in the original foreground set

background_set
  character vector of DNA or RNA sequences that constitute the background set

k
  length of k-mer, either 6 for hexamers or 7 for heptamers

n_permutations
  number of permutations to perform

n_cores
  number of computing cores to use
geometric_mean

**Value**

The result of `calculate_kmer_enrichment` for the random foreground sets.

**See Also**

Other k-mer functions: `calculate_kmer_enrichment()`, `check_kmers()`, `compute_kmer_enrichment()`, `count_homopolymer_corrected_kmers()`, `draw_volcano_plot()`, `estimate_significance_core()`, `estimate_significance()`., `generate_kmers()`, `run_kmer_spma()`, `run_kmer_tsma()`

---

<table>
<thead>
<tr>
<th>geometric_mean</th>
<th>Geometric Mean</th>
</tr>
</thead>
</table>

**Description**

Calculates the geometric mean of the specified values.

**Usage**

```r
gemean(x, na_rm = TRUE)
```

**Arguments**

- `x`: numeric vector of values for which the geometric mean will be computed
- `na_rm`: logical. Should missing values (including NaN) be removed?

**Value**

Geometric mean of `x` or 1 if length of `x` is 0

**Examples**

```r
gemean(c(0.123, 0.441, 0.83))
```

---

<table>
<thead>
<tr>
<th>get_motifs</th>
<th>Retrieve list of all motifs</th>
</tr>
</thead>
</table>

**Description**

Retrieves all Transite motifs

**Usage**

```r
get_motifs()
```
get_motifs_meta_info

Value

A list of objects of class Motif

See Also

Other motif functions: generate_iupac_by_kmers(), generate_iupac_by_matrix(), generate_kmers_from_iupac(), get_motif_by_id(), get_motif_by_rbp(), get_motifs_meta_info(), get_ppm(), init_iupac_lookup_table(), set_motifs()

Examples

transite_motifs <- get_motifs()

generate_iupac_by_kmers(), generate_iupac_by_matrix(), generate_kmers_from_iupac(),
get_motif_by_id(), get_motif_by_rbp(), get_motifs_meta_info(), get_ppm(), init_iupac_lookup_table(),
set_motifs()

get_motifs_meta_info()  Displays motif meta information.

Description

Generates a data frame with meta information about all Transite motifs.

Usage

generate_iupac_by_kmers()

Value

A data frame containing meta information for all Transite motifs, with the following columns:

• id
• rbps
• length
• iupac
• type
• species
• src

See Also

Other motif functions: generate_iupac_by_kmers(), generate_iupac_by_matrix(), generate_kmers_from_iupac(),
get_motif_by_id(), get_motif_by_rbp(), get_motifs_meta_info(), get_ppm(), init_iupac_lookup_table(),
set_motifs()

Examples

get_motifs_meta_info()
get_motif_by_id

Retrieve motif objects by id

Description
Retrieves one or more motif objects identified by motif id.

Usage
get_motif_by_id(id)

Arguments
id character vector of motif identifiers

Value
A list of objects of class RBPMotif

See Also
Other motif functions: generate_iupac_by_kmers(), generate_iupac_by_matrix(), generate_kmers_from_iupac(),
get_motif_by_rbp(), get_motifs_meta_info(), get_motifs(), get_ppm(), init_iupac_lookup_table(),
set_motifs()

Examples
get_motif_by_id("M178_0.6")
get_motif_by_id(c("M178_0.6", "M188_0.6"))

get_motif_by_rbp

Retrieve motif objects by gene symbol

Description
Retrieves one or more motif objects identified by gene symbol.

Usage
get_motif_by_rbp(rbp)

Arguments
rbp character vector of gene symbols of RNA-binding proteins
Value

A list of objects of class RBPMotif

See Also

Other motif functions: `generate_iupac_by_kmers()`, `generate_iupac_by_matrix()`, `generate_kmers_from_iupac()`, `get_motif_by_id()`, `get_motifs_meta_info()`, `get_motifs()`, `get_ppm()`, `init_iupac_lookup_table()`, `set_motifs()`

Examples

get_motif_by_rbp("ELAVL1")

get_motif_by_rbp(c("ELAVL1", "ELAVL2"))

get_ppm

Get Position Probability Matrix (PPM) from motif object

Description

Return the position probability matrix of the specified motif.

Usage

get_ppm(motif)

Arguments

motif object of class RBPMotif

Value

The position probability matrix of the specified motif

See Also

Other motif functions: `generate_iupac_by_kmers()`, `generate_iupac_by_matrix()`, `generate_kmers_from_iupac()`, `get_motif_by_id()`, `get_motif_by_rbp()`, `get_motifs_meta_info()`, `get_motifs()`, `init_iupac_lookup_table()`, `set_motifs()`

Examples

get_ppm(get_motif_by_id("M178_0.6")[[1]])
init_iupac_lookup_table

Initializes the IUPAC lookup table

Description

Initializes a hash table that serves as a IUPAC lookup table for the `generate_iupac_by_matrix` function.

Usage

`init_iupac_lookup_table()`

Details

IUPAC RNA nucleotide code:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Adenine</td>
</tr>
<tr>
<td>C</td>
<td>Cytosine</td>
</tr>
<tr>
<td>G</td>
<td>Guanine</td>
</tr>
<tr>
<td>U</td>
<td>Uracil</td>
</tr>
<tr>
<td>R</td>
<td>A or G</td>
</tr>
<tr>
<td>Y</td>
<td>C or U</td>
</tr>
<tr>
<td>S</td>
<td>G or C</td>
</tr>
<tr>
<td>W</td>
<td>A or U</td>
</tr>
<tr>
<td>K</td>
<td>G or U</td>
</tr>
<tr>
<td>M</td>
<td>A or C</td>
</tr>
<tr>
<td>B</td>
<td>C or G or U</td>
</tr>
<tr>
<td>D</td>
<td>A or G or U</td>
</tr>
<tr>
<td>H</td>
<td>A or C or U</td>
</tr>
<tr>
<td>V</td>
<td>A or C or G</td>
</tr>
<tr>
<td>N</td>
<td>any base</td>
</tr>
</tbody>
</table>

Value

an environment, the IUPAC lookup hash table

References

[http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html](http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html)

See Also

Other motif functions: `generate_iupac_by_kmers()`, `generate_iupac_by_matrix()`, `generate_kmers_from_iupac()`, `get_motif_by_id()`, `get_motif_by_rbp()`, `get_motifs_meta_info()`, `get_motifs()`, `get_ppm()`, `set_motifs()`
Examples

```r
generate_iupac_by_matrix(get_motif_matrix(get_motif_by_id("M178_0.6"))[1]),
code = init_iupac_lookup_table()
```

---

**kmers_enrichment**

**Example k-mer Enrichment Data**

**Description**

This data frame with k-mer enrichment data (as produced by `run_kmer_tsma`) is used in a code example for k-mer volcano plot function `draw_volcano_plot`.

**Usage**

```r
data(kmers_enrichment)
```

**Format**

A data frame with the following columns:

<table>
<thead>
<tr>
<th>kmer</th>
<th>contains all hexamers (AAAAAA to UUUUUU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>foreground_count</td>
<td>absolute k-mer frequency in foreground set</td>
</tr>
<tr>
<td>background_count</td>
<td>absolute k-mer frequency in background set</td>
</tr>
<tr>
<td>enrichment</td>
<td>enrichment of k-mer in foreground relative to background</td>
</tr>
<tr>
<td>p_value</td>
<td>associated p-value of enrichment</td>
</tr>
<tr>
<td>adj_p_value</td>
<td>multiple testing corrected p-value</td>
</tr>
</tbody>
</table>

---

**motifs**

**Transite Motif Database**

**Description**

The Transite motif database contains sequence motifs and associated k-mers of more than 100 different RNA-binding proteins, obtained from publicly available motif databases.

**Usage**

```r
data(motifs)
```

**Format**

A list of lists with the following components:
p_combine

id  motif id
rbps  gene symbols of RNA-binding proteins associated with motif
matrix  data frame of sequence motif (position weight matrix)
hexamers  all motif-associated hexamers
heptamers  all motif-associated heptamers
length  length of motif in nucleotides
iupac  IUPAC string of sequence motif
type  type of motif, e.g., RNAcompete
species  usually human
src  source of motif, e.g., RNA Zoo

References

http://cisbp-rna.ccbr.utoronto.ca/
http://rbpdb.ccbr.utoronto.ca/

p_combine  P-value aggregation

Description

p_combine is used to combine the p-values of independent significance tests.

Usage

p_combine(p, method = c("fisher", "SL", "MG", "tippett"), w = NULL)

Arguments

p  vector of p-values
method  one of the following: Fisher (1932) ("fisher"), Stouffer (1949), Liptak (1958) ("SL"), Mudholkar and George (1979) ("MG"), and Tippett (1931) ("tippett")
w  weights, only used in combination with Stouffer-Liptak. If is.null(w) then weights are set in an unbiased way

Details

The problem can be specified as follows: Given a vector of n p-values $p_1, ..., p_n$, find $p_c$, the combined p-value of the n significance tests. Most of the methods introduced here combine the p-values in order to obtain a test statistic, which follows a known probability distribution. The general procedure can be stated as:

$$T(h, C) = \sum_{i=1}^{n} h(p_i) * C$$

The function $T$, which returns the test statistic $t$, takes two arguments. $h$ is a function defined on the interval $[0, 1]$ that transforms the individual p-values, and $C$ is a correction term.
Fisher’s method (1932), also known as the inverse chi-square method is probably the most widely used method for combining p-values. Fisher used the fact that if $p_i$ is uniformly distributed (which p-values are under the null hypothesis), then $-2 \log p_i$ follows a chi-square distribution with two degrees of freedom. Therefore, if p-values are transformed as follows,
\[
h(p) = -2 \log p,
\]
and the correction term $C$ is neutral, i.e., equals 1, the following statement can be made about the sampling distribution of the test statistic $T_f$ under the null hypothesis: $t_f$ is distributed as chi-square with $2n$ degrees of freedom, where $n$ is the number of p-values.

Stouffer’s method, or the inverse normal method, uses a p-value transformation function $h$ that leads to a test statistic that follows the standard normal distribution by transforming each p-value to its corresponding normal score. The correction term scales the sum of the normal scores by the root of the number of p-values.

\[
h(p) = \Phi^{-1}(1 - p)
\]
\[
C = \frac{1}{\sqrt{n}}
\]

Under the null hypothesis, $t_s$ is distributed as standard normal. $\Phi^{-1}$ is the inverse of the cumulative standard normal distribution function.

An extension of Stouffer’s method with weighted p-values is called Liptak’s method. The logit method by Mudholkar and George uses the following transformation:

\[
h(p) = -\ln(p/(1 - p))
\]

When the sum of the transformed p-values is corrected in the following way:

\[
C = \sqrt{\frac{3(5n + 4)}{\pi^2 n(5n + 2)}}
\]

the test statistic $t_m$ is approximately t-distributed with $5n + 4$ degrees of freedom.

In Tippett’s method the smallest p-value is used as the test statistic $t_t$ and the combined significance is calculated as follows:

\[
Pr(t_t) = 1 - (1 - t_t)^n
\]

**Value**

A list with the following components:

<table>
<thead>
<tr>
<th>statistic</th>
<th>the test statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_value</td>
<td>the corresponding p-value</td>
</tr>
<tr>
<td>method</td>
<td>the method used</td>
</tr>
<tr>
<td>statistic_name</td>
<td>the name of the test statistic</td>
</tr>
</tbody>
</table>

**Examples**

\[
p_{\text{combine}}(c(0.01, 0.05, 0.5))
\]
\[
p_{\text{combine}}(c(0.01, 0.05, 0.5), \text{method} = "tippett")
\]
An S4 class to represent a RBPMotif

Description

An S4 class to represent a RBPMotif

Getter Method get_id
Getter Method get_rbps
Getter Method get_motif_matrix
Getter Method get_hexamers
Getter Method get_heptamers
Getter Method get_width
Getter Method get_iupac
Getter Method get_type
Getter Method get_species
Getter Method get_source

Usage

get_id(object)

## S4 method for signature 'RBPMotif'
get_id(object)

get_rbps(object)

## S4 method for signature 'RBPMotif'
get_rbps(object)

get_motif_matrix(object)

## S4 method for signature 'RBPMotif'
get_motif_matrix(object)

get_hexamers(object)

## S4 method for signature 'RBPMotif'
get_hexamers(object)

get_heptamers(object)

## S4 method for signature 'RBPMotif'
get_heptamers(object)
get_width(object)

## S4 method for signature 'RBPMotif'
get_width(object)

get_iupac(object)

## S4 method for signature 'RBPMotif'
get_iupac(object)

get_type(object)

## S4 method for signature 'RBPMotif'
get_type(object)

get_species(object)

## S4 method for signature 'RBPMotif'
get_species(object)

get_source(object)

## S4 method for signature 'RBPMotif'
get_source(object)

## S4 method for signature 'RBPMotif,ANY'
show(object)

## S4 method for signature 'RBPMotif,ANY'
plot(x)

Arguments

object RBPMotif object
x RBPMotif object

Value

Object of type RBPMotif

Slots

id motif id (character vector of length 1)
rbps character vector of names of RNA-binding proteins associated with this motif
matrix data frame with four columns (A, C, G, U) and 6 - 15 rows (positions), where cell (i, j) contains weight of nucleotide j on position i
hexamers character vector of hexamers associated with this motif
heptamers character vector of heptamers associated with this motif
length length of the motif (i.e., nrow(matrix))
iupac IUPAC code for motif matrix (see generate_iupac_by_matrix)
type type of motif (e.g., 'HITS-CLIP', 'EMSA', 'SELEX', etc.)
species species where motif was discovered (e.g., 'Homo sapiens')
src source of motif (e.g., 'RBPDB v1.3.1')

Examples

```r
kmers <- c("AAAAAAA", "CAAAAAA")
iupac <- generate_iupac_by_kmers(kmers,
  code = init_iupac_lookup_table())
hexamers <- generate_kmers_from_iupac(iupac, 6)
heptamers <- generate_kmers_from_iupac(iupac, 7)
new("RBPMotif", id = "custom_motif", rbps = "RBP1",
  matrix = NULL, hexamers = hexamers, heptamers = heptamers, length = 7L,
  iupac = iupac, type = "HITS-CLIP", species = "Homo sapiens", src = "user"
)
```

Description

SPMA helps to illuminate the relationship between RBP binding evidence and the transcript sorting
criterion, e.g., fold change between treatment and control samples.

Usage

```r
run_kmer_spma(
  sorted_transcript_sequences,
  sorted_transcript_values = NULL,
  transcript_values_label = "transcript value",
  motifs = NULL,
  k = 6,
  n_bins = 40,
  midpoint = 0,
  x_value_limits = NULL,
  max_model_degree = 1,
  max_cs_permutations = 1e+07,
  min_cs_permutations = 5000,
  fg_permutations = 5000,
  p_adjust_method = "BH",
  p_combining_method = "fisher",
  n_cores = 1
)
```
run_kmer_spma

Arguments

sorted_transcript_sequences
character vector of ranked sequences, either DNA (only containing upper case
characters A, C, G, T) or RNA (A, C, G, U). The sequences in sorted_transcript_sequences
must be ranked (i.e., sorted). Commonly used sorting criteria are measures of
differential expression, such as fold change or signal-to-noise ratio (e.g., be-
tween treatment and control samples in gene expression profiling experiments).

sorted_transcript_values
vector of sorted transcript values, i.e., the fold change or signal-to-noise ra-
tio or any other quantity that was used to sort the transcripts that were passed
to run_matrix_spma or run_kmer_spma (default value is NULL). These values
are displayed as a semi-transparent area over the enrichment value heatmaps of
spectrum plots.

transcript_values_label
label of transcript sorting criterion (e.g., "log fold change", default value is
"transcript value"), only shown if !is.null(sorted_transcript_values)

motifs
a list of motifs that is used to score the specified sequences. If is.null(motifs) then all Transite motifs are used.

k
length of k-mer, either 6 for hexamers or 7 for heptamers

n_bins
specifies the number of bins in which the sequences will be divided, valid values
are between 7 and 100

midpoint
for enrichment values the midpoint should be 1, for log enrichment values 0
(defaults to 0)

x_value_limits
sets limits of the x-value color scale (used to harmonize color scales of different
spectrum plots), see limits argument of continuous_scale (defaults to NULL, i.e., the data-dependent default scale range)

max_model_degree
maximum degree of polynomial

max_cs_permutations
maximum number of permutations performed in Monte Carlo test for consis-
tency score

min_cs_permutations
minimum number of permutations performed in Monte Carlo test for consist-
tency score

fg_permutations
number of foreground permutations

p_adjust_method
see p.adjust

p_combining_method
one of the following: Fisher (1932) ("fisher"), Stouffer (1949), Liptak (1958)
("SL"), Mudholkar and George (1979) ("MG"), and Tippett (1931) ("tippett")
(see p_combine)

n_cores
number of computing cores to use
Details

In order to investigate how motif targets are distributed across a spectrum of transcripts (e.g., all transcripts of a platform, ordered by fold change), Spectrum Motif Analysis visualizes the gradient of RBP binding evidence across all transcripts.

The k-mer-based approach differs from the matrix-based approach by how the sequences are scored. Here, sequences are broken into k-mers, i.e., oligonucleotide sequences of k bases. And only statistically significantly enriched or depleted k-mers are then used to calculate a score for each RNA-binding protein, which quantifies its target overrepresentation.

Value

A list with the following components:

- foreground_scores: the result of `run_kmer/tsma` for the binned data
- spectrum_info_df: a data frame with the SPMA results
- spectrum_plots: a list of spectrum plots, as generated by `score_spectrum`
- classifier_scores: a list of classifier scores, as returned by `classify_spectrum`

See Also

Other SPMA functions: `classify_spectrum()`, `run_matrix_spma()`, `score_spectrum()`, `subdivide_data()`

Other k-mer functions: `calculate_kmer/enrichment()`, `check_kmers()`, `compute_kmer/enrichment()`, `count_homo/polymer_corrected_kmers()`, `draw_volcano/plot()`, `estimate_significance/core()`, `estimate_significance()`, `generate_kmers()`, `generate_permuted/enrichments()`, `run_kmer/tsma()`

Examples

```r
# example data set
background_df <- transite::ge$background_df
# sort sequences by signal-to-noise ratio
background_df <- dplyr::arrange(background_df, value)
# character vector of named and ranked (by signal-to-noise ratio) sequences
background_seqs <- gsub("T", "U", background_df$seq)
names(background_seqs) <- paste0(background_df$refseq, "|", background_df$seq_type)

results <- run_kmer_spma(background_seqs,
                          sorted_transcript_values = background_df$value,
                          transcript_values_label = "signal-to-noise ratio",
                          motifs = get_motif_by_id("M178_0.6"),
                          n_bins = 20,
                          fg_permutations = 10)

## Not run:
results <- run_kmer_spma(background_seqs,
                          sorted_transcript_values = background_df$value,
                          transcript_values_label = "signal-to-noise ratio")

## End(Not run)
```
**Description**

Calculates the enrichment of putative binding sites in foreground sets versus a background set using \( k \)-mers to identify putative binding sites.

**Usage**

```r
run_kmer_tsma(
  foreground_sets,
  background_set,
  motifs = NULL,
  k = 6,
  fg_permutations = 5000,
  kmer_significance_threshold = 0.01,
  produce_plot = TRUE,
  p_adjust_method = "BH",
  p_combining_method = "fisher",
  n_cores = 1
)
```

**Arguments**

- `foreground_sets`: list of foreground sets; a foreground set is a character vector of DNA or RNA sequences (not both) and a strict subset of the `background_set`
- `background_set`: character vector of DNA or RNA sequences that constitute the background set
- `motifs`: a list of motifs that is used to score the specified sequences. If `is.null(motifs)` then all Transite motifs are used.
- `k`: length of \( k \)-mer, either 6 for hexamers or 7 for heptamers
- `fg_permutations`: number of foreground permutations
- `kmer_significance_threshold`: p-value threshold for significance, e.g., 0.05 or 0.01 (used for volcano plots)
- `produce_plot`: if TRUE volcano plots and distribution plots are created
- `p_adjust_method`: see `p.adjust`
- `p_combining_method`: one of the following: Fisher (1932) ("fisher"), Stouffer (1949), Liptak (1958) ("SL"), Mudholkar and George (1979) ("MG"), and Tippett (1931) ("tippett") (see `p_combine`)
- `n_cores`: number of computing cores to use
Details

Motif transcript set analysis can be used to identify RNA binding proteins, whose targets are significantly overrepresented or underrepresented in certain sets of transcripts.

The aim of Transcript Set Motif Analysis (TSMA) is to identify the overrepresentation and underrepresentation of potential RBP targets (binding sites) in a set (or sets) of sequences, i.e., the foreground set, relative to the entire population of sequences. The latter is called background set, which can be composed of all sequences of the genes of a microarray platform or all sequences of an organism or any other meaningful superset of the foreground sets.

The \( k \)-mer-based approach breaks the sequences of foreground and background sets into \( k \)-mers and calculates the enrichment on a \( k \)-mer level. In this case, motifs are not represented as position weight matrices, but as lists of \( k \)-mers.

Statistically significantly enriched or depleted \( k \)-mers are then used to calculate a score for each RNA-binding protein, which quantifies its target overrepresentation.

Value

A list of lists (one for each transcript set) with the following components:

- \( \text{enrichment}\_\text{df} \) the result of \text{compute_kmer_enrichment}
- \( \text{motif}\_\text{df} \)
- \( \text{motif}\_\text{kmers}\_\text{dfs} \)
- \( \text{volcano}\_\text{plots} \) volcano plots for each motif (see \text{draw_volcano_plot})
- \( \text{perm}\_\text{test}\_\text{plots} \) plots of the empirical distribution of \( k \)-mer enrichment values for each motif
- \( \text{enriched}\_\text{kmers}\_\text{combined}\_\text{p}\_\text{values} \)
- \( \text{depleted}\_\text{kmers}\_\text{combined}\_\text{p}\_\text{values} \)

See Also

Other TSMA functions: \text{draw_volcano_plot()}, \text{run_matrix_tsma()}

Other \( k \)-mer functions: \text{calculate_kmer_enrichment()}, \text{check_kmers()}, \text{compute_kmer_enrichment()}, \text{count_homopolymer_corrected_kmers()}, \text{draw_volcano_plot()}, \text{estimate_significance_core()}, \text{estimate_significance()}, \text{generate_kmers()}, \text{generate_permuted_enrichments()}, \text{run_kmer_spma()}

Examples

# define simple sequence sets for foreground and background
)
foreground_set2 <- c("UUUUGUAU", "AUCCUUAAACU", "UUUUCUUU", "UUUCUAAGACUU"
)
foreground_sets <- list(foreground_set1, foreground_set2)
background_set <- unique(c(foreground_set1, foreground_set2, c("CCACACAC", "UCAUGUGUACUAGUA", "UCAUGUUGGGACA", "CAGGUCGCA", "CCACACCCG", "GUCAUGCU", "GUCAUGCU", "CAGGUCGAGGGCA"))
### run_matrix_spma

Matrix-based Spectrum Motif Analysis

**Description**

SPMA helps to illuminate the relationship between RBP binding evidence and the transcript sorting criterion, e.g., fold change between treatment and control samples.

**Usage**

```r
run_matrix_spma(
  sorted_transcript_sequences,
  sorted_transcript_values = NULL,
  transcript_values_label = "transcript value",
  motifs = NULL,
  n_bins = 40,
  midpoint = 0,
)```

run_matrix_spma

x_value_limits = NULL,
max_model_degree = 1,
max_cs_permutations = 1e+07,
min_cs_permutations = 5000,
max_hits = 5,
threshold_method = "p_value",
threshold_value = 0.25^6,
max_fg_permutations = 1e+06,
min_fg_permutations = 1000,
e = 5,
p_adjust_method = "BH",
n_cores = 1,
 cache = paste0(tempdir(), "/sc/")
)

Arguments

sorted_transcript_sequences
named character vector of ranked sequences (only containing upper case characters A, C, G, T), where the names are RefSeq identifiers and sequence type qualifiers ("3UTR", "5UTR" or "mRNA"), separated by "|", e.g. "NM_010356|3UTR". Names are only used to cache results. The sequences in sorted_transcript_sequences must be ranked (i.e., sorted). Commonly used sorting criteria are measures of differential expression, such as fold change or signal-to-noise ratio (e.g., between treatment and control samples in gene expression profiling experiments).

sorted_transcript_values
vector of sorted transcript values, i.e., the fold change or signal-to-noise ratio or any other quantity that was used to sort the transcripts that were passed to run_matrix_spma or run_kmer_spma (default value is NULL). These values are displayed as a semi-transparent area over the enrichment value heatmaps of spectrum plots.

transcript_values_label
label of transcript sorting criterion (e.g., "log fold change", default value is "transcript value"), only shown if !is.null(sorted_transcript_values)

motifs
a list of motifs that is used to score the specified sequences. If is.null(motifs) then all Transite motifs are used.

n_bins
specifies the number of bins in which the sequences will be divided, valid values are between 7 and 100

midpoint
for enrichment values the midpoint should be 1, for log enrichment values 0 (defaults to 0)

x_value_limits
sets limits of the x-value color scale (used to harmonize color scales of different spectrum plots), see limits argument of continuous_scale (defaults to NULL, i.e., the data-dependent default scale range)

max_model_degree
maximum degree of polynomial

max_cs_permutations
maximum number of permutations performed in Monte Carlo test for consistency score
min_cs_permutations  
minimum number of permutations performed in Monte Carlo test for consistency score

max_hits  
maximum number of putative binding sites per mRNA that are counted

threshold_method  
either "p_value" (default) or "relative". If threshold_method equals "p_value", the default threshold_value is \(0.25^6\), which is lowest p-value that can be achieved by hexamer motifs, the shortest supported motifs. If threshold_method equals "relative", the default threshold_value is 0.9, which is 90% of the maximum PWM score.

threshold_value  
semantics of the threshold_value depend on threshold_method (default is \(0.25^6\))

max_fg_permutations  
maximum number of foreground permutations performed in Monte Carlo test for enrichment score

min_fg_permutations  
minimum number of foreground permutations performed in Monte Carlo test for enrichment score

e  
integer-valued stop criterion for enrichment score Monte Carlo test: aborting permutation process after observing e random enrichment values with more extreme values than the actual enrichment value

p_adjust_method  
adjustment of p-values from Monte Carlo tests to avoid alpha error accumulation, see `p.adjust`

n_cores  
the number of cores that are used

cache  
either logical or path to a directory where scores are cached. The scores of each motif are stored in a separate file that contains a hash table with RefSeq identifiers and sequence type qualifiers as keys and the number of putative binding sites as values. If cache is FALSE, scores will not be cached.

Details

In order to investigate how motif targets are distributed across a spectrum of transcripts (e.g., all transcripts of a platform, ordered by fold change), Spectrum Motif Analysis visualizes the gradient of RBP binding evidence across all transcripts.

The matrix-based approach skips the k-merization step of the k-mer-based approach and instead scores the transcript sequence as a whole with a position specific scoring matrix.

For each sequence in foreground and background sets and each sequence motif, the scoring algorithm evaluates the score for each sequence position. Positions with a relative score greater than a certain threshold are considered hits, i.e., putative binding sites.

By scoring all sequences in foreground and background sets, a hit count for each motif and each set is obtained, which is used to calculate enrichment values and associated p-values in the same way in which motif-compatible hexamer enrichment values are calculated in the k-mer-based approach. P-values are adjusted with one of the available adjustment methods.
An advantage of the matrix-based approach is the possibility of detecting clusters of binding sites. This can be done by counting regions with many hits using positional hit information or by simply applying a hit count threshold per sequence, e.g., only sequences with more than some number of hits are considered. Homotypic clusters of RBP binding sites may play a similar role as clusters of transcription factors.

Value

A list with the following components:

- **foreground_scores**: the result of `score_transcripts` for the foreground sets (the bins)
- **background_scores**: the result of `score_transcripts` for the background set
- **enrichment_dfs**: a list of data frames, returned by `calculate_motif_enrichment`
- **spectrum_info_df**: a data frame with the SPMA results
- **spectrum_plots**: a list of spectrum plots, as generated by `score_spectrum`
- **classifier_scores**: a list of classifier scores, as returned by `classify_spectrum`

See Also

Other SPMA functions: `classify_spectrum()`, `run_kmer_spma()`, `score_spectrum()`, `subdivide_data()`

Other matrix functions: `calculate_motif_enrichment()`, `run_matrix_tsma()`, `score_transcripts_single_motif()`, `score_transcripts()`

Examples

```r
# example data set
background_df <- transite:::ge$background_df
# sort sequences by signal-to-noise ratio
background_df <- dplyr::arrange(background_df, value)
# character vector of named and ranked (by signal-to-noise ratio) sequences
background_seqs <- gsub("T", "U", background_df$seq)
names(background_seqs) <- paste0(background_df$refseq, "|", background_df$seq_type)

results <- run_matrix_spma(background_seqs,
                           sorted_transcript_values = background_df$value,
                           transcript_values_label = "signal-to-noise ratio",
                           motifs = get_motif_by_id("M178_0.6"),
                           n_bins = 20,
                           max_fg_permutations = 10000)

## Not run:
results <- run_matrix_spma(background_seqs,
                           sorted_transcript_values = background_df$value,
                           transcript_values_label = "SNR")

## End(Not run)
```
run_matrix_tsma  
Matrix-based Transcript Set Motif Analysis

Description
Calculates motif enrichment in foreground sets versus a background set using position weight matrices to identify putative binding sites

Usage
```r
run_matrix_tsma(
  foreground_sets,
  background_set,
  motifs = NULL,
  max_hits = 5,
  threshold_method = "p_value",
  threshold_value = 0.25^6,
  max_fg_permutations = 1e+06,
  min_fg_permutations = 1000,
  e = 5,
  p_adjust_method = "BH",
  n_cores = 1,
  cache = paste0(tempdir(), "/sc/"
)
)
```

Arguments
foreground_sets  
a list of named character vectors of foreground sequences (only containing upper case characters A, C, G, T), where the names are RefSeq identifiers and sequence type qualifiers ("3UTR", "5UTR", "mRNA"), e.g. "NM_010356|3UTR". Names are only used to cache results.

background_set  
a named character vector of background sequences (naming follows same rules as foreground set sequences)

motifs  
a list of motifs that is used to score the specified sequences. If is.null(motifs) then all Transite motifs are used.

max_hits  
maximum number of putative binding sites per mRNA that are counted

threshold_method  
either "p_value" (default) or "relative". If threshold_method equals "p_value", the default threshold_value is 0.25^6, which is lowest p-value that can be achieved by hexamer motifs, the shortest supported motifs. If threshold_method equals "relative", the default threshold_value is 0.9, which is 90% of the maximum PWM score.

threshold_value  
semantics of the threshold_value depend on threshold_method (default is 0.25^6)
Details

Motif transcript set analysis can be used to identify RNA binding proteins, whose targets are significantly overrepresented or underrepresented in certain sets of transcripts.

The aim of Transcript Set Motif Analysis (TSMA) is to identify the overrepresentation and underrepresentation of potential RBP targets (binding sites) in a set (or sets) of sequences, i.e., the foreground set, relative to the entire population of sequences. The latter is called background set, which can be composed of all sequences of the genes of a microarray platform or all sequences of an organism or any other meaningful superset of the foreground sets.

The matrix-based approach skips the $k$-merization step of the $k$-mer-based approach and instead scores the transcript sequence as a whole with a position specific scoring matrix.

For each sequence in foreground and background sets and each sequence motif, the scoring algorithm evaluates the score for each sequence position. Positions with a relative score greater than a certain threshold are considered hits, i.e., putative binding sites.

By scoring all sequences in foreground and background sets, a hit count for each motif and each set is obtained, which is used to calculate enrichment values and associated p-values in the same way in which motif-compatible hexamer enrichment values are calculated in the $k$-mer-based approach. P-values are adjusted with one of the available adjustment methods.

An advantage of the matrix-based approach is the possibility of detecting clusters of binding sites. This can be done by counting regions with many hits using positional hit information or by simply applying a hit count threshold per sequence, e.g., only sequences with more than some number of hits are considered. Homotypic clusters of RBP binding sites may play a similar role as clusters of transcription factors.

Value

A list with the following components:
foreground_scores <- the result of `score_transcripts` for the foreground sets
background_scores <- the result of `score_transcripts` for the background set
enrichment_dfs <- a list of data frames, returned by `calculate_motif_enrichment`

See Also

Other TSMA functions: `draw_volcano_plot()`, `run_kmer_tsma()

Other matrix functions: `calculate_motif_enrichment()`, `run_matrix_spma()`, `score_transcripts_single_motif()`, `score_transcripts()`

Examples

```r
# define simple sequence sets for foreground and background
foreground_set1 <- c(
  "CAACAGCCUAAAU", "CAGUCAAGACUCC", "CUUUGGGAAU",
  "UCAUUUUAUAAA", "AAUUGGUGUCGGAUACUUCCCUGUACAU",
  "AUCAAUAU", "AGAU", "GACACUUAAGACCU",
  "UAGCAUUAACUUUUG", "AUGGA", "GAAGAGCUCA",
  "AUAGAC", "AGUUC", "CCAGUA"
)
names(foreground_set1) <- c(
  "NM_1_DUMMY|3UTR", "NM_2_DUMMY|3UTR", "NM_3_DUMMY|3UTR",
  "NM_4_DUMMY|3UTR", "NM_5_DUMMY|3UTR", "NM_6_DUMMY|3UTR",
  "NM_7_DUMMY|3UTR",
  "NM_8_DUMMY|3UTR", "NM_9_DUMMY|3UTR", "NM_10_DUMMY|3UTR",
  "NM_11_DUMMY|3UTR",
  "NM_12_DUMMY|3UTR", "NM_13_DUMMY|3UTR", "NM_14_DUMMY|3UTR"
)

foreground_set2 <- c("UUUUUA", "AUCCUUUACA", "UUUUUU", "UUUCAUCAUU")
names(foreground_set2) <- c(
  "NM_15_DUMMY|3UTR", "NM_16_DUMMY|3UTR", "NM_17_DUMMY|3UTR",
  "NM_18_DUMMY|3UTR"
)

foreground_sets <- list(foreground_set1, foreground_set2)

background_set <- c(
  "CAACAGCCUAAAU", "CAGUCAAGACUCC", "CUUUGGGAAU",
  "UCAUUUUAUAAA", "AAUUGGUGUCGGAUACUUCCCUGUACAU",
  "AUCAAUAU", "AGAU", "GACACUUAAGACCU",
  "UAGCAUUAACUUUUG", "AUGGA", "GAAGAGCUCA",
  "AUAGAC", "AGUUC", "CCAGUA",
  "UUUUUA", "AUCCUUUACA", "UUUUUU", "UUUCAUCAUU",
  "CCACACAC", "CUCAUUGGAG", "ACUUUUGGGACA", "CAGGUCGCA"
)
names(background_set) <- c(
  "NM_1_DUMMY|3UTR", "NM_2_DUMMY|3UTR", "NM_3_DUMMY|3UTR",
  "NM_4_DUMMY|3UTR", "NM_5_DUMMY|3UTR", "NM_6_DUMMY|3UTR",
  "NM_7_DUMMY|3UTR",
  "NM_8_DUMMY|3UTR", "NM_9_DUMMY|3UTR", "NM_10_DUMMY|3UTR",
  "NM_11_DUMMY|3UTR",
  "NM_12_DUMMY|3UTR", "NM_13_DUMMY|3UTR", "NM_14_DUMMY|3UTR"
)
"NM_12_DUMMY|3UTR", "NM_13_DUMMY|3UTR", "NM_14_DUMMY|3UTR",
"NM_15_DUMMY|3UTR",
"NM_16_DUMMY|3UTR", "NM_17_DUMMY|3UTR", "NM_18_DUMMY|3UTR",
"NM_19_DUMMY|3UTR",
"NM_20_DUMMY|3UTR", "NM_21_DUMMY|3UTR", "NM_22_DUMMY|3UTR"
)

# run cached version of TSMA with all Transite motifs (recommended):
# results <- run_matrix_tsma(foreground_sets, background_set)

# run uncached version with one motif:
# results <- run_matrix_tsma(foreground_sets, background_set, motifs = motif_db, cache = FALSE)

## Not run:
# define example sequence sets for foreground and background
foreground1_df <- transite:::ge$foreground1_df
foreground_set1 <- gsub("T", "U", foreground1_df$seq)
names(foreground_set1) <- paste0(foreground1_df$refseq, "|", foreground1_df$seq_type)

foreground2_df <- transite:::ge$foreground2_df
foreground_set2 <- gsub("T", "U", foreground2_df$seq)
names(foreground_set2) <- paste0(foreground2_df$refseq, "|", foreground2_df$seq_type)

foreground_sets <- list(foreground_set1, foreground_set2)

background_df <- transite:::ge$background_df
background_set <- gsub("T", "U", background_df$seq)
names(background_set) <- paste0(background_df$refseq, "|", background_df$seq_type)

# run cached version of TSMA with all Transite motifs (recommended)
results <- run_matrix_tsma(foreground_sets, background_set)

# run uncached version of TSMA with all Transite motifs
results <- run_matrix_tsma(foreground_sets, background_set, cache = FALSE)

# run TSMA with a subset of Transite motifs
results <- run_matrix_tsma(foreground_sets, background_set, motifs = get_motif_by_rbp("ELAVL1"))

# run TSMA with user-defined motif

## End(Not run)
score_sequences

Score Sequences with PWM

Description

C++ implementation of PWM scoring algorithm

Usage

score_sequences(sequences, pwm)

Arguments

- sequences: list of sequences
- pwm: position weight matrix

Value

list of PWM scores for each sequence

Examples

```r
motif <- get_motif_by_id("M178_0.6")[[1]]
sequences <- c("CAACAGCCUAAUU", "CAGUCAAGACUCC", "CUUUGGGAAAU",
               "UCAUUUGUAUG", "AAUUGGUGUCUGUAACUUCUCCUUGACAU",
               "AUCAAAUUUA", "UGUGGGG", "GACACUUAAAGAUAUCCU",
               "UAGCAUUAACUUAUG", "AUGGA", "GAAGAGUGCUCA", "AUAGAC",
               "AGUUU", "CCAGUAA")
seq_char_vectors <- lapply(sequences, function(seq) {
  unlist(strsplit(seq, ""))
})
score_sequences(seq_char_vectors, as.matrix(get_motif_matrix(motif)))
```

score_spectrum

Calculates spectrum scores and creates spectrum plots

Description

Spectrum scores are a means to evaluate if a spectrum has a meaningful (i.e., biologically relevant) or a random pattern.
score_spectrum

Usage

score_spectrum(
  x,
  p_values = array(1, length(x)),
  x_label = "log enrichment",
  sorted_transcript_values = NULL,
  transcript_values_label = "transcript value",
  midpoint = 0,
  x_value_limits = NULL,
  max_model_degree = 3,
  max_cs_permutations = 1e+07,
  min_cs_permutations = 5000,
  e = 5
)

Arguments

x vector of values (e.g., enrichment values, normalized RBP scores) per bin
p_values vector of p-values (e.g., significance of enrichment values) per bin
x_label label of values (e.g., "enrichment value")
sorted_transcript_values vector of sorted transcript values, i.e., the fold change or signal-to-noise ratio or any other quantity that was used to sort the transcripts that were passed to run_matrix_spma or run_kmer_spma (default value is NULL). These values are displayed as a semi-transparent area over the enrichment value heatmaps of spectrum plots.
transcript_values_label label of transcript sorting criterion (e.g., "log fold change", default value is "transcript value"), only shown if !is.null(sorted_transcript_values)
midpoint for enrichment values the midpoint should be 1, for log enrichment values 0 (defaults to 0)
x_value_limits sets limits of the x-value color scale (used to harmonize color scales of different spectrum plots), see limits argument of continuous_scale (defaults to NULL, i.e., the data-dependent default scale range)
max_model_degree maximum degree of polynomial
max_cs_permutations maximum number of permutations performed in Monte Carlo test for consistency score
min_cs_permutations minimum number of permutations performed in Monte Carlo test for consistency score
e integer-valued stop criterion for consistency score Monte Carlo test: aborting permutation process after observing e random consistency values with more extreme values than the actual consistency value
Details

One way to quantify the meaningfulness of a spectrum is to calculate the deviance between the linear interpolation of the scores of two adjoining bins and the score of the middle bin, for each position in the spectrum. The lower the score, the more consistent the trend in the spectrum plot. Formally, the local consistency score $x_c$ is defined as

$$x_c = \frac{1}{n} \sum_{i=1}^{n-2} \left| \frac{s_i + s_{i+2}}{2} - s_{i+1} \right|.$$  

In order to obtain an estimate of the significance of a particular score $x'_c$, Monte Carlo sampling is performed by randomly permuting the coordinates of the scores vector $s$ and recomputing $x_c$. The probability estimate $\hat{p}$ is given by the lower tail version of the cumulative distribution function

$$\hat{P}(T(x)) = \frac{\sum_{i=1}^{n} 1(T(y_i) \leq T(x)) + 1}{n + 1},$$

where 1 is the indicator function, $n$ is the sample size, i.e., the number of performed permutations, and $T$ equals $x_c$ in the above equation.

An alternative approach to assess the consistency of a spectrum plot is via polynomial regression. In a first step, polynomial regression models of various degrees are fitted to the data, i.e., the dependent variable $s$ (vector of scores), and orthogonal polynomials of the independent variable $b$ (vector of bin numbers). Secondly, the model that reflects best the true nature of the data is selected by means of the F-test. And lastly, the adjusted $R^2$ and the sum of squared residuals are calculated to indicate how well the model fits the data. These statistics are used as scores to rank the spectrum plots. In general, the polynomial regression equation is

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_m x_i^m + \epsilon_i,$$

where $m$ is the degree of the polynomial (usually $m \leq 5$), and $\epsilon_i$ is the error term. The dependent variable $y$ is the vector of scores $s$ and $x$ to $x^m$ are the orthogonal polynomials of the vector of bin numbers $b$. Orthogonal polynomials are used in order to reduce the correlation between the different powers of $b$ and therefore avoid multicollinearity in the model. This is important, because correlated predictors lead to unstable coefficients, i.e., the coefficients of a polynomial regression model of degree $m$ can be greatly different from a model of degree $m + 1$.

The orthogonal polynomials of vector $b$ are obtained by centering (subtracting the mean), QR decomposition, and subsequent normalization. Given the dependent variable $y$ and the orthogonal polynomials of $b$ to $x^m$, the model coefficients $\beta$ are chosen in a way to minimize the deviance between the actual and the predicted values characterized by

$$M(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m$$

$$M = \arg\min_M \left( \sum_{i=1}^{n} L(y_i, M(x_i)) \right),$$

where $L$ (actual value, predicted value) denotes the loss function.

Ordinary least squares is used as estimation method for the model coefficients $\beta$. The loss function of ordinary least squares is the sum of squared residuals (SSR) and is defined as follows

$$\text{SSR}(y, \hat{y}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where $y$ are the observed data and $\hat{y}$ the model predictions.
Thus the ordinary least squares estimate of the coefficients $\hat{\beta}$ (including the intercept $\hat{\beta}_0$) of the model $M$ is defined by

$$\hat{\beta} = \arg \min_\beta \left( \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^m \beta_j x_{ij})^2 \right).$$

After polynomial models of various degrees have been fitted to the data, the F-test is used to select the model that best fits the data. Since the SSR monotonically decreases with increasing model degree (model complexity), the relative decrease of the SSR between the simpler model and the more complex model must outweigh the increase in model complexity between the two models. The F-test gives the probability that a relative decrease of the SSR between the simpler and the more complex model given their respective degrees of freedom is due to chance. A low p-value indicates that the additional degrees of freedom of the more complex model lead to a better fit of the data than would be expected after a mere increase of degrees of freedom.

The F-statistic is calculated as follows

$$F = \frac{(SSR_1 - SSR_2)/(p_2 - p_1)}{SSR_2/(n - p_2)},$$

where $SSR_i$ is the sum of squared residuals and $p_i$ is the number of parameters of model $i$. The number of data points, i.e., bins, is denoted as $n$. $F$ is distributed according to the F-distribution with $df_1 = p_2 - p_1$ and $df_2 = n - p_2$.

Value

A list object of class SpectrumScore with the following components:

- `adj_r_squared`: adjusted $R^2$ of polynomial model
- `degree`: maximum degree of polynomial
- `residuals`: residuals of polynomial model
- `slope`: coefficient of the linear term of the polynomial model (spectrum "direction")
- `f_statistic`: F-test statistic
- `f_statistic_p_value`: p-value of F-test
- `consistency_score`: normalized sum of deviance between the linear interpolation of the scores of two adjoining bins and the score of the middle bin, for each position in the spectrum
- `consistency_score_p_value`: obtained by Monte Carlo sampling (randomly permuting the coordinates of the scores vector)
- `consistency_score_n`: number of permutations

See Also

Other SPMA functions: `classify_spectrum()`, `run_kmer_spma()`, `run_matrix_spma()`, `subdivide_data()`

Examples

```r
# random spectrum
score_spectrum(runif(n = 40, min = -1, max = 1), max_model_degree = 1)

# two random spectrums with harmonized color scales
plot(score_spectrum(runif(n = 40, min = -1, max = 1), max_model_degree = 1, x_value_limits = c(-2.0, 2.0)))
```
score_transcripts

Scores transcripts with position weight matrices

Description

This function is used to count the binding sites in a set of sequences for all or a subset of RNA-binding protein sequence motifs and returns the result in a data frame, which is subsequently used by calculate_motif_enrichment to obtain binding site enrichment scores.

Usage

```r
score_transcripts(
  sequences,
  motifs = NULL,
  max_hits = 5,
  threshold_method = c("p_value", "relative"),
  threshold_value = 0.25^6,
  n_cores = 1,
  cache = paste0(tempdir(), "/sc/"))
```
score_transcripts

Arguments

sequences character vector of named sequences (only containing upper case characters A, C, G, T), where the names are RefSeq identifiers and sequence type qualifiers ("3UTR", "5UTR", "mRNA"), e.g. "NM_010356|3UTR"

motifs a list of motifs that is used to score the specified sequences. If is.null(motifs) then all Transite motifs are used.

max_hits maximum number of putative binding sites per mRNA that are counted

threshold_method either "p_value" (default) or "relative". If threshold_method equals "p_value", the default threshold_value is $0.25^6$, which is lowest p-value that can be achieved by hexamer motifs, the shortest supported motifs. If threshold_method equals "relative", the default threshold_value is 0.9, which is 90% of the maximum PWM score.

threshold_value semantics of the threshold_value depend on threshold_method (default is $0.25^6$)

n_cores the number of cores that are used

cache either logical or path to a directory where scores are cached. The scores of each motif are stored in a separate file that contains a hash table with RefSeq identifiers and sequence type qualifiers as keys and the number of putative binding sites as values. If cache is FALSE, scores will not be cached.

Value

A list with three entries:

(1) df: a data frame with the following columns:

- motif_id: the motif identifier that is used in the original motif library
- motif_rbps: the gene symbol of the RNA-binding protein(s)
- absolute_hits: the absolute frequency of putative binding sites per motif in all transcripts
- relative_hits: the relative, i.e., absolute divided by total, frequency of binding sites per motif in all transcripts
- total_sites: the total number of potential binding sites
- one_hit, two_hits, ...: number of transcripts with one, two, three, ... putative binding sites

(2) total_sites: a numeric vector with the total number of potential binding sites per transcript

(3) absolute_hits: a numeric vector with the absolute (not relative) number of putative binding sites per transcript

See Also

Other matrix functions: calculate_motif_enrichment(), run_matrix_spma(), run_matrix_tsma(), score_transcripts_single_motif()
Examples

foreground_set <- c(
  "CAACAGCUUAUU", "CAGUCAAGACUCC", "CUUUGGGAAU",
  "UCAUUUUUAUAA", "AAUUGGUGCAGAUACUCCGCCUGACAU",
  "AUAAAUUA", "AGAU", "GACACUAAAGAUCCU",
  "UGAUAAACACUGAUG", "AUGGA", "GAAGAGUGCUCA",
  "AUAGAC", "AGUUC", "CCAGUAA"
)

# names are used as keys in the hash table (cached version only)
# ideally sequence identifiers (e.g., RefSeq ids) and region labels
# (e.g., 3' UTR)
names(foreground_set) <- c(
  "NM_1_DUMMY|3UTR", "NM_2_DUMMY|3UTR", "NM_3_DUMMY|3UTR",
  "NM_4_DUMMY|3UTR", "NM_5_DUMMY|3UTR", "NM_6_DUMMY|3UTR",
  "NM_7_DUMMY|3UTR", "NM_8_DUMMY|3UTR", "NM_9_DUMMY|3UTR",
  "NM_10_DUMMY|3UTR", "NM_11_DUMMY|3UTR", "NM_12_DUMMY|3UTR",
  "NM_13_DUMMY|3UTR", "NM_14_DUMMY|3UTR"
)

# specific motifs, uncached
motifs <- get_motif_by_rbp("ELAVL1")
scores <- score_transcripts(foreground_set, motifs = motifs, cache = FALSE)
## Not run:
# all Transite motifs, cached (writes scores to disk)
scores <- score_transcripts(foreground_set)

# all Transite motifs, uncached
scores <- score_transcripts(foreground_set, cache = FALSE)

foreground_df <- transite:::ge$foreground1_df
foreground_set <- foreground_df$seq
names(foreground_set) <- paste0(foreground_df$refseq, "|",
  foreground_df$seq_type)
scores <- score_transcripts(foreground_set)
## End(Not run)

score_transcripts_single_motif

Scores transcripts with position weight matrices

Description

This function is used to count the putative binding sites (i.e., motifs) in a set of sequences for the
specified RNA-binding protein sequence motifs and returns the result in a data frame, which is
aggregated by score_transcripts and subsequently used by calculate_motif_enrichment to
obtain binding site enrichment scores.
Usage

score_transcripts_single_motif(
  motif,
  sequences,
  max_hits = 5,
  threshold_method = c("p_value", "relative"),
  threshold_value = 0.25^6,
  cache_path = paste0(tempdir(), "/sc/")
)

Arguments

- **motif**: a Transite motif that is used to score the specified sequences
- **sequences**: character vector of named sequences (only containing upper case characters A, C, G, T), where the names are RefSeq identifiers and sequence type qualifiers ("3UTR", "5UTR", "mRNA"), e.g. "NM_010356|3UTR"
- **max_hits**: maximum number of putative binding sites per mRNA that are counted
- **threshold_method**: either "p_value" (default) or "relative". If threshold_method equals "p_value", the default threshold_value is 0.25^6, which is lowest p-value that can be achieved by hexamer motifs, the shortest supported motifs. If threshold_method equals "relative", the default threshold_value is 0.9, which is 90% of the maximum PWM score.
- **threshold_value**: semantics of the threshold_value depend on threshold_method (default is 0.25^6)
- **cache_path**: the path to a directory where scores are cached. The scores of each motif are stored in a separate file that contains a hash table with RefSeq identifiers and sequence type qualifiers as keys and the number of binding sites as values. If is.null(cache_path), scores will not be cached.

Value

A list with the following items:

- **motif_id**: the motif identifier of the specified motif
- **motif_rbps**: the gene symbol of the RNA-binding protein(s)
- **absolute_hits**: the absolute frequency of binding sites per motif in all transcripts
- **relative_hits**: the relative, i.e., absolute divided by total, frequency of binding sites per motif in all transcripts
- **total_sites**: the total number of potential binding sites
- **one_hit, two_hits, ...**: number of transcripts with one, two, three, ... binding sites

See Also

Other matrix functions: `calculate_motif_enrichment()`, `run_matrix_spma()`, `run_matrix_tsma()`, `score_transcripts()`
SpectrumScore-class

**set_motifs**

*Set Transite motif database*

**Description**

Globally sets Transite motif database, use with care.

**Usage**

`set_motifs(value)`

**Arguments**

- **value**
  - list of Motif objects

**Value**

void

**See Also**

Other motif functions: `generate_iupac_by_kmers()`, `generate_iupac_by_matrix()`, `generate_kmers_from_iupac()`, `get_motif_by_id()`, `get_motif_by_rbp()`, `get_motifs_meta_info()`, `get_motifs()`, `get_ppm()`, `init_iupac_lookup_table()`

**Examples**

```r
custom_motif <- create_kmer_motif(
  "custom_motif", "RBP1",
  c("AAAAAAA", "CAAAAAA"), "HITS-CLIP",
  "Homo sapiens", "user"
)
set_motifs(list(custom_motif))
```

---

**SpectrumScore-class**

*An S4 class to represent a scored spectrum*

**Description**

An S4 class to represent a scored spectrum

Getter Method get_adj_r_squared

Getter Method get_model_degree

Getter Method get_model_residuals

Getter Method get_model_slope

Getter Method get_model_f_statistic
**SpectrumScore-class**

Getter Method `get_model_f_statistic_p_value`
Getter Method `get_consistency_score`
Getter Method `get_consistency_score_p_value`
Getter Method `get_consistency_score_n`

**Usage**

```r
get_adj_r_squared(object)

## S4 method for signature 'SpectrumScore'
get_adj_r_squared(object)

get_model_degree(object)

## S4 method for signature 'SpectrumScore'
get_model_degree(object)

get_model_residuals(object)

## S4 method for signature 'SpectrumScore'
get_model_residuals(object)

get_model_slope(object)

## S4 method for signature 'SpectrumScore'
get_model_slope(object)

get_model_f_statistic(object)

## S4 method for signature 'SpectrumScore'
get_model_f_statistic(object)

get_model_f_statistic_p_value(object)

## S4 method for signature 'SpectrumScore'
get_model_f_statistic_p_value(object)

get_consistency_score(object)

## S4 method for signature 'SpectrumScore'
get_consistency_score(object)

get_consistency_score_p_value(object)

## S4 method for signature 'SpectrumScore'
get_consistency_score_p_value(object)

get_consistency_score_n(object)
```
## SpectrumScore-class

### S4 method for signature 'SpectrumScore'
get_consistency_score_n(object)

### S4 method for signature 'SpectrumScore'
show(object)

### S4 method for signature 'SpectrumScore,ANY'
plot(x)

#### Arguments
- object: SpectrumScore object
- x: SpectrumScore object

#### Value
Object of type SpectrumScore

#### Slots
- adj_r_squared: adjusted $R^2$ of polynomial model
- degree: degree of polynomial (integer between 0 and 5)
- residuals: residuals of the polynomial model
- slope: coefficient of the linear term of the polynomial model (spectrum "direction")
- f_statistic: F statistic from the F test used to determine the degree of the polynomial model
- f_statistic_p_value: p-value associated with the F statistic
- consistency_score: raw local consistency score of the spectrum
- consistency_score_p_value: p-value associated with the local consistency score
- consistency_score_n: number of permutations performed to calculate p-value of local consistency score (permutations performed before early stopping criterion reached)
- plot: spectrum plot

#### Examples
```
new("SpectrumScore",
    adj_r_squared = 0,
    degree = 0L,
    residuals = 0,
    slope = 0,
    f_statistic = 0,
    f_statistic_p_value = 1,
    consistency_score = 1,
    consistency_score_p_value = 1,
    consistency_score_n = 1000L,
    plot = NULL
)
```
subdivide_data  Subdivides Sequences into n Bins

Description

Preprocessing function for SPMA, divides transcript sequences into \( n \) bins.

Usage

\[
\text{subdivide\_data}(\text{sorted\_transcript\_sequences}, \text{n\_bins} = 40)
\]

Arguments

- \text{sorted\_transcript\_sequences}:
  character vector of named sequences (names are usually RefSeq identifiers and sequence region labels, e.g., "NM_1\_DUMMY\|3\_UTR"). It is important that the sequences are already sorted by fold change, signal-to-noise ratio or any other meaningful measure.

- \text{n\_bins}:
  specifies the number of bins in which the sequences will be divided, valid values are between 7 and 100

Value

An array of \( n\_bins \) length, containing the binned sequences

See Also

Other SPMA functions: \text{classify\_spectrum()}, \text{run\_kmer\_spma()}, \text{run\_matrix\_spma()}, \text{score\_spectrum()}

Examples

```r
# toy example
toy\_seqs \<- c(
  "CAACAGCCUA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\AA\A```
foreground_sets <- subdivide_data(toy_seqs, n_bins = 7)

# example data set
background_df <- transite::ge$background_df
# sort sequences by signal-to-noise ratio
background_df <- dplyr::arrange(background_df, value)
# character vector of named sequences
background_seqs <- background_df$seq
names(background_seqs) <- paste0(background_df$refseq, "|",
    background_df$seq_type)

foreground_sets <- subdivide_data(background_seqs)

---

**toy_motif_matrix**  
**Toy Motif Matrix**

**Description**

This toy motif matrix is used in code examples for various functions.

**Usage**

```r
data(toy_motif_matrix)
```

**Format**

A data frame with four columns (A, C, G, U) and seven rows (position 1 - 7)

---

**transite**  
**transite**

**Description**

transite is a computational method that allows comprehensive analysis of the regulatory role of RNA-binding proteins in various cellular processes by leveraging preexisting gene expression data and current knowledge of binding preferences of

**Author(s)**

Konstantin Krismer
Index

* -mer functions
  calculate_kmer_enrichment, 3
  check_kmers, 8
  compute_kmer_enrichment, 11
  count_homopolymer_corrected_kmers, 13
  draw_volcano_plot, 15
  estimate_significance, 17
  estimate_significance_core, 18
  generate_kmers, 22
  generate_permuted_enrichments, 24
  run_kmer_spma, 35
  run_kmer_tsma, 38

* SPMA functions
  classify_spectrum, 9
  run_kmer_spma, 35
  run_matrix_spma, 40
  score_spectrum, 48
  subdivide_data, 59

* TSMA functions
  draw_volcano_plot, 15
  run_kmer_tsma, 38
  run_matrix_tsma, 44

* datasets
  ge, 19
  kmer_enrichment, 30
  motifs, 30
  toy_motif_matrix, 60

* list(k)
  calculate_kmer_enrichment, 3
  check_kmers, 8
  compute_kmer_enrichment, 11
  count_homopolymer_corrected_kmers, 13
  draw_volcano_plot, 15
  estimate_significance, 17
  estimate_significance_core, 18
  generate_kmers, 22
  generate_permuted_enrichments, 24
  run_kmer_spma, 35
  run_kmer_tsma, 38

* matrix functions
  calculate_motif_enrichment, 5
  run_matrix_spma, 40
  run_matrix_tsma, 44
  score_transcripts, 52
  score_transcripts_single_motif, 54

* motif functions
  generate_iupac_by_kmers, 19
  generate_iupac_by_matrix, 20
  generate_kmers_from_iupac, 23
  get_motif_by_id, 27
  get_motif_by_rbp, 27
  get_motifs, 56
  get_motifs_meta_info, 26
  get_ppm, 28
  init_iupac_lookup_table, 29
  set_motifs, 56
  .RBPMotif (RBPMotif-class), 33
  .SpectrumScore (SpectrumScore-class), 56

  calculate_kmer_enrichment, 3, 9, 12, 13, 16–18, 22, 25, 37, 39
  calculate_local_consistency, 4
  calculate_motif_enrichment, 5, 43, 46, 52–55
  calculate_transcript_mc, 7
  check_kmers, 4, 8, 12, 13, 16–18, 22, 25, 37, 39
  classify_spectrum, 9, 37, 43, 51, 59
  compute_kmer_enrichment, 3, 4, 9, 11, 13, 16–18, 22, 25, 37, 39
  continuous_scale, 36, 41, 49
  count_homopolymer_corrected_kmers, 4, 9, 12, 13, 16–18, 22, 25, 37, 39
  create_kmer_motif, 14
  create_matrix_motif, 14
  draw_volcano_plot, 4, 9, 12, 13, 15, 17, 18,
estimación de significancia, 4, 9, 12, 13, 16, 17, 18, 22, 25, 37, 39
estimación de significancia, 4, 9, 12, 13, 16, 17, 18, 22, 25, 37, 39
ge, 19
generate_iupac_by_kmers, 19, 21, 24, 26–29, 56
generate_iupac_by_matrix, 20, 20, 24, 26–29, 35, 56
generate_kmers, 4, 9, 11–13, 16–18, 22, 25, 37, 39
generate_kmers_from_iupac, 20, 21, 23, 26–29, 56
generate_permuted_enrichments, 4, 9, 12, 13, 16–18, 22, 24, 37, 39
génetico mediano, 25
get_adj_r_squared (SpectrumScore-class), 56
get_adj_r_squared, SpectrumScore-method (SpectrumScore-class), 56
get_consistency_score (SpectrumScore-class), 56
get_consistency_score, SpectrumScore-method (SpectrumScore-class), 56
get_consistency_score_n (SpectrumScore-class), 56
get_consistency_score_n, SpectrumScore-method (SpectrumScore-class), 56
get_consistency_score_p_value (SpectrumScore-class), 56
get_consistency_score_p_value, SpectrumScore-method (SpectrumScore-class), 56
get_heptamers (RBPMotif-class), 33
get_heptamers, RBPMotif-method (RBPMotif-class), 33
generate_kmers (RBPMotif-class), 33
generate_kmers, RBPMotif-method (RBPMotif-class), 33
generate_kmers, RBPMotif-method (RBPMotif-class), 33
get_id (RBPMotif-class), 33
generate_iupac (RBPMotif-class), 33
generate_iupac, RBPMotif-method (RBPMotif-class), 33
get_model_degree (SpectrumScore-class), 56
get_model_degree, SpectrumScore-method (SpectrumScore-class), 56
get_model_f_statistic (SpectrumScore-class), 56
get_model_f_statistic, SpectrumScore-method (SpectrumScore-class), 56
get_model_f_statistic_p_value (SpectrumScore-class), 56
get_model_f_statistic_p_value, SpectrumScore-method (SpectrumScore-class), 56
get_model_residuals (SpectrumScore-class), 56
get_model_residuals, SpectrumScore-method (SpectrumScore-class), 56
get_model_slope (SpectrumScore-class), 56
get_model_slope, SpectrumScore-method (SpectrumScore-class), 56
get_motif_by_id, 20, 21, 24, 26, 27, 28, 29, 56
get_motif_by_id, rbp, 20, 21, 24, 26, 27, 28, 29, 56
get_motif_matrix (RBPMotif-class), 33
get_motif_matrix, RBPMotif-method (RBPMotif-class), 33
get_motifs, 20, 21, 24, 25, 26–29, 56
get_motifs_meta_info, 20, 21, 24, 26, 27, 29, 56
get_ppm, 20, 21, 24, 26–28, 28, 29, 56
generate_rbps (RBPMotif-class), 33
generate_rbps, RBPMotif-method (RBPMotif-class), 33
get_species (RBPMotif-class), 33
generate_species, RBPMotif-method (RBPMotif-class), 33
generate_type (RBPMotif-class), 33
generate_type, RBPMotif-method (RBPMotif-class), 33
get_width (RBPMotif-class), 33
generate_width, RBPMotif-method (RBPMotif-class), 33
init_iupac_lookup_table, 19–21, 24, 26–28, 29, 56
kmers_enrichment, 30
INDEX

motifs, 30
p.adjust, 3, 6, 12, 36, 38, 42, 45
p_combine, 31, 36, 38
plot, RBPMotif, ANY-method
  (RBPMotif-class), 33
plot, RBPMotif-method (RBPMotif-class), 33
plot, SpectrumScore, ANY-method
  (SpectrumScore-class), 56
plot, SpectrumScore-method
  (SpectrumScore-class), 56

RBPMotif-class, 33
run_kmer_spma, 4, 9, 10, 12, 13, 16–18, 22, 25, 35, 39, 43, 51, 59
run_kmer_tsma, 4, 9, 12, 13, 16–18, 22, 25, 30, 37, 38, 46
run_matrix_spma, 6, 10, 37, 40, 46, 51, 53, 55, 59
run_matrix_tsma, 6, 16, 39, 43, 44, 53, 55

score_sequences, 48
score_spectrum, 9, 10, 37, 43, 48, 59
score_transcripts, 6, 7, 43, 46, 52, 54, 55
score_transcripts_single_motif, 6, 43, 46, 53, 54
set_motifs, 20, 21, 24, 26–29, 56
show, RBPMotif-method (RBPMotif-class), 33
show, SpectrumScore-method
  (SpectrumScore-class), 56
SpectrumScore-class, 56
subdivide_data, 10, 37, 43, 51, 59

toy_motif_matrix, 60
transite, 60