Package ‘sigminer’

June 17, 2020

Title Extract, Analyze and Visualize Signatures for Genomic Variations

Version 1.0.7

Date 2020-06-01

Description Genomic alterations including single nucleotide substitution, copy number alteration, etc. are the major force for cancer initialization and development. Due to the specificity of molecular lesions caused by genomic alterations, we can generate characteristic alteration spectra, called 'signature' (Wang, Shixiang, et al. (2020) <DOI:10.1101/2020.04.27.20082404> & Alexandrov, Ludmil B., et al. (2020) <DOI:10.1038/s41586-020-1943-3> & Macintyre, Geoff, et al. (2018) <DOI:10.1038/s41588-018-0179-8>). This package helps users to extract, analyze and visualize signatures from genomic alteration records, thus providing new insight into cancer study.

License MIT + file LICENSE

URL https://github.com/ShixiangWang/sigminer

BugReports https://github.com/ShixiangWang/sigminer/issues

Depends R (>= 3.5)

Imports cli (>= 2.0.0), cowplot, data.table, dplyr, foreach, furrr, future, ggplot2 (>= 3.3.0), ggpubr, maftools, magrittr, methods, NMF, purrr, rlang (>= 0.1.2), stats, tidyr

Suggests Biobase, Biostrings, BSgenome, BSgenome.Hsapiens.UCSC.hg19, circlize, covr, doFuture, flexmix, GenomicRanges, GenSA, ggalluvial, ggfittext, ggplotify, IRanges, knitr, patchwork, pheatmap, pracma, quadprog, R.utils, RColorBrewer, rmarkdown, roxygen2, testthat, tibble

VignetteBuilder knitr

biocViews

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0
NeedsCompilation no

Author Shixiang Wang [aut, cre] (<https://orcid.org/0000-0001-9855-7357>),
Xue-Song Liu [aut, ctb] (<https://orcid.org/0000-0002-7736-0077>),
Tao Wu [aut, ctb],
Ziyu Tao [aut, ctb],
Geoffrey Macintyre [ctb],
Anand Mayakonda [ctb]

Maintainer Shixiang Wang <w_shixiang@163.com>

Repository CRAN

Date/Publication 2020-06-17 05:20:02 UTC

R topics documented:

- add_h_arrow ................................. 3
- add_labels .................................. 4
- centromeres.hg19 .......................... 6
- centromeres.hg38 .......................... 6
- chromsize.hg19 ............................ 7
- chromsize.hg38 ............................ 7
- CN_features ................................ 8
- CopyNumber-class ......................... 8
- cytobands.hg19 ............................ 9
- cytobands.hg38 ............................ 9
- enrich_component_strand_bias ............. 10
- get_adj_p .................................. 10
- get_bayesian_result ....................... 12
- get_cn_ploidy ............................. 13
- get_genome_annotation .................... 14
- get_groups ................................ 15
- get_group_comparison ..................... 16
- get_sig_exposure ......................... 18
- get_sig_feature_association ............. 19
- get_sig_similarity ....................... 20
- get_tidy_association ..................... 22
- get_tidy_parameter ....................... 22
- handle_hyper_mutation .................... 23
- hello ..................................... 24
- MAF-class .................................. 24
- read_copynumber ......................... 25
- read_maf .................................. 26
- report_bootstrap_p_value ................ 27
- scoring ................................... 28
- show_catalogue ............................ 30
- show_cn_circos ........................... 31
- show_cn_components ....................... 32
- show_cn_distribution ..................... 34
Add Horizontal Arrow with Text Label to a ggplot

Description
Add Horizontal Arrow with Text Label to a ggplot

Usage
```r
add_h_arrow(p, x, y, label = "optimal number", space = 0.01, vjust = 0.3, seg_len = 0.1,
```
add_labels

arrow_len = unit(2, "mm"),
arrow_type = c("closed", "open"),
font_size = 5,
font_family = c("serif", "sans", "mono"),
font_face = c("plain", "bold", "italic")
)

Arguments

p a ggplot.

x position at x axis.

y position at y axis.

label text label.

space a small space between arrow and text.

vjust vertical adjustment, set to 0 to align with the bottom, 0.5 for the middle, and 1 (the default) for the top.

seg_len length of the arrow segment.

arrow_len length of the arrow.

arrow_type type of the arrow.

font_size font size.

font_family font family.

font_face font face.

Value

a ggplot object.

add_labels(p, x, y, y_end = NULL, n_label = NULL, labels = NULL, font_size = 5,

Description

Add text labels to a ggplot object, such as the result from show_sig_profile.

Usage

add_labels(p, x, y, y_end = NULL, n_label = NULL, labels = NULL, font_size = 5,
font_family = "serif",
font_face = c("plain", "bold", "italic"),
...)

Arguments

p a ggplot.
x position at x axis.
y position at y axis.
y_end end position of y axis when n_label is set.
n_label the number of label, when this is set, the position of labels at y axis is auto-generated according to y and y_end.
labels text labels or a similarity object from get_sig_similarity.
font_size font size.
font_family font family.
font_face font face.
... other parameters passing to ggplot2::annotate.

Value

a ggplot object.

Examples

# Load mutational signature
load(system.file("extdata", "toy_mutational_signature.RData",
    package = "sigminer", mustWork = TRUE))
# Show signature profile
p <- show_sig_profile(sig2, mode = "SBS")

# Method 1
p1 <- add_labels(p,
    x = 0.75, y = 0.3, y_end = 0.9, n_label = 3,
    labels = paste0("text", 1:3))
p1

# Method 2
p2 <- add_labels(p,
    x = c(0.15, 0.6, 0.75), y = c(0.3, 0.6, 0.9),
    labels = paste0("text", 1:3))
p2

# Method 3
sim <- get_sig_similarity(sig2)
centromeres.hg19  
*Location of Centromeres at Genome Build hg19*

**Description**

Location of Centromeres at Genome Build hg19

**Format**

A data.frame

**Source**

Generate from UCSC gold path

**Examples**

data(centromeres.hg19)

---

centromeres.hg38  
*Location of Centromeres at Genome Build hg38*

**Description**

Location of Centromeres at Genome Build hg38

**Format**

A data.frame

**Source**

Generate from Genome Reference Consortium

**Examples**

data(centromeres.hg38)
<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
<th>Format</th>
<th>Source</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromsize.hg19</td>
<td>Chromosome Size of Genome Build hg19</td>
<td>A data.frame</td>
<td>Generate from UCSC gold path</td>
<td>data(chromsize.hg19)</td>
</tr>
<tr>
<td>chromsize.hg38</td>
<td>Chromosome Size of Genome Build hg38</td>
<td>A data.frame</td>
<td>Generate from UCSC gold path</td>
<td>data(chromsize.hg38)</td>
</tr>
</tbody>
</table>
**CopyNumber-class**

Classification Table of Copy Number Features Devised by Wang et al.

**Description**

A `data.table` with "sigminer.features" class name

**Format**

Generate from code under data_raw/

**Examples**

```r
data(CN.features)
```

---

**CopyNumber-class**

Class CopyNumber

**Description**

S4 class for storing summarized absolute copy number profile.

**Slots**

- `data` data.table of absolute copy number calling.
- `summary.per.sample` data.table of copy number variation summary per sample.
- `genome_build` genome build version, should be one of 'hg19' or 'hg38'.
- `genome.measure` Set 'called' will use autosomo called segments size to compute total size for CNA burden calculation, this option is useful for WES and target sequencing. Set 'wg' will autosome size from genome build, this option is useful for WGS, SNP etc.
- `annotation` data.table of annotation for copy number segments.
- `dropoff.segs` data.table of copy number segments dropped from raw input.
cytobands.hg19  

*Location of Chromosome Cytobands at Genome Build hg19*

**Description**

Location of Chromosome Cytobands at Genome Build hg19

**Format**

A data.frame

**Source**

from UCSC

**Examples**

```r
data(cytobands.hg19)
```

---

cytobands.hg38  

*Location of Chromosome Cytobands at Genome Build hg38*

**Description**

Location of Chromosome Cytobands at Genome Build hg38

**Format**

A data.frame

**Source**

from UCSC

**Examples**

```r
data(cytobands.hg38)
```
enrich_component_strand_bias

Performs Strand Bias Enrichment Analysis for a Given Sample-by-Component Matrix

Description

See `sig_tally` for examples.

Usage

enrich_component_strand_bias(mat)

Arguments

mat an object of class `data.frame` containing the sample-by-component matrix from the `sig_tally` function with strand bias labels "T:" and "B:"

Value

A `data.table` object sorted by `p_value`.

get_adj_p

Get Adjust P Values from Group Comparison

Description

Setting `aes(label=..p.adj..)` in `ggpubr::compare_means()` does not show adjust p values. The returned result of this function can be combined with `ggpubr::stat_pvalue_manual()` to fix this problem.

Usage

get_adj_p(
  data,
  .col,
  .grp = "Sample",
  comparisons = NULL,
  method = "wilcox.test",
  p.adjust.method = "fdr",
  p.digits = 3L,
  ...
)
get_adj_p

Arguments

data a data.frame containing column for groups and column for comparison.
.col column name for comparison.
.grp column name for groups.
.comparisons Default is NULL, use all combination in group column. It can be a list of length-2 vectors. The entries in the vector are either the names of 2 values on the x-axis or the 2 integers that correspond to the index of the groups of interest, to be compared.
.method a character string indicating which method to be used for comparing means. It can be 't.test', 'wilcox.test' etc..
.p.adjust.method correction method, default is 'fdr'. Run p.adjust.methods to see all available options.
.p.digits how many significant digits are to be used.
... other arguments passed to ggpubr::compare_means()

details

More info see ggpubr::compare_means(), ggpubr::stat_compare_means() and stats::p.adjust().

Value

a data.frame containing comparison result

Source

https://github.com/kassambara/ggpubr/issues/143

Examples

library(ggpubr)
# T-test
stat.test <- compare_means(
  len ~ dose,
  data = ToothGrowth,
  method = "t.test",
  p.adjust.method = "fdr"
)
stat.test
# Create a simple box plot
p <- ggboxplot(ToothGrowth, x = "dose", y = "len")
p

# Add p values
my_comparisons <- list(c("0.5", "1"), c("1", "2"), c("0.5", "2"))
p + stat_compare_means(method = "t.test", comparisons = my_comparisons)

# Try adding adjust p values
get_bayesian_result

get_bayesian_result  Get Specified Bayesian NMF Result from Run

Description

Sometimes, we may want to use or inspect specified run result from sig_auto_extract. This function is designed for this purpose.

Usage

get_bayesian_result(run_info)

Arguments

run_info  a data.frame with 1 row and two necessary columns Run and file.

Value

a list.

Author(s)

Shixiang Wang
get_cn_ploidy

Examples

```r
load(system.file("extdata", "toy_copynumber_tally_W.RData",
    package = "sigminer", mustWork = TRUE
))

res <- sig_auto_extract(cn_tally_W$nmf_matrix, result_prefix = "Test_copynumber", nrun = 1)

# All run info are stored in res$Raw$summary_run
# Obtain result of run 1
res_run1 <- get_bayesian_result(res$Raw$summary_run[1, ])
```

**Description**

Get Ploidy from Absolute Copy Number Profile

**Usage**

```r
get_cn_ploidy(data)
```

**Arguments**

- `data` a `CopyNumber` object or a `data.frame` containing at least 'chromosome', 'start', 'end', 'segVal' these columns.

**Value**

a value or a `data.table`

**Examples**

```r
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData",
    package = "sigminer", mustWork = TRUE
))

df <- get_cn_ploidy(cn)
df
```
get_genome_annotation  Get Genome Annotation

Description

Get Genome Annotation

Usage

get_genome_annotation(
  data_type = c("chr_size", "centro_loc", "cytobands"),
  chrs = paste0("chr", c(1:22, "X", "Y")),
  genome_build = c("hg19", "hg38")
)

Arguments

data_type  'chr_size' for chromosome size, 'centro_loc' for location of centromeres and 'cytobands' for location of chromosome cytobands.

chrs  chromosomes start with 'chr'

genome_build  one of 'hg19', 'hg38'

Value

a data.frame containing annotation data

Examples

df1 <- get_genome_annotation()
df1

df2 <- get_genome_annotation(genome_build = "hg38")
df2

df3 <- get_genome_annotation(data_type = "centro_loc")
df3

df4 <- get_genome_annotation(data_type = "centro_loc", genome_build = "hg38")
df4

df5 <- get_genome_annotation(data_type = "cytobands")
df5

df6 <- get_genome_annotation(data_type = "cytobands", genome_build = "hg38")
df6
get_groups

Get Sample Groups from Signature Decomposition Information

Description

One of key results from signature analysis is to cluster samples into different groups. This function takes Signature object as input and return the membership in each cluster.

Usage

```
get_groups(
  Signature,  # a Signature object obtained either from sig_extract or sig_auto_extract. Now it can be used to relative exposure result in data.table format from sig_fit.
  method = c("consensus", "k-means", "exposure", "samples"),  # grouping method, more see details, could be one of the following:
  n_cluster = NULL,  # only used when the method is 'k-means'.
  match_consensus = TRUE  # only used when the method is 'consensus'. If TRUE, the result will match order as shown in consensus map.
)
```

Arguments

- **Signature**: a Signature object obtained either from sig_extract or sig_auto_extract. Now it can be used to relative exposure result in data.table format from sig_fit.
- **method**: grouping method, more see details, could be one of the following:
  - 'consensus' - returns the cluster membership based on the hierarchical clustering of the consensus matrix, it can only be used for the result obtained by sig_extract() with multiple runs using NMF package.
  - 'k-means' - returns the clusters by k-means.
  - 'exposure' - assigns a sample into a group whose signature exposure is dominant.
  - 'samples' - returns the cluster membership based on the contribution of signature to each sample, it can only be used for the result obtained by sig_extract() using NMF package.
- **n_cluster**: only used when the method is 'k-means'.
- **match_consensus**: only used when the method is 'consensus'. If TRUE, the result will match order as shown in consensus map.

Details

Users may find there are bigger differences between using method 'samples' and 'exposure' but they use a similar idear to find dominant signature, here goes the reason:

Method 'samples' using data directly from NMF decomposition, this means the two matrix $W$ (basis matrix or signature matrix) and $H$ (coefficient matrix or exposure matrix) are the results of NMF. For method 'exposure', it uses the signature exposure loading matrix. In this situation, each signature represents a number of mutations (alterations) about implementation please see source code of sig_extract() function.
get_group_comparison

Value

a data.table object

See Also

NMF::predict(), show_groups.

Examples

```r
# Load copy number prepare object
load(system.file("extdata", "toy_copynumber_tally_W.RData", 
    package = "sigminer", mustWork = TRUE
))
# Extract copy number signatures
library(NMF)
sig <- sig_extract(cn_tally_W$nmf_matrix, 2, 
    nrun = 10, 
    pConstant = 1e-13
)
# Methods 'consensus' and 'samples' are from NMF::predict()
get_groups(sig, method = "consensus", match_consensus = TRUE)
get_groups(sig, method = "samples")
# Use k-means clustering
get_groups(sig, method = "k-means")
```

group_comparison  Get Comparison Result between Signature Groups

Description

Compare genotypes/phenotypes based on signature groups (samples are assigned to several groups). For categorical type, calculate fisher p value (using stats::fisher.test) and count table. In larger than 2 by 2 tables, compute p-values by Monte Carlo simulation. For continuous type, calculate anova p value (using stats::aov), summary table and Tukey Honest significant difference (using stats::TukeyHSD). The result of this function can be plotted by show_group_comparison().

Usage

```r
get_group_comparison(
    data, 
    col_group, 
    cols_to_compare, 
    type = "ca", 
    NAs = NA, 
    verbose = FALSE
)
```
get_group_comparison

Arguments

data a data.frame containing signature groups and genotypes/phenotypes (including categorical and continuous type data) want to analyze. User need to construct this data.frame by him/herself.
col_group column name of signature groups.
cols_to_compare column names of genotypes/phenotypes want to summarize based on groups.
type a character vector with length same as cols_to_compare, 'ca' for categorical type and 'co' for continuous type.
NAs default is NA, filter NAs for categorical columns. Otherwise a value (either length 1 or length same as cols_to_compare) fill NAs.
verbose if TRUE, print extra information.

Value
a list contains data, summary, p value etc..

Author(s)
Shixiang Wang w_shixiang@163.com

Examples

load(system.file("extdata", "toy_copynumber_signature_by_M.RData",  
  package = "sigminer", mustWork = TRUE  
))

# Assign samples to clusters
groups <- get_groups(sig, method = "k-means")
set.seed(1234)

groups$prob <- rnorm(10)
groups$new_group <- sample(c("1", "2", "3", "4", NA), size = nrow(groups), replace = TRUE)

# Compare groups (filter NAs for categorical columns)
groups.cmp <- get_group_comparison(groups[, -1],  
  col_group = "group",  
  cols_to_compare = c("prob", "new_group"),  
  type = c("co", "ca"), verbose = TRUE  
)

# Compare groups (Set NAs of categorical columns to 'Rest')
groups.cmp2 <- get_group_comparison(groups[, -1],  
  col_group = "group",  
  cols_to_compare = c("prob", "new_group"),  
  type = c("co", "ca"), NAs = "Rest", verbose = TRUE  
)
get_sig_exposure  

Get Signature Exposure from 'Signature' Object

Description

The expected number of mutations (or copy number segment records) with each signature was determined after a scaling transformation $V \sim WH = W'H'$ where $W' = WU'$ and $H' = UH$. The scaling matrix $U$ is a $K \times K$ diagonal matrix ($K$ is signature number, $U'$ is the inverse of $U$) with the element corresponding to the L1-norm of column vectors of $W$ (i.e., the sum of the elements of the vector). As a result, the $k$-th row vector of the final matrix $H'$ represents the absolute exposure (activity) of the $k$-th process across samples (e.g., for SBS, the estimated (or expected) number of mutations generated by the $k$-th process). Of note, for copy number signatures, only components of feature CN was used for calculating $H'$.

Usage

```r
get_sig_exposure(
    Signature,
    type = c("absolute", "relative"),
    rel_threshold = 0.01
)
```

Arguments

- **Signature**: a Signature object obtained either from `sig_extract` or `sig_auto_extract`, or just a raw exposure matrix with column representing samples (patients) and row representing signatures.
- **type**: 'absolute' for signature exposure and 'relative' for signature relative exposure.
- **rel_threshold**: used when type is 'relative', relative exposure less than this value will be set to 0 and the remaining signature exposure will be scaled to make sum as 1 accordingly. Of note, this is a little different from the same parameter in `sig_fit`.

Value

a data.table

Author(s)

Shixiang Wang w_shixiang@163.com

References

get_sig_feature_association

Examples

# Load mutational signature
load(system.file("extdata", "toy_mutational_signature.RData",  
    package = "sigminer", mustWork = TRUE
  ))
# Get signature exposure
expo1 <- get_sig_exposure(sig2)
expo1
expo2 <- get_sig_exposure(sig2, type = "relative")
expo2

get_sig_feature_association

Calculate Association between Signature Exposures and Other Features

Description

Association of signature exposures with other features will be performed using one of two procedures: for a continuous association variable (including ordinal variable), correlation is performed; for a binary association variable, samples will be divided into two groups and Mann-Whitney U-test is performed to test for differences in signature exposure medians between the two groups. See get_tidy_association for cleaning association result.

Usage

get_sig_feature_association(
  data,
  cols_to_sigs,
  cols_to_features,
  type = "ca",
  method_co = c("spearman", "pearson", "kendall"),
  method_ca = stats::wilcox.test,
  min_n = 0.01,
  verbose = FALSE,
  ...
)

Arguments

data a data.frame contains signature exposures and other features
cols_to_sigs colnames for signature exposure
cols_to_features colnames for other features
type a character vector containing 'ca' for categorical variable and 'co' for continuous variable, it must have the same length as cols_to_features.
get_sig_similarity

method_co method for continuous variable, default is "spearman", could also be "pearson" and "kendall".
method_ca method for categorical variable, default is "wilcox.test"
min_n a minimal fraction (e.g. 0.01) or a integer number (e.g. 10) for filtering some variables with few positive events. Default is 0.01.
verbose if TRUE, print extra message.
... other arguments passing to test functions, like cor.test.

Value

a list. For ‘co’ features, ‘measure’ means correlation coefficient. For ‘ca’ features, ‘measure’ means difference in means of signature exposure.

References


See Also

get_tidy_association

get_sig_similarity  Calculate Similarity between Identified Signatures and Reference Signatures

Description

The reference signatures can be either a Signature object specified by Ref argument or known COSMIC signatures specified by sig_db argument. Two COSMIC databases are used for comparisons - "legacy" which includes 30 signatures, and "SBS" - which includes updated/refined 65 signatures. This function is modified from compareSignatures() in maftools package.

Usage

get_sig_similarity(
  Signature,
  Ref = NULL,
  sig_db = c("legacy", "human-exome", "human-genome"),
  db_type = c("", "human-exome", "human-genome"),
  method = "cosine",
  normalize = c("row", "feature"),
  feature_setting = sigminer::CN.features,
  pattern_to_rm = NULL,
  verbose = TRUE
)
get_sig_similarity

Arguments

Signature a Signature object or a component-by-signature matrix (sum of each column is 1). More please see examples.

Ref default is NULL, can be a same object as Signature.

sig_db can be 'legacy' (for COSMIC v2 'SBS'), 'SBS', 'DBS', 'ID' and 'TSB' (for SBS transcriptional strand bias signatures). Default 'legacy'.

db_type only used when sig_db is enabled. "" for keeping default, "human-exome" for transforming to exome frequency of component, and "human-genome" for transforming to whole genome frequency of component. Currently only works for 'SBS'.

method default is 'cosine' for cosine similarity.

normalize one of "row" and "feature". "row" is typically used for common mutational signatures. "feature" is designed by me to use when input are copy number signatures.

feature_setting a data.frame used for classification. Only used when method is "Wang" ("W"). Default is CN.features. Users can also set custom input with "feature", "min" and "max" columns available. Valid features can be printed by unique(CN.features$feature).

pattern_to_rm patterns for removing some features/components in similarity calculation. A vector of component name is also accepted. The remove operation will be done after normalization. Default is NULL.

verbose if TRUE, print extra info.

Value

a list containing similarities, aetiologies if available, and best match.

Author(s)

Shixiang Wang w_shixiang@163.com

Examples

# Load mutational signature
load(system.file("extdata", "toy_mutational_signature.RData", package = "sigminer", mustWork = TRUE))

s1 <- get_sig_similarity(sig2, Ref = sig2)
s1

s2 <- get_sig_similarity(sig2)
s2

s3 <- get_sig_similarity(sig2, sig_db = "SBS")
s3
## Remove some components
## in similarity calculation
s4 <- get_sig_similarity(sig2,
Ref = sig2,
)
s4

## Same to DBS and ID signatures

---

### get_tidy_association

**Get Tidy Signature Association Results**

#### Description

Get Tidy Signature Association Results

#### Usage

```r
get_tidy_association(cor_res, p_adjust = FALSE, method = "fdr")
```

#### Arguments

- `cor_res`: data returned by `get_sig_feature_association`
- `p_adjust`: logical, if `TRUE`, adjust p values by data type.
- `method`: p value correction method, see `stats::p.adjust` for more detail.

#### Value

a data.frame

#### See Also

`get_sig_feature_association`

---

### get_tidy_parameter

**Get Tidy Parameter from Flexmix Model**

#### Description

When users derive copy number features, it is useful to know the parameters of the fit components, including mean, sd and coefficient of variation. This function is used by `sig_tally` function and exported to users for extra usage.

#### Usage

```r
get_tidy_parameter(x)
```
Arguments

- `x`: a `flexmix` object or a list of `flexmix` objects.

Value

- A tibble.

Examples

```r
load(system.file("extdata", "toy_copynumber_tally_M.RData",  
 package = "sigminer", mustWork = TRUE  
))
# Get all parameters
d1 <- get_tidy_parameter(cn_tally_M$components)
d1
# Get parameters for segsize feature
d2 <- get_tidy_parameter(cn_tally_M$components$segsize)
d2
```

---

**handle_hyper_mutation**  
**Handle Hypermutant Samples**

Description

This can be used for SNV/INDEL count matrix. For copy number analysis, please skip it.

Usage

```r
handle_hyper_mutation(nmf_matrix)
```

Arguments

- `nmf_matrix`: a matrix used for NMF decomposition with rows indicate samples and columns indicate components.

Value

- A matrix.

References

Say Hello to Users

hello()
read_copynumber

Read Absolute Copy Number Profile

Description

Read **absolute** copy number profile for preparing CNV signature analysis.

Usage

```r
read_copynumber(
  input,
  pattern = NULL,
  ignore_case = FALSE,
  seg_cols = c("Chromosome", "Start.bp", "End.bp", "modal_cn"),
  samp_col = "sample",
  join_adj_seg = TRUE,
  use_all = FALSE,
  min_segnum = 0L,
  max_copynumber = 20L,
  genome_build = c("hg19", "hg38"),
  genome_measure = c("called", "wg"),
  complement = TRUE,
  ...
)
```

Arguments

- **input**: a data.frame or a file or a directory contains copy number profile.
- **pattern**: an optional regular expression used to select part of files if `input` is a directory, more detail please see `list.files` function.
- **ignore_case**: logical. Should pattern-matching be case-insensitive?
- **seg_cols**: four characters used to specify chromosome, start position, end position and copy number value in `input`, respectively. Default use names from ABSOLUTE calling result.
- **samp_col**: a character used to specify the sample column name. If `input` is a directory and cannot find `samp_col`, sample names will use file names (set this parameter to `NULL` is recommended in this case).
- **join_adj_seg**: if `TRUE` (default), join adjacent segments with same copy number value. This is helpful for precisely count the number of breakpoint. When set `use_all=TRUE`, the mean function will be applied to extra numeric columns and unique string columns will be pasted by comma for joined records.
- **use_all**: default is `FALSE`. If `TRUE`, use all columns from raw input.
- **min_segnum**: minimal number of copy number segments within a sample.
- **max_copynumber**: bigger copy number within a sample will be reset to this value.
Read MAF Files

Description

This function is a wrapper of maftools::read.maf. Useless options in maftools::read.maf are dropped here. You can also use maftools::read.maf to read the data.

Value

A CopyNumber object.

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

read_maf for reading mutation data to MAF object.

Examples

# Load toy dataset of absolute copynumber profile
load(system.file("extdata", "toy_segTab.RData", 
  package = "sigminer", mustWork = TRUE
))
cn <- read_copynumber(segTabs, 
  seg_cols = c("chromosome", "start", "end", "segVal"), 
  genome_build = "hg19", complement = FALSE 
)
cn

# Load metastatic tumor segtab.txt
load(system.file("extdata", "metastatic_tumor.segtab.txt", 
  package = "sigminer", mustWork = TRUE
))
cn2 <- read_copynumber(tab_file)
cn2

---

read_maf

Read MAF Files

# Load toy dataset of absolute copynumber profile
load(system.file("extdata", "toy_segTab.RData", 
  package = "sigminer", mustWork = TRUE
))
cn <- read_copynumber(segTabs, 
  seg_cols = c("chromosome", "start", "end", "segVal"), 
  genome_build = "hg19", complement = FALSE 
)
cn

# Load metastatic tumor segtab.txt
load(system.file("extdata", "metastatic_tumor.segtab.txt", 
  package = "sigminer", mustWork = TRUE
))
cn2 <- read_copynumber(tab_file)
cn2

---

read_maf

Read MAF Files

# Load toy dataset of absolute copynumber profile
load(system.file("extdata", "toy_segTab.RData", 
  package = "sigminer", mustWork = TRUE
))
cn <- read_copynumber(segTabs, 
  seg_cols = c("chromosome", "start", "end", "segVal"), 
  genome_build = "hg19", complement = FALSE 
)
cn

# Load metastatic tumor segtab.txt
load(system.file("extdata", "metastatic_tumor.segtab.txt", 
  package = "sigminer", mustWork = TRUE
))
cn2 <- read_copynumber(tab_file)
cn2

---
Usage

```
read_maf(maf, verbose = TRUE)
```

Arguments

- `maf`: tab delimited MAF file. File can also be gz compressed. Required. Alternatively, you can also provide already read MAF file as a dataframe.
- `verbose`: TRUE logical. Default to be talkative and prints summary.

See Also

- `read_copynumber` for reading copy number data to `CopyNumber` object.

Examples

```
laml.maf <- system.file("extdata", "tcga_laml.maf.gz", package = "maftools", mustWork = TRUE)
if (!require("R.utils")) {
  message("Please install 'R.utils' package firstly")
} else {
  laml <- read_maf(maf = laml.maf)
  laml
}
```

---

`report_bootstrap_p_value`

*Report P Values from bootstrap Results*

Description

See examples in `sig_fit_bootstrap`.

Usage

```
report_bootstrap_p_value(x, thresholds = c(0.01, 0.05, 0.1))
```

Arguments

- `x`: a (list of) result from `sig_fit_bootstrap`.
- `thresholds`: a vector of relative exposure threshold for calculating p values.

Value

a (list of) matrix
scoring  

**Score Copy Number Profile**

**Description**  
Returns quantification of copy number profile and events including tandem duplication and Chromothripsis etc. Only copy number data from autosome is used here. **Some of the quantification methods are rough, you use at your risk.** You should do some extra work to check the result scores.

**Usage**

```r
scoring(object, TD_size_cutoff = c(1000, 1e+05, 2e+06), TD_cn_cutoff = Inf)
```

**Arguments**

- `object`  
a object of `CopyNumber`.

- `TD_size_cutoff`  
a length-3 numeric vector used to specify the start, midpoint, end segment size for determining tandem duplication size range, midpoint is used to split TD into short TD and long TD. Default is 1Kb to 100Kb for short TD, 100Kb to 2Mb for long TD.

- `TD_cn_cutoff`  
a number defining the maximum copy number of TD, default is `Inf`, i.e. no cutoff.

**Value**

a data.table with following scores:

- **cnaBurden**: CNA burden representing the altered genomic fraction as previously reported.
- **cnaLoad**: CNA load representing the quantity of copy number alteration.
- **MACN**: mean altered copy number (MACN) reflecting the property of altered copy number segments, calculated as

  \[
  MACN = \frac{\sum_i CN_i}{N_{cnv}}
  \]

  where \( CN_i \) is the copy number of altered segment \( i \), \( N_{cnv} \) is the number of CNV.

- **weightedMACN**: same as MACN but weighted with segment length.

  \[
  MACN_{weighted} = \frac{\sum_i (CN_i \times L_i)}{\sum_i L_i}
  \]

  where \( L_i \) is the length of altered copy number segment \( i \).

- **Ploidy**: ploidy, the formula is same as `weightedMACN` but using all copy number segments instead of altered copy number segments.
• TDP_pnas: tandem duplication phenotype score from https://www.pnas.org/content/113/17/E2373, the threshold k in reference is omitted.

\[ TDP = \sum_{chr} |TD_{obs} - TD_{exp}| \]

where \( TD_{total} \) is the number of TD, \( TD_{obs} \) and \( TD_{exp} \) are observed number of TD and expected number of TD for each chromosome.

• TDP: tandem duplication score used defined by our group work, TD represents segment with copy number greater than 2.

\[ TD = \sum_{chr} \frac{TD_{obs} - TD_{exp}}{TD_{total}} + 1 \]

• sTDP: TDP score for short TD.

• ITDP: TDP score for long TD.

• TDP_size : TDP region size (Mb).

• sTDP_size: sTDP region size (Mb).

• lTDP_size: lTDP region size(Mb).

• Chromoth_state: chromothripsis state score, according to reference http://dx.doi.org/10.1016/j.cell.2013.02.023, chromothripsis frequently leads to massive loss of segments on the affected chromosome with segmental losses being interspersed with regions displaying normal (disomic) copy-number (e.g., copy-number states oscillating between copy-number = 1 and copy-number = 2), form tens to hundreds of locally clustered DNA rearrangements. Most of methods use both SV and CNV to infer chromothripsis, here we roughly quantify it with

\[ \sum_{chr} N_{OsCN} \]

where \( N_{OsCN} \) is the number of oscillating copy number pattern “2-1-2” for each chromosome.

Examples

```r
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData", package = "sigminer", mustWork = TRUE ))

d <- scoring(cn)
d

d2 <- scoring(cn, TD_cn_cutoff = 4L)
d2
```
show_catalogue

Show Alteration Catalogue Profile

Usage

```r
show_catalogue(
  catalogue,
  mode = c("SBS", "copynumber", "DBS", "ID"),
  method = "Wang",
  normalize = c("raw", "row", "feature"),
  style = c("default", "cosmic"),
  samples = NULL,
  samples_name = NULL,
  x_lab = "Components",
  y_lab = "Counts",
  ...
)
```

Arguments

- `catalogue`: result from `sig_tally` or a matrix with row representing components (motifs) and column representing samples.
- `mode`: signature type for plotting, now supports 'copynumber', 'SBS', 'DBS' and 'ID'.
- `method`: method for copy number feature classification in `sig_tally`, can be one of "Macintyre" ("M") and "Wang" ("W").
- `normalize`: normalize method.
- `style`: plot style, one of 'default' and 'cosmic'.
- `samples`: default is NULL, show sum of all samples in one row. If not NULL, show specified samples.
- `samples_name`: set the sample names shown in plot.
- `x_lab`: x axis lab.
- `y_lab`: y axis lab.
- `...`: other arguments passing to `show_sig_profile`.

Value

a ggplot object
show_cn_circos

Examples

load(system.file("extdata", "toy_copynumber_tally_M.RData", package = "sigminer", mustWork = TRUE))
p <- show_catalogue(cn_tally_M, mode = "copynumber", method = "M", style = "cosmic", paint_axis_text = FALSE)
p

show_cn_circos  Show Copy Number Profile in Circos

Description

Another visualization method for copy number profile like show_cn_profile.

Usage

show_cn_circos(
data, samples = NULL, show_title = TRUE, chrs = paste0("chr", 1:22), genome_build = c("hg19", "hg38"), col = NULL, side = "inside", ...
)

Arguments

data a CopyNumber object or a data.frame containing at least 'chromosome', 'start', 'end', 'segVal' these columns.
samples default is NULL, can be a character vector representing multiple samples or number of samples to show. If data argument is a data.frame, a column called sample must exist.
show_title if TRUE (default), show title with sample ID.
chrs chromosomes start with 'chr'.
genome_build genome build version, used when data is a data.frame, should be 'hg19' or 'hg38'.
col colors for the heatmaps. If it is NULL, set to circlize::colorRamp2(c(1,2,4),c("blue","black","red")
side side of the heatmaps.
... other parameters passing to circlize::circos.genomicHeatmap.
show_cn_components

Value

a circos plot

Examples

```r
load(system.file("extdata", "toy_copynumber.RData", package = "sigminer", mustWork = TRUE))

show_cn_circos(cn, samples = 1)
show_cn_circos(cn, samples = "TCGA-99-7458-01A-11D-2035-01")

## Remove title
show_cn_circos(cn, samples = 1, show_title = FALSE)

## Subset chromosomes
show_cn_circos(cn, samples = 1, chrs = c("chr1", "chr2", "chr3"))

## Arrange plots
layout(matrix(1:4, 2, 2))
show_cn_circos(cn, samples = 4)
layout(1) # reset layout
```

---

**show_cn_components**

Show Copy Number Components

### Description

Show mixture fit model components ("Macintyre" ("M") method) or standard classified components ("Wang" ("W") method) for copy number data.

### Usage

```r
show_cn_components(parameters, method = "Macintyre", show_weights = TRUE, log_segsizesize = TRUE, log_y = FALSE, auto_transform = TRUE, return_plotlist = FALSE, base_size = 12, nrow = 2, align = "hv", ...)
```
show_cn_components  

Arguments

parameters  a data.frame contain parameter components, obtain this from sig_tally function.

method  method for feature classification, can be one of "Macintyre" ("M"), "Wang" ("W") and "Tao & Wang" ("T").

show_weights  default is TRUE, show weights for each component. Only used when method is "Macintyre".

log_segszie  default is TRUE, show log10 based segsize, only works for input from "Macintyre" ("M") method.

log_y  logical, if TRUE, show log10 based y axis, only works for input from "Wang" ("W") method.

auto_transform  default is TRUE, it will auto increase the SD for components for showing them better in the plot. Only used when method is "Macintyre".

return_plotlist  if TRUE, return a list of ggplot objects but a combined plot.

base_size  overall font size.

nrow  (optional) Number of rows in the plot grid.

align  (optional) Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. Options are "none" (default), "hv" (align in both directions), "h", and "v".

...  other options pass to plot_grid function of cowplot package.

Value

a ggplot object

Author(s)

Shixiang Wang w_shixiang@163.com

Examples

load(system.file("extdata", "toy_copynumber_tally_M.RData",  
    package = "sigminer", mustWork = TRUE  
))
p1 <- show_cn_components(cn_tally_M$parameters)  
p1
p2 <- show_cn_components(cn_tally_M$parameters, show_weights = FALSE)  
p2

load(system.file("extdata", "toy_copynumber_tally_W.RData",  
    package = "sigminer", mustWork = TRUE  
))
p3 <- show_cn_components(cn_tally_W$parameters, method = "W")  
p3
show_cn_distribution  Show Copy Number Distribution either by Length or Chromosome

Description
Visually summarize copy number distribution either by copy number segment length or chromosome. Input is a CopyNumber object, genome_build option will read from genome_build slot of object.

Usage

```
show_cn_distribution(
  data,
  rm_normal = TRUE,
  mode = c("ld", "cd"),
  fill = FALSE,
  scale_chr = TRUE,
  base_size = 14
)
```

Arguments

- **data**: a CopyNumber object.
- **rm_normal**: logical. Whether remove normal copy (i.e. "segVal" equals 2), default is TRUE.
- **mode**: either "ld" for distribution by CN length or "cd" for distribution by chromosome.
- **fill**: when mode is "cd" and fill is TRUE, plot percentage instead of count.
- **scale_chr**: logical. If TRUE, normalize count to per Megabase unit.
- **base_size**: overall font size.

Value

a ggplot object

Author(s)

Shixiang Wang w_shixiang@163.com

Examples

```
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData", package = "sigminer", mustWork = TRUE))

# Plot distribution
p1 <- show_cn_distribution(cn)
p1
p2 <- show_cn_distribution(cn, mode = "cd")
```
show_cn_features(p2, p3) <- show_cn_distribution(cn, mode = "cd", fill = TRUE) p3

show_cn_features

Show Copy Number Feature Distributions

Description

Show Copy Number Feature Distributions

Usage

show_cn_features(features, method = "Macintyre", rm_outlier = FALSE, ylab = NULL, log_segsize = TRUE, log_y = FALSE, return_plotlist = FALSE, base_size = 12, nrow = 2, align = "hv", ...)

Arguments

features a feature list generate from `sig_tally` function.
method method for feature classification, can be one of "Macintyre" ("M"), "Wang" ("W") and "Tao & Wang" ("T").
rm_outlier default is FALSE, if TRUE, remove outliers. Only works when method is "Wang" ("W").
ylab lab of y axis.
log_segsize default is TRUE, show log10 based segsize, only works for input from "Macintyre" ("M") method.
log_y logical, if TRUE, show log10 based y axis, only works for input from "Wang" ("W") method.
return_plotlist if TRUE, return a list of ggplot objects but a combined plot.
base_size overall font size.
nrow (optional) Number of rows in the plot grid.
align (optional) Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. Options are "none" (default), "hv" (align in both directions), "h", and "v".
... other options pass to `plot_grid` function of cowplot package.
Value

a ggplot object

Examples

```r
# Load copy number prepare object
load(system.file("extdata", "toy_copynumber_tally_M.RData",
    package = "sigminer", mustWork = TRUE
))
p <- show_cn_features(cn_tally_M$features)
p
```

show_cn_profile  
Show Sample Copy Number Profile

Description

Sometimes it is very useful to check details about copy number profile for one or multiple samples. This function is designed to do this job and can be further modified by ggplot2 related packages.

Usage

```r
show_cn_profile(
    data,
    samples = NULL,
    show_n = NULL,
    show_title = FALSE,
    chrs = paste0("chr", 1:22),
    genome_build = c("hg19", "hg38"),
    nrow = NULL,
    ncol = NULL,
    return_plotlist = FALSE,
    .call = FALSE
)
```

Arguments

data  
a CopyNumber object or a data.frame containing at least 'chromosome', 'start', 'end', 'segVal' these columns.
samples  
default is NULL, can be a chracter vector representing multiple samples. If data argument is a data.frame, a column called sample must exist.
show_n  
number of samples to show, this is used for checking.
show_title  
if TRUE, show title for multiple samples.
chrs  
chromosomes start with 'chr'.
genome_build  
genome build version, used when data is a data.frame, should be 'hg19' or 'hg38'.
show_cosmic_sig_profile  

**nrow**
number of rows in the plot grid when multiple samples are selected.

**ncol**
number of columns in the plot grid when multiple samples are selected.

**return_plotlist**
default is FALSE, if TRUE, return a plot list instead of a combined plot.

**.call**
User should not use it.

**Value**
a ggplot object or a list

**Examples**

```r
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData", 
  package = "sigminer", mustWork = TRUE
))

p <- show_cn_profile(cn, nrow = 2, ncol = 1)
p
```

---

**show_cosmic_sig_profile**

*Plot COSMIC Signature Profile*

**Description**

Plot COSMIC Signature Profile

**Usage**

```r
show_cosmic_sig_profile(
  sig_index = NULL,
  show_index = TRUE,
  sig_db = "legacy",
  ...
)
```

**Arguments**

- **sig_index**: a vector for signature index. "ALL" for all signatures.
- **show_index**: if TRUE, show valid indices.
- **sig_db**: can be 'legacy' (for COSMIC v2 'SBS'), 'SBS', 'DBS', 'ID' and 'TSB' (for SBS transcriptional strand bias signatures). Default 'legacy'.
- **...**: other arguments passing to `show_sig_profile`.  

---

**Examples**

```r
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData", 
  package = "sigminer", mustWork = TRUE
))

p <- show_cn_profile(cn, nrow = 2, ncol = 1)
p
```
Value

a ggplot object

Author(s)

Shixiang Wang w_shixiang@163.com

Examples

show_cosmic_sig_profile()
show_cosmic_sig_profile(sig_db = "SBS")
show_cosmic_sig_profile(sig_index = 1:5)
show_cosmic_sig_profile(sig_db = "SBS", sig_index = c("10a", "17a"))

gg <- show_cosmic_sig_profile(sig_index = 1:5)
gg$aetiology

show_groups(grp_dt, ...)

show_groups

Show Signature Contribution in Clusters

Description

See example section in sig_fit() for an examples.

Usage

show_groups(grp_dt, ...)

Arguments

grp_dt
da result data.table from get_groups.

...parameters passing to legend(), e.g. x ="topleft".

Value

nothing.

See Also
get_groups, sig_fit.
show_group_comparison   Plot Group Comparison Result

Description

Using result data from get_group_comparison, this function plots genotypes/phenotypes comparison between signature groups using ggplot2 package and return a list of ggplot object contains individual and combined plots. The combined plot is easily saved to local using cowplot::save_plot(). Of note, default fisher test p values are shown for categorical data and fdr values are shown for continuous data.

Usage

```r
show_group_comparison(
    group_comparison,
    xlab = "group",
    ylab_co = NA,
    legend_title_ca = NA,
    legend_position_ca = "bottom",
    set_ca_sig_yaxis = FALSE,
    set_ca_custom_xlab = FALSE,
    show_pvalue = TRUE,
    ca_p_threshold = 0.01,
    method = "wilcox.test",
    p.adjust.method = "fdr",
    base_size = 12,
    font_size_x = 12,
    text_angle_x = 30,
    text_hjust_x = 0.2,
    ...
)
```

Arguments

- `group_comparison`  
  a list from result of get_group_comparison function.
- `xlab`  
  lab name of x axis for all plots. if it is NA, remove title for x axis.
- `ylab_co`  
  lab name of y axis for plots of continuous type data. Of note, this argument should be a character vector has same length as group_comparison, the location for categorical type data should mark with NA.
- `legend_title_ca`  
  legend title for plots of categorical type data.
- `legend_position_ca`  
  legend position for plots of categorical type data. Of note, this argument should be a character vector has same length as group_comparison, the location for continuous type data should mark with NA.
set_ca_sig_yaxis
if TRUE, use y axis to show signature proportion instead of variable proportion.

set_ca_custom_xlab
only works when set_ca_sig_yaxis is TRUE. If TRUE, set x labels using input xlab, otherwise variable names will be used.

show_pvalue
if TRUE, show p values.

c_a_p_threshold
a p threshold for categorical variables, default is 0.01. A p value less than 0.01 will be shown as \( P < 0.01 \).

method
a character string indicating which method to be used for comparing means. It can be 't.test', 'wilcox.test' etc..

p.adjust.method
the correction method, default is 'fdr'. Run p.adjust.methods to see all available options.

base_size
overall font size.

font_size_x
font size for x.

text_angle_x
text angle for x.

text_hjust_x
adjust x axis text

... other parameters pass to \texttt{ggpubr::compare_means()} or \texttt{ggpubr::stat_compare_means()} according to the specified method.

Value

a list of ggplot objects.

Author(s)

Shixiang Wang \texttt{w\_shixiang@163.com}

Examples

```r
load(system.file("extdata", "toy_copynumber_signature_by_M.RData", 
package = "sigminer", mustWork = TRUE 
))

# Assign samples to clusters 
groups <- get_groups(sig, method = "k-means")

set.seed(1234)

groups$prob <- rnorm(10)
groups$new_group <- sample(c("1", "2", "3", "4", NA), size = nrow(groups), replace = TRUE)

# Compare groups (filter NAs for categorical columns) 
groups.cmp <- get_group_comparison(groups[, -1], 
  col_group = "group", 
  cols_to_compare = c("prob", "new_group"), 
  type = c("co", "ca"), verbose = TRUE 
)
```
# Compare groups (Set NAs of categorical columns to 'Rest')
groups.cmp2 <- get_group_comparison(groups[, -1],
  col_group = "group",
  cols_to_compare = c("prob", "new_group"),
  type = c("co", "ca"), NAs = "Rest", verbose = TRUE
)

show_group_comparison(groups.cmp)

ggcomp <- show_group_comparison(groups.cmp2)
ggcomp$co_comb
ggcomp$ca_comb

---

show_group_mapping  

Map Groups using Sankey

Description

This feature is designed for signature analysis. However, users can also use it in other similar situations.

Usage

show_group_mapping(
  data,  
  col_to_flow,  
  cols_to_map,  
  include_sig = FALSE,  
  fill_na = FALSE,  
  title = NULL,  
  xlab = NULL,  
  ylab = NULL,  
  custom_theme = cowplot::theme_minimal_hgrid()
)

Arguments

data  
a data.frame containing signature group and other categorical groups.
col_to_flow  
length-1 character showing the column to flow, typically a signature group.
cols_to_map  
character vector showing colnames of other groups.
include_sig  
default if FALSE, if TRUE, showing signature group.
fill_na  
length-1 string to fill NA, default is FALSE.
title  
the title.
xlab  
label for x axis.
ylab  
label for y axis.
custom_theme  
theme for plotting, default is cowplot::theme_minimal_hgrid().
show_sig_bootstrap

Value

a ggplot object

Examples

```r
data <- dplyr::tibble(
  Group1 = rep(LETTERS[1:5], each = 10),
  Group2 = rep(LETTERS[6:15], each = 5),
  zzzz = c(rep("xx", 20), rep("yy", 20), rep(NA, 10))
)
p1 <- show_group_mapping(data, col_to_flow = "Group1", cols_to_map = colnames(data)[-1])
p1

p2 <- show_group_mapping(data,
  col_to_flow = "Group1", cols_to_map = colnames(data)[-1],
  include_sig = TRUE
)
p2
```

show_sig_bootstrap  
Show Signature Bootstrap Analysis Results

Description

See details for description.

Usage

```r
show_sig_bootstrap_exposure(
  bt_result, 
  sample = NULL, 
  signatures = NULL, 
  methods = "QP", 
  plot_fun = c("boxplot", "violin"), 
  agg_fun = c("mean", "median", "min", "max"), 
  highlight = "auto", 
  highlight_size = 4, 
  palette = "aaas", 
  title = NULL, 
  xlab = FALSE, 
  ylab = "Signature exposure", 
  width = 0.3, 
  dodge_width = 0.8, 
  outlier.shape = NA, 
  add = "jitter", 
  add.params = list(alpha = 0.3),
  ... 
)
```
show_sig_bootstrap_error(
  bt_result,
  sample = NULL,
  methods = "QP",
  plot_fun = c("boxplot", "violin"),
  agg_fun = c("mean", "median"),
  highlight = "auto",
  highlight_size = 4,
  palette = "aaas",
  title = NULL,
  xlab = FALSE,
  ylab = "Reconstruction error (F2 norm)",
  width = 0.3,
  dodge_width = 0.8,
  outlier.shape = NA,
  add = "jitter",
  add.params = list(alpha = 0.3),
  legend = "none",
  ...
)

show_sig_bootstrap_stability(
  bt_result,
  signatures = NULL,
  measure = c("RMSE", "MAE", "AbsDiff"),
  methods = "QP",
  plot_fun = c("boxplot", "violin"),
  palette = "aaas",
  title = NULL,
  xlab = FALSE,
  ylab = "Signature instability",
  width = 0.3,
  outlier.shape = NA,
  add = "jitter",
  add.params = list(alpha = 0.3),
  ...
)

Arguments

bt_result       result object from sig_fit_bootstrap_batch.
sample          a sample id.
signatures       signatures to show.
methods          a subset of c("NNLS", "QP", "SA").
plot_fun         set the plot function.
agg_fun          set the aggregation function when sample is NULL.
highlight

set the color for optimal solution. Default is "auto", which use the same color as bootstrap results, you can set it to color like "red", "gold", etc.

highlight_size

size for highlighting triangle, default is 4.

palette

the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".

title

plot main title.

xlab

character vector specifying x axis labels. Use xlab = FALSE to hide xlab.

ylab

character vector specifying y axis labels. Use ylab = FALSE to hide ylab.

width

numeric value between 0 and 1 specifying box width.

dodge_width

dodge width.

outlier.shape

Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting outlier.shape = NA. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

add

character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.

add.params

parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").

... other parameters passing to ggpubr::ggboxplot or ggpubr::ggviolin.

legend

character specifying legend position. Allowed values are one of "top", "bottom", "left", "right", "none"). To remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.

measure

measure to estimate the exposure instability, can be one of 'RMSE', 'MAE' and 'AbsDiff'.

Details

Functions:

• show_sig_bootstrap_exposure - this function plots exposures from bootstrap samples with both dotted boxplot. The optimal exposure (the exposure from original input) is shown as triangle point. Only one sample can be plotted.


• **show_sig_bootstrap_error** - this function plots decomposition errors from bootstrap samples with both dotted boxplot. The error from optimal solution (the decomposition error from original input) is shown as triangle point. **Only one sample can be plotted.**

• **show_sig_bootstrap_stability** - this function plots the signature exposure instability for specified signatures. Currently, the instability measure supports 3 types:
  
  – ’RMSE’ for Mean Root Squared Error (default) of bootstrap exposures and original exposures for each sample.
  – ’MAE’ for Mean Absolute Error of bootstrap exposures and original exposures for each sample.
  – ’AbsDiff’ for Absolute Difference between mean bootstrap exposure and original exposure.

**Value**

a ggplot object

**References**


**See Also**

sig_fit_bootstrap_batch, sig_fit, sig_fit_bootstrap

**Examples**

```r
if (require("BSgenome.Hsapiens.UCSC.hg19")) {
  laml.maf <- system.file("extdata", "tcga_laml.maf.gz", package = "maftools")
  laml <- read_maf(maf = laml.maf)
  mt_tally <- sig_tally(laml,
                       ref_genome = "BSgenome.Hsapiens.UCSC.hg19",
                       use_syn = TRUE)

  library(NMF)
  mt_sig <- sig_extract(mt_tally$nmf_matrix,
                        n_sig = 3,
                        nrun = 2,
                        cores = 1,
                        pConstant = 1e-13)

  mat <- t(mt_tally$nmf_matrix)
  mat <- mat[, colSums(mat) > 0]
  bt_result <- sig_fit_bootstrap_batch(mat, sig = mt_sig, n = 10)
  ## Parallel computation
  ## bt_result = sig_fit_bootstrap_batch(mat, sig = mt_sig, n = 10, use_parallel = TRUE)
```

## At default, mean bootstrap exposure for each sample has been calculated
p <- show_sig_bootstrap_exposure(bt_result, methods = c("QP"))
## Show bootstrap exposure (optimal exposure is shown as triangle)
p1 <- show_sig_bootstrap_exposure(bt_result, methods = c("QP"), sample = "TCGA-AB-2802")
p1
p2 <- show_sig_bootstrap_exposure(bt_result, 
  methods = c("QP"),
  sample = "TCGA-AB-3012",
  signatures = c("Sig1", "Sig2")
)
p2

## Show bootstrap error
## Similar to exposure above
p <- show_sig_bootstrap_error(bt_result, methods = c("QP"))
p
p3 <- show_sig_bootstrap_error(bt_result, methods = c("QP"), sample = "TCGA-AB-2802")
p3

## Show exposure (in)stability
p4 <- show_sig_bootstrap_stability(bt_result, methods = c("QP"))
p4
p5 <- show_sig_bootstrap_stability(bt_result, methods = c("QP"), measure = "MAE")
p5
p6 <- show_sig_bootstrap_stability(bt_result, methods = c("QP"), measure = "AbsDiff")
p6

} else {
  message("Please install package 'BSgenome.Hsapiens.UCSC.hg19' firstly!")
}

show_sig_consensusmap  
---

**Show Signature Consensus Map**

### Description
This function is a wrapper of `NMF::consensusmap()`.

### Usage

```r
show_sig_consensusmap(
  sig,
  main = "Consensus matrix",
  tracks = c("consensus:", "silhouette:"),
  lab_row = NA,
  lab_col = NA,
  ...
)
```
show_sig_exposure

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sig</td>
<td>a Signature object obtained from <code>sig_extract</code>.</td>
</tr>
<tr>
<td>main</td>
<td>Main title as a character string or a grob.</td>
</tr>
</tbody>
</table>
| tracks     | Special additional annotation tracks to highlight associations between basis components and sample clusters:

**basis** matches each row (resp. column) to the most contributing basis component in basismap (resp. coefmap). In basismap (resp. coefmap), adding a track `':basis'` to annCol (resp. annRow) makes the column (resp. row) corresponding to the component being also highlighted using the matching colours.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lab_row</td>
<td>labels for the rows.</td>
</tr>
<tr>
<td>lab_col</td>
<td>labels for the columns.</td>
</tr>
<tr>
<td>...</td>
<td>other parameters passing to <code>NMF::consensusmap()</code>.</td>
</tr>
</tbody>
</table>

Value

nothing

show_sig_exposure

Plot Signature Exposure

Description

Currently support copy number signatures and mutational signatures.

Usage

```r
show_sig_exposure(
  Signature,
  sig_names = NULL,
  groups = NULL,
  grp_order = NULL,
  grp_size = NULL,
  cutoff = NULL,
  style = c("default", "cosmic"),
  palette = use_color_style(style),
  base_size = 12,
  font_scale = 1,
  rm_space = FALSE,
  rm_grid_line = TRUE,
  rm_panel_border = FALSE,
  hide_samps = TRUE,
  legend_position = "top"
)
```
show_sig_exposure

Arguments

Signature | a Signature object obtained either from `sig_extract` or `sig_auto_extract`, or just a raw exposure matrix with column representing samples (patients) and row representing signatures (row names must start with `Sig`).

sig_names | set name of signatures, can be a character vector.

groups | sample groups, default is NULL.

grp_order | order of groups, default is NULL.

grp_size | font size of groups.

cutoff | a cutoff value to remove hyper-mutated samples.

style | plot style, one of 'default' and 'cosmic', works when parameter `set_gradient_color` is FALSE.

palette | palette used to plot, default use a built-in palette according to parameter `style`.

base_size | overall font size.

font_scale | a number used to set font scale.

rm_space | default is FALSE, if TRUE, it will remove border color and expand the bar width to 1. This is useful when the sample size is big.

rm_grid_line | default is FALSE, if TRUE, remove grid lines of plot.

rm_panel_border | default is TRUE for style 'cosmic', remove panel border to keep plot tight.

hide_samps | if TRUE, hide sample names.

legend_position | position of legend, default is 'top'.

Value

a ggplot object

Author(s)

Shixiang Wang

Examples

```r
# Load mutational signature
load(system.file("extdata", "toy_mutational_signature.RData",
               package = "sigminer", mustWork = TRUE
             ))
# Show signature exposure
p1 <- show_sig_exposure(sig2)
p1

# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature_by_M.RData",
                  package = "sigminer", mustWork = TRUE
               ))
# Show signature exposure
```
show_sig_feature_corrplot

*Draw Corrplot for Signature Exposures and Other Features*

**Description**

This function is for association visualization. Of note, the parameters `p_val` and `drop` will affect the visualization of association results under p value threshold.

**Usage**

```r
show_sig_feature_corrplot(
  tidy_cor,
  feature_list,
  sort_features = FALSE,
  drop = TRUE,
  return_plotlist = FALSE,
  p_val = 0.05,
  xlab = "Signatures",
  ylab = "Features",
  co_gradient_colors = scale_color_gradient2(low = "blue", mid = "white", high = "red",
                                            midpoint = 0),
  ca_gradient_colors = co_gradient_colors,
  plot_ratio = "auto",
  breaks_count = c(0L, 200L, 400L, 600L, 800L, 1020L)
)
```

**Arguments**

- `tidy_cor`: data returned by `get_tidy_association`.
- `feature_list`: a character vector contains features want to be plotted. If missing, all features will be used.
- `sort_features`: default is `FALSE`, use feature order obtained from the previous step. If `TRUE`, sort features as `feature_list`.
- `drop`: if `TRUE`, when a feature has no association with all signatures (p value larger than threshold set by `p_val`), this feature will be removed from the plot. Otherwise, this feature (a row) will keep with all blank white.
- `return_plotlist`: if `TRUE`, return as a list of `ggplot` objects.
- `p_val`: p value threshold. If p value larger than this threshold, the result becomes blank white.
- `xlab`: label for x axis.
show_sig_fit

Description

See `sig_fit` for examples.

Usage

show_sig_fit(
  fit_result,
  samples = NULL,
  signatures = NULL,
  plot_fun = c("boxplot", "violin", "scatter"),
  palette = "aaas",
  title = NULL,
  xlab = FALSE,
  ylab = "Signature exposure",
)

Examples

# The data is generated from Wang, Shixiang et al.
load(system.file("extdata", "asso_data.RData",

  package = "sigminer", mustWork = TRUE))

p <- show_sig_feature_corrplot(tidy_data.seqz.feature, p_val = 0.05)
p

Value

a ggplot2 object

See Also

`get_tidy_association` and `get_sig_feature_association`
```r
show_sig_fit

legend = "none",
width = 0.3,
outlier.shape = NA,
add = "jitter",
add.params = list(alpha = 0.3),
...
)

Arguments

fit_result result object from `sig_fit`.
samples samples to show, if NULL, all samples are used.
signatures signatures to show.
plot_fun set the plot function.
palette the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
title plot main title.
xlabs character vector specifying x axis labels. Use xlab = FALSE to hide xlab.
ylab character vector specifying y axis labels. Use ylab = FALSE to hide ylab.
legend character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). To remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.
width numeric value between 0 and 1 specifying box width.
outlier.shape Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting outlier.shape = NA. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.
add character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.
add.params parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
...
other arguments to be passed to `geom_boxplot`, `ggpar` and `facet`.
```
show_sig_number_survey

Show Simplified Signature Number Survey

Description

`sig_estimate` shows comprehensive rank survey generated by NMF package, sometimes it is hard to consider all measures. Here provides a one or two y-axis visualization method to help users determine the optimal signature number (showing both stability ("cophenetic") and error (RSS) at default). Users can also set custom measures to show.

Usage

```r
show_sig_number_survey(
  object,
  x = "rank",
  left_y = "cophenetic",
  right_y = "rss",
  left_name = left_y,
  right_name = toupper(right_y),
  left_color = "black",
  right_color = "red"
)
```

Arguments

- `object` a `Survey` object generated from `sig_estimate`, or a `data.frame` contains at least rank columns and columns for one measure.
- `x` column name for x axis.
- `left_y` column name for left y axis.
- `right_y` column name for right y axis.
- `left_name` label name for left y axis.
- `right_name` label name for right y axis.
- `left_color` color for left axis.
- `right_color` color for right axis.

Value

a `ggplot` object

See Also

`sig_fit`, `show_sig_bootstrap_exposure`, `sig_fit_bootstrap`, `sig_fit_bootstrap_batch`
show_sig_number_survey2

See Also

`sig_estimate` for estimating signature number for `sig_extract`, `show_sig_number_survey2` for more visualization method.

Examples

```r
load(system.file("extdata", "toy_copynumber_tally_M.RData", 
    package = "sigminer", mustWork = TRUE 
))
library(NMF)

cn_estimate <- sig_estimate(cn_tally_M$nmf_matrix, 
    cores = 1, nrun = 5, 
    verbose = TRUE 
)

# Show two measures
show_sig_number_survey(cn_estimate)
# Show one measure
p <- show_sig_number_survey(cn_estimate, right_y = NULL)
p <- add_h_arrow(p, x = 4.1, y = 0.953, label = "selected number")
p

# Show data from a data.frame
show_sig_number_survey(cn_estimate$survey)
# Show other measures
head(cn_estimate$survey)
show_sig_number_survey(cn_estimate$survey, 
    right_y = "dispersion", 
    right_name = "dispersion" 
)
show_sig_number_survey(cn_estimate$survey, 
    right_y = "evar", 
    right_name = "evar"
)
```

show_sig_number_survey2

*Show Comprehensive Signature Survey*

Description

This function is modified from `NMF` package to better help users to explore survey of signature number.
show_sig_number_survey2

Usage

show_sig_number_survey2(
  x,
  y = NULL,
  what = c("all", "cophenetic", "rss", "residuals", "dispersion", "evar", "sparseness",
           "sparseness.basis", "sparseness.coef", "silhouette", "silhouette.coef",
           "silhouette.basis", "silhouette.consensus"),
  na.rm = FALSE,
  xlab = "Number of signature",
  ylab = "",
  main = "Signature number survey using NMF package"
)

Arguments

x a data.frame or NMF.rank object obtained from sig_estimate()

y for random simulation, a data.frame or NMF.rank object obtained from sig_estimate()

what a character vector whose elements partially match one of the following items, which correspond to the measures computed by summary() on each – multi-run – NMF result: 'all', 'cophenetic', 'rss', 'residuals', 'dispersion', 'evar', 'silhouette' (and more specific *.coef, *.basis, *.consensus), 'sparseness' (and more specific *.coef, *.basis). It specifies which measure must be plotted (what='all' plots all the measures).

na.rm single logical that specifies if the rank for which the measures are NA values should be removed from the graph or not (default to FALSE). This is useful when plotting results which include NAs due to error during the estimation process. See argument stop for nmfEstimateRank.

xlab x-axis label

ylab y-axis label

main main title

Value

a ggplot object

Examples

load(system.file("extdata", "toy_copynumber_tally_M.RData", package = "sigminer", mustWork = TRUE))
library(NMF)
cn_estimate <- sig_estimate(cn_tally_M$nmf_matrix, cores = 1, nrun = 5, verbose = TRUE, keep_nmfObj = TRUE)
show_sig_profile

Show Signature Profile

Description

Who don’t like to show a barplot for signature profile? This is for it.

Usage

```
show_sig_profile(
  Signature,
  mode = c("SBS", "copynumber", "DBS", "ID"),
  method = "Wang",
  normalize = c("row", "column", "raw", "feature"),
  filters = NULL,
  feature_setting = sigminer::CN.features,
  style = c("default", "cosmic"),
  set_gradient_color = FALSE,
  free_space = "free_x",
  rm_panel_border = style == "cosmic",
  rm_grid_line = FALSE,
  bar_border_color = ifelse(style == "default", "grey50", "white"),
  bar_width = 0.7,
  paint_axis_text = TRUE,
  x_label_angle = ifelse(mode == "copynumber", 60, 90),
  x_label_vjust = 1,
  x_label_hjust = 1,
  x_lab = "Components",
  y_lab = "auto",
  params = NULL,
  show_cv = FALSE,
  params_label_size = 3,
  params_label_angle = 60,
  y_expand = 1,
  digits = 2,
  base_size = 12,
  font_scale = 1,
  sig_names = NULL,
)```
show_sig_profile

```r
show_sig_profile
  
  sig_orders = NULL,
  check_sig_names = TRUE
)

Arguments
  
  Signature  a Signature object obtained either from `sig_extract` or `sig_auto_extract`, or just
  a raw signature matrix with row representing components (motifs) and column
  representing signatures (column names must start with 'Sig').
  
  mode  signature type for plotting, now supports 'copynumber', 'SBS', 'DBS' and 'ID'.
  
  method  method for copy number feature classification in `sig_tally`, can be one of "Mac-
  intyre" ("M") and "Wang" ("W").
  
  normalize  one of 'row', 'column', 'raw' and "feature", for row normalization (signature),
  column normalization (component), raw data, row normalization by feature, re-
  spectively. Of note, 'feature' only works when the mode is 'copynumber'.
  
  filters  a pattern used to select components to plot.
  
  feature_setting  a data.frame used for classification. Only used when method is "Wang"
  ("W"). Default is `CN.features`. Users can also set custom input with "feature",
  "min" and "max" columns available. Valid features can be printed by
  `unique(CN.features$feature)`.  
  
  style  plot style, one of 'default' and 'cosmic', works when parameter `set_gradient_color`
  is FALSE.
  
  palette  palette used to plot when `set_gradient_color` is FALSE, default use a built-in
  palette according to parameter style.
  
  set_gradient_color  default is FALSE, if TRUE, use gradient colors to fill bars. This is very useful
  when signatures are extracted from "Macintyre" method and `normalize` is
  'column'.
  
  free_space  default is 'free_x'. If "fixed", all panels have the same size. If "free_y" their
  height will be proportional to the length of the y scale; if "free_x" their width
  will be proportional to the length of the x scale; or if "free" both height and width
  will vary. This setting has no effect unless the appropriate scales also vary.
  
  rm_panel_border  default is TRUE for style 'cosmic', remove panel border to keep plot tight.
  
  rm_grid_line  default is FALSE, if TRUE, remove grid lines of plot.
  
  bar_border_color  the color of bar border.
  
  bar_width  bar width. By default, set to 70% of the resolution of the data.
  
  paint_axis_text  if TRUE, color on text of x axis.
  
  x_label_angle  font angle for x label.
  
  x_label_vjust  font vjust for x label.
  
  x_label_hjust  font hjust for x label.
```
show_sig_profile

- x_lab: x axis lab.
- y_lab: y axis lab.
- params: params data.frame of components, obtained from `sig_tally`
- show_cv: default is FALSE, if TRUE, show coefficient of variation when `params` is not NULL.
- params_label_size: font size for params label.
- params_label_angle: font angle for params label.
- y_expand: y expand height for plotting params of copy number signatures.
- digits: digits for plotting params of copy number signatures.
- base_size: overall font size.
- font_scale: a number used to set font scale.
- sig_names: set name of signatures, can be a character vector. Default is NULL, prefix 'Sig_' plus number is used.
- sig_orders: set order of signatures, can be a character vector. Default is NULL, the signatures are ordered by alphabetical order.
- check_sig_names: if TRUE, check signature names when input is a matrix, i.e., all signatures (col-names) must start with 'Sig'.

Value

a `ggplot` object

Author(s)

Shixiang Wang

Examples

```r
# Load SBS signature
load(system.file("extdata", "toy_mutational_signature.RData", package = "sigminer", mustWork = TRUE))
# Show signature profile
p1 <- show_sig_profile(sig2, mode = "SBS")
p1

# Load copy number signature from method "W"
load(system.file("extdata", "toy_copynumber_signature_by_W.RData", package = "sigminer", mustWork = TRUE))
# Show signature profile
p2 <- show_sig_profile(sig, 
  style = "cosmic", 
  mode = "copynumber", 
  method = "W")
```
normalize = "feature"
)
p2

# Load copy number signature from method "M"
load(system.file("extdata", "toy_copynumber_signature_by_M.RData",
    package = "sigminer", mustWork = TRUE
))
# Show signature profile
# The 'column' normalization is consistent with
# original paper
p3 <- show_sig_profile(sig,
    paint_axis_text = FALSE,
    mode = "copynumber",
    method = "M", normalize = "column"
)
p3

# Add params label
# ================
# Load copy number prepare object
load(system.file("extdata", "toy_copynumber_tally_M.RData",
    package = "sigminer", mustWork = TRUE
))
params <- get_tidy_parameter(cn_tally_M$components)
p4 <- show_sig_profile(sig,
    mode = "copynumber",
    method = "M", normalize = "column",
    params = params, y_expand = 2
)
p4

---

**Description**

Please go to [https://shixiangwang.github.io/sigminer-doc/](https://shixiangwang.github.io/sigminer-doc/) for full vignette.

**Details**

Result visualization for MAF is provide by [maftools](https://shixiangwang.github.io/sigminer-doc/) package, please read its vignette.
sig_auto_extract

Extract Signatures through the Automatic Relevance Determination Technique

Description

A bayesian variant of NMF algorithm to enable optimal inferences for the number of signatures through the automatic relevance determination technique. This functions delevers highly interpretable and sparse representations for both signature profiles and attributions at a balance between data fitting and model complexity (this method may introduce more signatures than expected, especially for copy number signatures (thus **I don’t recommend you to use this feature to extract copy number signatures**)). See detail part and references for more.

Usage

```r
sig_auto_extract(
  nmf_matrix = NULL,
  result_prefix = "BayesNMF",
  destdir = tempdir(),
  method = c("L1W.L2H", "L1KL", "L2KL"),
  strategy = c("stable", "optimal"),
  K0 = 25,
  nrun = 10,
  niter = 2e+05,
  tol = 1e-07,
  cores = 1,
  optimize = FALSE,
  skip = FALSE,
  recover = FALSE
)
```

Arguments

- **nmf_matrix**: a matrix used for NMF decomposition with rows indicate samples and columns indicate components.
- **result_prefix**: prefix for result data files.
- **destdir**: path to save data runs, default is `tempdir()`.
- **method**: default is "L1W.L2H", which uses an exponential prior for W and a half-normal prior for H (This method is used by PCAWG project, see reference #3). You can also use "L1KL" to set exponential priors for both W and H, and "L2KL" to set half-normal priors for both W and H. The latter two methods are originally implemented by SignatureAnalyzer software.
- **strategy**: the selection strategy for returned data. Set 'stable' for getting optimal result from the most frequent K. Set 'optimal' for getting optimal result from all Ks. If you want select other solution, please check `get_bayesian_result`.  

**Details**

There are three methods available in this function: "L1W.L2H", "L1KL" and "L2KL". They use different priors for the bayesian variant of NMF algorithm (see method parameter) written by reference #1 and implemented in SignatureAnalyzer software (reference #2).

I copied source code for the three methods from Broad Institute and supplementary files of reference #3, and wrote this higher function. It is more friendly for users to extract, visualize and analyze signatures by combining with other powerful functions in sigminer package. Besides, I implemented parallel computation to speed up the calculation process and a similar input and output structure like sig_extract().

**Value**

a list with Signature class.

**Author(s)**

Shixiang Wang

**References**


**See Also**

sig_tally for getting variation matrix, sig_extract for extracting signatures using NMF package, sig_estimate for estimating signature number for sig_extract.
Examples

```r
load(system.file("extdata", "toy_copynumber_tally_M.RData", 
  package = "sigminer", mustWork = TRUE

))
res <- sig_auto_extract(cn_tally_M$nmf_matrix, result_prefix = "Test_copynumber", nrun = 1)
# At default, all run files are stored in tempdir()
dir(tempdir(), pattern = "Test_copynumber")
```

---

**sig_convert**

*Convert Signatures between different Genomic Distribution of Components*

Description

Converts signatures between two representations relative to different sets of mutational opportunities. Currently, only SBS signature is supported.

Usage

```r
sig_convert(sig, from = "human-genome", to = "human-exome")
```

Arguments

- `sig` a Signature object obtained either from `sig_extract` or `sig_auto_extract`, or just a raw signature matrix with row representing components (motifs) and column representing signatures.
- `from` either one of "human-genome" and "human-exome" or an opportunity matrix (repeated n columns with each row represents the total number of mutations for a component, n is the number of signature).
- `to` same as `from`.

Details

The default opportunity matrix for "human-genome" and "human-exome" comes from COSMIC signature database v2 and v3.

Value

a matrix.

References

`convert_signatures` function from sigfit package.
Examples

```r
# Load SBS signature
load(system.file("extdata", "toy_mutational_signature.RData", 
    package = "sigminer", mustWork = TRUE
))
# Exome-relative to Genome-relative
sig_converted <- sig_convert(sig2, 
    from = "human-exome", 
    to = "human-genome"
)
sig_converted

show_sig_profile(sig2, style = "cosmic")
show_sig_profile(sig_converted, style = "cosmic")
```

---

**sig_estimate**

*Estimate Signature Number*

Description

Use **NMF** package to evaluate the optimal number of signatures. This is used along with **sig_extract**. Users should `library(NMF)` firstly. If NMF objects are returned, the result can be further visualized by NMF plot methods like `NMF::consensusmap()` and `NMF::basismap()`.

Usage

```r
sig_estimate(
    nmf_matrix, 
    range = 2:5, 
    nrun = 10, 
    use_random = FALSE, 
    method = "brunet", 
    seed = 123456, 
    cores = 1, 
    keep_nmfObj = FALSE, 
    save_plots = FALSE, 
    plot_basename = file.path(tempdir(), "nmf"), 
    what = "all", 
    pConstant = NULL, 
    verbose = FALSE
)
```

Arguments

- `nmf_matrix` : a matrix used for NMF decomposition with rows indicate samples and columns indicate components.
range a numeric vector containing the ranks of factorization to try. Note that duplicates are removed and values are sorted in increasing order. The results are notably returned in this order.

nrun a numeric giving the number of run to perform for each value in range, nrun set to 30-50 is enough to achieve robust result.

use_random Should generate random data from input to test measurements. Default is TRUE.


seed specification of the starting point or seeding method, which will compute a starting point, usually using data from the target matrix in order to provide a good guess.

cores number of cpu cores to run NMF.

keep_nmfObj default is FALSE, if TRUE, keep NMF objects from runs, and the result may be huge.

save_plots if TRUE, save signature number survey plot to local machine.

plot_basename when save plots, set custom basename for file path.

what a character vector whose elements partially match one of the following item, which correspond to the measures computed by summary on each multi-run NMF result: ‘all’, ‘cophenetic’, ‘rss’, ‘residuals’, ‘dispersion’, ‘evar’, ‘silhouette’ (and more specific .coef, .basis, .consensus), ‘sparseness’ (and more specific .coef, .basis). It specifies which measure must be plotted (what=’all’ plots all the measures).

pConstant A small positive value (like 1e-9) to add to the matrix. Use it ONLY if the functions throws an non-conformable arrays error.

verbose if TRUE, print extra message.

Details

The most common approach is to choose the smallest rank for which cophenetic correlation coefficient starts decreasing (Used by this function). Another approach is to choose the rank for which the plot of the residual sum of squares (RSS) between the input matrix and its estimate shows an inflection point. More custom features please directly use NMF::nmfEstimateRank.

Value

a list contains information of NMF run and rank survey.

Author(s)
Shixiang Wang

References

See Also

sig_extract for extracting signatures using NMF package, sig_auto_extract for extracting signatures using automatic relevance determination technique.

Examples

```r
load(system.file("extdata", "toy_copynumber_tally_M.RData",
              package = "sigminer", mustWork = TRUE
))
## Not run:
library(NMF)
ecn_estimate <- sig_estimate(cn_tally_M$nmf_matrix,
               cores = 1, nrun = 5,
               verbose = TRUE
)
## End(Not run)
```

sig_extract

Extract Signatures through NMF

Description

Do NMF de-composition and then extract signatures.

Usage

```r
sig_extract(
    nmf_matrix,
    n_sig,
    nrun = 10,
    cores = 1,
    method = "brunet",
    optimize = FALSE,
    pConstant = NULL,
    seed = 123456,
    ...
)
```

Arguments

- `nmf_matrix`: a matrix used for NMF decomposition with rows indicate samples and columns indicate components.
- `n_sig`: number of signature. Please run sig_estimate to select a suitable value.
- `nrun`: a numeric giving the number of run to perform for each value in range, nrun set to 30–50 is enough to achieve robust result.
- `cores`: number of cpu cores to run NMF.
specification of the NMF algorithm. Use 'brunet' as default. Available methods for nmf decompositions are 'brunet', 'lee', 'ls-nmf', 'nsNMF', 'offset'.

optimize logical, for exposure optimization, especially useful for copy number signature.

pConstant A small positive value (like 1e-9) to add to the matrix. Use it ONLY if the functions throws an non-conformable arrays error.

seed specification of the starting point or seeding method, which will compute a starting point, usually using data from the target matrix in order to provide a good guess.

... other arguments passed to \texttt{NMF::nmf()}.

Value

a list with Signature class.

Author(s)

Shixiang Wang

References


See Also

\texttt{sig_tally} for getting variation matrix, \texttt{sig_estimate} for estimating signature number for \texttt{sig_extract}, \texttt{sig_auto_extract} for extracting signatures using automatic relevance determination technique.

Examples

\begin{verbatim}
load(system.file("extdata", "toy_copynumber_tally_M.RData", package = "sigminer", mustWork = TRUE))
# Extract copy number signatures
library(NMF)
res <- sig_extract(cn_tally_M$nmf_matrix, 2, nrun = 1)
\end{verbatim}
sig_fit (Fit Signature Exposures with Linear Combination Decomposition)

Description

The function performs a signatures decomposition of a given mutational catalogue V with known signatures W by solving the minimization problem min(\|W*H - V\|) where W and V are known.

Usage

```r
sig_fit(
  catalogue_matrix,
  sig,
  sig_index = NULL,
  sig_db = "legacy",
  show_index = TRUE,
  method = c("QP", "NNLS", "SA"),
  type = c("absolute", "relative"),
  return_class = c("matrix", "data.table"),
  return_error = FALSE,
  rel_threshold = 0,
  mode = c("SBS", "DBS", "ID", "copynumber"),
  true_catalog = NULL,
  ...
)
```

Arguments

- `catalogue_matrix`: a numeric matrix V with row representing components and columns representing samples, typically you can get `nmf_matrix` from `sig_tally()` and transpose it by `t()`.
- `sig`: a Signature object obtained either from `sig_extract` or `sig_auto_extract`, or just a raw signature matrix with row representing components (motifs) and column representing signatures.
- `sig_index`: a vector for signature index. "ALL" for all signatures.
- `sig_db`: can be 'legacy' (for COSMIC v2 'SBS'), 'SBS', 'DBS', 'ID' and 'TSB' (for SBS transcriptional strand bias signatures). Default 'legacy'.
- `db_type`: only used when `sig_db` is enabled. "" for keeping default, "human-exome" for transforming to exome frequency of component, and "human-genome" for transforming to whole genome frequency of component. Currently only works for 'SBS'.
- `show_index`: if TRUE, show valid indices.
- `method`: method to solve the minimazation problem. 'NNLS' for nonnegative least square; 'QP' for quadratic programming; 'SA' for simulated annealing.
- `type`: 'absolute' or 'relative'.
- `return_class`: output type. 'matrix' or 'data.table'.
- `return_error`: whether to return error message.
- `rel_threshold`: relative threshold.
- `mode`: 'SBS', 'DBS', 'ID', 'copynumber'.
- `true_catalog`: not used.
- `...`: further arguments passed to `sig_tally()` and `nmf()`. 
type 'absolute' for signature exposure and 'relative' for signature relative exposure.

return_class string, 'matrix' or 'data.table'.

return_error if TRUE, also return method error (Frobenius norm). NOTE: it is better to obtain the error when the type is 'absolute', because the error is affected by relative exposure accuracy.

rel_threshold numeric vector, a relative exposure lower than this value will be set to 0. Of note, this is a little different from the same parameter in get_sig_exposure.

mode signature type for plotting, now supports 'copynumber', 'SBS', 'DBS' and 'ID'.

true_catalog used by sig_fit_bootstrap, user never use it.

... control parameters passing to argument control in GenSA function when use method 'SA'.

Details

The method 'NNLS' solves the minimization problem with nonnegative least-squares constraints. The method 'QP' and 'SA' are modified from SignatureEstimation package. See references for details. Of note, when fitting exposures for copy number signatures, only components of feature CN is used.

Value

The exposure result either in matrix or data.table format. If return_error set TRUE, a list is returned.

References


See Also

sig_extract, sig_auto_extract, sig_fit_bootstrap, sig_fit_bootstrap_batch

Examples

W <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
colnames(W) <- c("sig1", "sig2")
W <- apply(W, 2, function(x) x / sum(x))

H <- matrix(c(2, 5, 3, 6, 1, 9, 1, 2), ncol = 4)
colnames(H) <- paste0("samp", 1:4)

V <- W %*% H
V
if (requireNamespace("quadprog", quietly = TRUE)) {
    H_infer <- sig_fit(V, W, method = "QP")
    H_infer
    H

    H_dt <- sig_fit(V, W, method = "QP", return_class = "data.table")
    H_dt

    ## Show results
    show_sig_fit(H_infer)
    show_sig_fit(H_dt)

    ## Get clusters/groups
    H_dt_rel <- sig_fit(V, W, return_class = "data.table", type = "relative")
    z <- get_groups(H_dt_rel, method = "k-means")
    show_groups(z)
}

if (requireNamespace("GenSA", quietly = TRUE)) {
    H_infer <- sig_fit(V, W, method = "SA")
    H_infer
    H

    H_dt <- sig_fit(V, W, method = "SA", return_class = "data.table")
    H_dt

    ## Modify arguments to method
    sig_fit(V, W, method = "SA", maxit = 10, temperature = 100)

    ## Show results
    show_sig_fit(H_infer)
    show_sig_fit(H_dt)
}

---

**sig_fit_bootstrap**

*Obtain Bootstrap Distribution of Signature Exposures of a Certain Tumor Sample*

**Description**

This can be used to obtain the confidence of signature exposures or search the suboptimal decomposition solution.

**Usage**

```r
sig_fit_bootstrap(
    catalog,
    sig,
    n = 100L,
)```

sig_index = NULL,
sig_db = "legacy",
db_type = c("", "human-exome", "human-genome"),
show_index = TRUE,
method = c("QP", "NNLS", "SA"),
SA_not_bootstrap = FALSE,
type = c("absolute", "relative"),
rel_threshold = 0,
mode = c("SBS", "DBS", "ID", "copynumber"),
find_suboptimal = FALSE,
suboptimal_ref_error = NULL,
suboptimal_factor = 1.05,
...)

Arguments

catalog a named numeric vector or a numeric matrix with dimension Nx1. N is the number of component, 1 is the sample.

sig a Signature object obtained either from sig_extract or sig_auto_extract, or just a raw signature matrix with row representing components (motifs) and column representing signatures.

n the number of bootstrap replicates.

sig_index a vector for signature index. "ALL" for all signatures.

sig_db can be 'legacy' (for COSMIC v2 'SBS'), 'SBS', 'DBS', 'ID' and 'TSB' (for SBS transcriptional strand bias signatures). Default 'legacy'.

db_type only used when sig_db is enabled. "" for keeping default, "human-exome" for transforming to exome frequency of component, and "human-genome" for transforming to whole genome frequency of component. Currently only works for 'SBS'.

show_index if TRUE, show valid indices.

method method to solve the minimazation problem. 'NNLS' for nonnegative least square; 'QP' for quadratic programming; 'SA' for simulated annealing.

SA_not_bootstrap if TRUE, directly run 'SA' multiple times with original input instead of bootstrap samples.

type 'absolute' for signature exposure and 'relative' for signature relative exposure.

rel_threshold numeric vector, a relative exposure lower than this value will be set to 0. Of note, this is a little different from the same parameter in get_sig_exposure.

mode signature type for plotting, now supports 'copynumber', 'SBS', 'DBS' and 'ID'.

find_suboptimal logical, if TRUE, find suboptimal decomposition with slightly higher error than the optimal solution by method 'SA'. This is useful to explore hidden dependencies between signatures. More see reference.
suboptimal_ref_error
  baseline error used for finding suboptimal solution. if it is NULL, then use 'SA'
  method to obtain the optimal error.

suboptimal_factor
  suboptimal factor to get suboptimal error, default is 1.05, i.e., suboptimal error
  is 1.05 times baseline error.

... control parameters passing to argument control in GenSA function when use
  method 'SA'.

Value
  a list

References
  Huang X, Wojtowicz D, Przytycka TM. Detecting presence of mutational signatures in cancer with

See Also
  sig_fit, sig_fit_bootstrap_batch

Examples
  W <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
colnames(W) <- c("sig1", "sig2")
W <- apply(W, 2, function(x) x / sum(x))

  H <- matrix(c(2, 5, 3, 6, 1, 9, 1, 2), ncol = 4)
colnames(H) <- paste0("samp", 1:4)

  V <- W %*% H
V

  if (requireNamespace("quadprog", quietly = TRUE)) {
H_bootstrap <- sig_fit_bootstrap(V[, 1], W, n = 10, type = "absolute")
## Typically, you have to run many times to get close to the answer
boxplot(t(H_bootstrap$expo))
H[, 1]

## Return P values
## In practice, run times >= 100
## is recommended
report_bootstrap_p_value(H_bootstrap)
## For multiple samples
## Input a list
report_bootstrap_p_value(list(samp1 = H_bootstrap, samp2 = H_bootstrap))

## Find suboptimal decomposition
H_suboptimal <- sig_fit_bootstrap(V[, 1], W,
  n = 10,
Exposure Instability Analysis of Signature Exposures with Bootstrapping

Usage

```
sig_fit_bootstrap_batch(
  catalogue_matrix,
  methods = c("QP"),
  n = 100L,
  min_count = 1L,
  p_val_thresholds = c(0.05),
  use_parallel = FALSE,
  seed = 123456L,
  job_id = NULL,
  result_dir = tempdir(),
  ...)
```

Arguments

- **catalogue_matrix**
  a numeric matrix $V$ with row representing components and columns representing samples, typically you can get `nmf_matrix` from `sig_tally()` and transpose it by `t()`.

- **methods**
  a subset of `c("NNLS","QP","SA")`.

- **n**
  the number of bootstrap replicates.

- **min_count**
  minimal exposure in a sample, default is 1. Any patient has total exposure less than this value will be filtered out.

- **p_val_thresholds**
  a vector of relative exposure threshold for calculating p values.

- **use_parallel**
  if `TRUE`, use parallel computation based on `furrr` package.

- **seed**
  random seed to reproduce the result.
job_id  |  a job ID, default is NULL, can be a string. When not NULL, all bootstrapped results will be saved to local machine location defined by result_dir. This is very useful for running more than 10 times for more than 100 samples.

result_dir |  see above, default is temp directory defined by R.

...  |  other common parameters passing to sig_fit_bootstrap, including sig, sig_index, sig_db, db_type, mode, etc.

Value

a list of data.table.

See Also

sig_fit, sig_fit_bootstrap

Examples

```R
W <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
colnames(W) <- c("sig1", "sig2")
W <- apply(W, 2, function(x) x / sum(x))

H <- matrix(c(2, 5, 3, 6, 1, 9, 1, 2), ncol = 4)
colnames(H) <- paste0("samp", 1:4)

V <- W %*% H
V
```

```R
if (requireNamespace("quadprog")) {
  z10 <- sig_fit_bootstrap_batch(V, sig = W, n = 10)
z10
}
```

---

## sig_names

### Obtain or Modify Signature Information

**Description**

Obtain or Modify Signature Information

**Usage**

```R
sig_names(sig)
```

```R
sig_modify_names(sig, new_names)
```

```R
sig_number(sig)
```

```R
sig_attrs(sig)
```
```r
sig_signature(sig, normalize = c("row", "column", "raw", "feature"))

sig_exposure(sig, type = c("absolute", "relative"))
```

### Arguments

- **sig**: a Signature object obtained either from `sig_extract` or `sig_auto_extract`.
- **new_names**: new signature names.
- **normalize**: one of 'row', 'column', 'raw' and "feature", for row normalization (signature), column normalization (component), raw data, row normalization by feature, respectively.
- **type**: one of 'absolute' and 'relative'.

### Value

a Signature object or data.

### Examples

```r
## Operate signature names
load(system.file("extdata", "toy_mutational_signature.RData", 
    package = "sigminer", mustWork = TRUE 
))
sig_names(sig2)
cc <- sig_modify_names(sig2, new_names = c("Sig2", "Sig1", "Sig3"))
sig_names(cc)

# The older names are stored in tags.
print(attr(cc, "tag"))
## Get signature number
sig_number(sig2)
## Get signature attributes
sig_number(sig2)
## Get signature matrix
z <- sig_signature(sig2)
z <- sig_signature(sig2, normalize = "raw")
## Get exposure matrix
## Of note, this is different from get_sig_exposure()
## it returns a matrix instead of data table.
z <- sig_exposure(sig2) # it is same as sig$Exposure
z <- sig_exposure(sig2, type = "relative") # it is same as sig$Exposure.norm
```

---

**sig_tally**  
*Tally a Genomic Alteration Object*
Description

Tally a variation object like **MAF**, **CopyNumber** and return a matrix for NMF de-composition and more. This is a generic function, so it can be further extended to other mutation cases. Please read details about how to set sex for identifying copy number signatures. Please read [https://osf.io/s93d5/](https://osf.io/s93d5/) for the generation of SBS, DBS and ID (INDEL) components. **Of note, many options are designed for method "M" only, and they are highlighted by bold fonts** (you can ignore them if you don't use "M" method).

Usage

```r
sig_tally(object, ...)

## S3 method for class 'CopyNumber'
sig_tally(
  object,
  method = "Wang",
  ignore_chrs = NULL,
  feature_setting = sigminer::CN.features,
  type = c("probability", "count"),
  reference_components = FALSE,
  cores = 1,
  seed = 123456,
  min_comp = 2,
  max_comp = 15,
  min_prior = 0.001,
  model_selection = "BIC",
  threshold = 0.1,
  nrep = 1,
  niter = 1000,
  keep_only_matrix = FALSE,
  ...
)

## S3 method for class 'MAF'
sig_tally(
  object,
  mode = c("SBS", "DBS", "ID", "ALL"),
  ref_genome = NULL,
  genome_build = NULL,
  add_trans_bias = FALSE,
  ignore_chrs = NULL,
  use_syn = TRUE,
  keep_only_matrix = FALSE,
  ...
)
```
**Arguments**

- **object**: a `CopyNumber` object or MAF object.
- **method**: custom setting for operating object. Detail see S3 method for corresponding class (e.g. `CopyNumber`).
- **ignore_chrs**: Chromosomes to ignore from analysis. e.g. chrX and chrY.
- **feature_setting**: a data.frame used for classification. **Only used when method is "Wang"** ("W"). Default is `CN.features`. Users can also set custom input with "feature", "min" and "max" columns available. Valid features can be printed by `unique(CN.features$feature)`.
- **type**: one of "probability", "count". Default is "probability", return a matrix with the sum of posterior probabilities for each components. If set to 'count', return a matrix with event count assigned to each components. The result for both types should be close. **Only used when method is "Macintyre"**.
- **reference_components**: default is `FALSE`, calculate mixture components from `CopyNumber` object. **Only used when method is "Macintyre"**.
- **cores**: number of computer cores to run this task. You can use `future::availableCores()` function to check how many cores you can use.
- **seed**: seed number. **Only used when method is "Macintyre"**.
- **min_comp**: minimal number of components to fit, default is 2. Can also be a vector with length 6, which apply to each feature. **Only used when method is "Macintyre"**.
- **max_comp**: maximal number of components to fit, default is 15. Can also be a vector with length 6, which apply to each feature. **Only used when method is "Macintyre"**.
- **min_prior**: the minimum relative size of components, default is 0.001. Details about custom setting please refer to `flexmix` package. **Only used when method is "Macintyre"**.
- **model_selection**: model selection strategy, default is 'BIC'. Details about custom setting please refer to `flexmix` package. **Only used when method is "Macintyre"**.
- **threshold**: default is 0.1. Sometimes, the result components include adjacent distributions with similar mu (two and more distribution are very close), we use this threshold to obtain a more meaningful fit with less components. **Only used when method is "Macintyre"**.
- **nrep**: number of run times for each value of component, keep only the solution with maximum likelihood. **Only used when method is "Macintyre"**.
- **niter**: the maximum number of iterations. **Only used when method is "Macintyre"**.
- **keep_only_matrix**: if `TRUE`, keep only matrix for signature extraction. For a MAF object, this will just return the most useful matrix.
mode

ref_genome

genome_build

add_trans_bias

use_syn

Details

For identifying copy number signatures, we have to derive copy number features firstly. Due to the difference of copy number values in sex chromosomes between male and female, we have to do an extra step if we don’t want to ignore them.

I create two options to control this, the default values are shown as the following, you can use the same way to set (per R session).

options(sigminer.sex = "female", sigminer.copynumber.max = NA_integer_)

- If your cohort are all females, you can totally ignore this.
- If your cohort are all males, set sigminer.sex to 'male' and sigminer.copynumber.max to a proper value (the best is consistent with read_copynumber).
- If your cohort contains both males and females, set sigminer.sex as a data.frame with two columns "sample" and "sex". And set sigminer.copynumber.max to a proper value (the best is consistent with read_copynumber).

Value

a list contains a matrix used for NMF de-composition.

Methods (by class)

- CopyNumber: Returns copy number features, components and component-by-sample matrix
- MAF: Returns SBS mutation sample-by-component matrix and APOBEC enrichment

Author(s)

Shixiang Wang
References


See Also

`sig_estimate` for estimating signature number for `sig_extract`, `sig_auto_extract` for extracting signatures using automatic relevance determination technique.

Examples

```r
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData", package = "sigminer", mustWork = TRUE
 )

# Use method designed by Wang, Shixiang et al.
cn_tally_W <- sig_tally(cn, method = "W")
# Use method designed by Macintyre et al.
cn_tally_M <- sig_tally(cn, method = "M")

# Prepare SBS signature analysis
laml.maf <- system.file("extdata", "tcga_laml.maf.gz", package = "maftools")
laml <- read_maf(maf = laml.maf)
if (require("BSgenome.Hsapiens.UCSC.hg19")) {
  mt_tally <- sig_tally(
    laml,
    ref_genome = "BSgenome.Hsapiens.UCSC.hg19",
    use_syn = TRUE
  )
  mt_tally$nmf_matrix[1:5, 1:5]
}

## Use strand bias categories
mt_tally <- sig_tally(
  laml,
  ref_genome = "BSgenome.Hsapiens.UCSC.hg19",
  use_syn = TRUE, add_trans_bias = TRUE
)
```

```
## Test it by enrichment analysis

```r
enrich_component_strand_bias(mt_tally$nmf_matrix)
enrich_component_strand_bias(mt_tally$all_matrices$SBS_24)
```

} else {

```r
message("Please install package 'BSgenome.Hsapiens.UCSC.hg19' firstly!")
```

} # End of main function

---

**subset.CopyNumber**

*Subsetting CopyNumber object*

**Description**

Subset data slot of `CopyNumber` object, un-selected rows will move to `dropoff.segs` slot, annotation slot will update in the same way.

**Usage**

```r
## S3 method for class 'CopyNumber'
subset(x, subset = TRUE, ...)
```

**Arguments**

- `x`: a `CopyNumber` object to be subsetted.
- `subset`: logical expression indicating rows to keep.
- `...`: further arguments to be passed to or from other methods. Useless here.

**Value**

a `CopyNumber` object

**Author(s)**

Shixiang Wang

---

**transcript.hg19**

*Merged Transcript Location at Genome Build hg19*

**Description**

Merged Transcript Location at Genome Build hg19

**Format**

A `data.table`
transcript.hg38

Source
from GENCODE release v33.

Examples
data(transcript.hg19)

transcript.hg38  Merged Transcript Location at Genome Build hg38

Description
Merged Transcript Location at Genome Build hg38

Format
A data.table

Source
from GENCODE release v33.

Examples
data(transcript.hg38)

use_color_style  Set Color Style for Plotting

Description
Set Color Style for Plotting

Usage
use_color_style(style, mode = c("SBS", "copynumber", "DBS", "ID"))

Arguments

style  one of 'default' and 'cosmic'.

mode  only used when the style is 'cosmic', can be one of "SBS", "copynumber", "DBS", "ID".

Value
color values.
Examples

use_color_style("default")
use_color_style("cosmic")
Index

*Topic `bootstrap`
  - sig_fit_bootstrap, 68

add_h_arrow, 3
add_labels, 4
centromeres.hg19, 6
centromeres.hg38, 6
cchromsize.hg19, 7
cchromsize.hg38, 7
circlize::circos.genomicHeatmap, 31
CopyNumber, 13, 26–28, 31, 34, 36, 74, 75, 78
CopyNumber (CopyNumber-class), 8
CopyNumber-class, 8
cowplot::save_plot(), 39
cyto bands.hg19, 9
cyto bands.hg38, 9
data.table::fread(), 26
enrich_component_strand_bias, 10
facet, 51
future::availableCores(), 75
geom_boxplot, 51
get_adj_p, 10
get_bayesian_result, 12, 59
get_cn_ploidy, 13
get_genome_annotation, 14
get_group_comparison, 16, 39
get_groups, 15, 38
get_sig_exposure, 18, 67, 69
get_sig_feature_association, 19, 22, 50
get_sig_feature_association(), 22
get_sig_similarity, 5, 20
get_tidy_association, 19, 20, 22, 49, 50
get_tidy_parameter, 22
ggpar, 51
ggplot2::annotate, 5
ggpubr::compare_means(), 10, 11, 40
ggpubr::ggboxplot, 44
ggpubr::ggviolin, 44
ggpubr::stat_compare_means(), 11, 40
ggpubr::stat_pvalue_manual(), 10
handle_hyper_mutation, 23
hello, 24
legend(), 38
list.files, 25
MAF, 26, 58, 74, 75
MAF (MAF-class), 24
MAF-class, 24
maftools::read.maf, 26
NMF::nmf(), 65
NMF::nmfEstimateRank, 63
NMF::predict(), 16
plot_grid, 33, 35
read_copynumber, 25, 27, 76
read_maf, 26, 26
report_bootstrap_p_value, 27
scoring, 28
show_catalogue, 30
show_cn_circos, 31
show_cn_components, 32
show_cn_distribution, 34
show_cn_features, 35
show_cn_profile, 31, 36
show_cosmic_sig_profile, 37
show_group_comparison, 39
show_group_comparison(), 16
show_group_mapping, 41
show_groups, 16, 38
show_sig_bootstrap, 42
show_sig_bootstrap_error, 45
show_sig_bootstrap_error
  (show_sig_bootstrap), 42
show_sig_bootstrap_exposure, 44, 52
show_sig_bootstrap_exposure
  (show_sig_bootstrap), 42
show_sig_bootstrap_stability, 45
show_sig_bootstrap_stability
  (show_sig_bootstrap), 42
show_sig_consensusmap, 46
show_sig_exposure, 47
show_sig_feature_corrplot, 49
show_sig_fit, 50
show_sig_number_survey, 52
show_sig_number_survey2, 53, 53
show_sig_profile, 4, 30, 37, 55
sig_attrs (sig_names), 72
sig_auto_extract, 12, 15, 18, 48, 56, 59, 61,
  64–67, 69, 73, 77
sig_convert, 61
sig_estimate, 52, 53, 60, 62, 64, 65, 77
sig_estimate(), 54
sig_exposure (sig_names), 72
sig_extract, 15, 18, 47, 48, 53, 56, 60–62,
  64, 65–67, 69, 73, 77
sig_extract(), 15, 60
sig_fit, 15, 18, 38, 45, 50–52, 66, 70, 72
sig_fit(), 38
sig_fit_bootstrap, 27, 45, 52, 67, 68, 72
sig_fit_bootstrap_batch, 43, 45, 52, 67,
  70, 71
sig_modify_names (sig_names), 72
sig_names, 72
sig_number (sig_names), 72
sig_signature (sig_names), 72
sig_tally, 10, 22, 30, 33, 35, 56, 57, 60, 65,
  73
sigminer, 58
stats::aov, 16
stats::fisher.test, 16
stats::p.adjust, 22
stats::p.adjust(), 11
stats::TukeyHSD, 16
subset.CopyNumber, 78

transcript.hg19, 78
transcript.hg38, 79

use_color_style, 79