Package 'sigminer'

April 25, 2019

Title Capture Genomic Variation Signatures using Non-Negative Matrix Factorization

Version 0.1.9

Description Contains functions for identification of copy number signatures (Geoffrey et al. (2018) <doi:10.1038/s41588-018-0179-8>) and mutation signatures (Alexandrov et al. (2018) <doi:10.1038/nature12477>) by non-negative matrix factorization, signature analysis and visualization. It can be used to capture signatures of genomic variation, compare genotype or phenotype features of different signatures and thus uncover the relationship between the mechanism of genomic variation and phenotypes in cancer.

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

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

Depends R (>= 3.5)

Imports cluster, corrplot, cowplot, data.table, doParallel, dplyr, flexmix, foreach, ggplot2, maftools, methods, NMF, RColorBrewer, tidyr, magrittr, purrr

biocViews

Suggests BSgenome.Hsapiens.UCSC.hg19, cowsay, ggpubr, knitr, rmarkdown, covr, testthat, prettydoc, pheatmap

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Shixiang Wang [aut, cre] (<https://orcid.org/0000-0001-9855-7357>), Geoffrey Macintyre [ctb], Xue-Song Liu [ctb]

Maintainer Shixiang Wang <w_shixiang@163.com>

Repository CRAN

Date/Publication 2019-04-25 11:00:03 UTC

R topics documented:

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)
```

chromsize.hg19 *chromosome size of genome build hg19*

Description

chromosome size of genome build hg19

Format

A data.frame

Source

Generate from UCSC gold path

Examples

```
data(chromsize.hg19)
```

chromsize.hg38 *chromosome size of genome build hg38*

Description

chromosome size of genome build hg38

Format

A data.frame

Source

Generate from UCSC gold path

Examples

```
data(chromsize.hg38)
```

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.

`clinical.data` data associated with each sample in copy number profile.

draw_cn_components *Plot mixture fit model components*

Description

Plot mixture fit model components

Usage

```
draw_cn_components(features, components, ...)
```

Arguments

`features`	a `list` generate from get_features or sig_prepare function.
`components`	a `list` contain flexmix object of copy-number features, obtain this from get_components function or use pre-compiled components data which come from CNV signature paper `https://www.nature.com/articles/s41588-018-0179-8` (set this argument as `NULL`).
`...`	other options pass to `plot_grid` function of **cowplot** package.

Value

a ggplot object

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

Other copy number plot: draw_cn_distribution, draw_cn_features

Examples

```
# Load copy number prepare object
load(system.file("extdata", "toy_copynumber_prepare.RData",
  package = "sigminer", mustWork = TRUE
))
draw_cn_components(cn_prepare$features, cn_prepare$components)
```

---

draw_cn_distribution *Plot copy number distribution either by length or chromosome*

Description

Visually summarize copy number distribution either by copy number segment length or chromosome. When input is a CopyNumber object, genome_build option will read from genome_build slot of object instead of using argument set in function.

Usage

```
draw_cn_distribution(data, rm_normal = TRUE, mode = c("ld", "cd"),
  fill = FALSE, scale_chr = TRUE, genome_build = c("hg19", "hg38"))
```

Arguments

data	result from get_LengthFraction function result or 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.
genome_build	genome build version, should be 'hg19' or 'hg38'.

Value

a ggplot object

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

Other copy number plot: draw_cn_components, draw_cn_features

Examples

```
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData",
  package = "sigminer", mustWork = TRUE
))
# Plot distribution
draw_cn_distribution(cn)
draw_cn_distribution(cn, mode = "cd")
draw_cn_distribution(cn, mode = "cd", fill = TRUE)
```

---

draw_cn_features *Plot copy number feature distribution*

Description

Plot copy number feature distribution

Usage

```
draw_cn_features(features, ylab = "", ...)
```

Arguments

features	a list generate from get_features or sig_prepare function.
ylab	lab of y axis.
...	other options pass to plot_grid function of cowplot package.

Value

a ggplot object

See Also

Other copy number plot: draw_cn_components, draw_cn_distribution

Examples

```
# Load copy number prepare object
load(system.file("extdata", "toy_copynumber_prepare.RData",
  package = "sigminer", mustWork = TRUE
))
draw_cn_features(cn_prepare$features)
```

`draw_sig_activity` *Plot signature activity*

Description

Currently support copy number signatures and mutational signatures.

Usage

```
draw_sig_activity(nmfObj, mode = c("copynumber", "mutation"),
  font_scale = 1, hide_samps = TRUE)
```

Arguments

`nmfObj`	a `NMF` result object which is an element return from sig_extract or run results of **NMF** package.
`mode`	variation type to decompose, currently support "copynumber" or "mutation".
`font_scale`	a number used to set font scale.
`hide_samps`	if `TRUE`, not show sample names.

Value

a `ggplot` object

Author(s)

Shixiang Wang

See Also

Other signature plot: `draw_sig_corrplot`, `draw_sig_profile`, `draw_subtypes_comparison`

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
draw_sig_activity(res$nmfObj)
```

`draw_sig_corrplot` *Plot correlation between signature activities*

Description

Plot correlation between signature activities

Usage

```
draw_sig_corrplot(mat_list, order = "original", type = "lower",
  sig.level = 0.05, ...)
```

Arguments

`mat_list`	a `list` contain correlation and p value matrix etc., obtain it from sig_get_correlation function.
`order`	Character, the ordering method of the correlation matrix. • `"original"` for original order (default). • `"AOE"` for the angular order of the eigenvectors. • `"FPC"` for the first principal component order. • `"hclust"` for the hierarchical clustering order. • `"alphabet"` for alphabetical order. See function `corrMatOrder` for details.
`type`	Character, `"full"` (default), `"upper"` or `"lower"`, display full matrix, lower triangular or upper triangular matrix.
`sig.level`	Significant level, if the p-value in `p-mat` is bigger than `sig.level`, then the corresponding correlation coefficient is regarded as insignificant. If `insig` is `"label_sig"`, this may be an increasing vector of significance levels, in which case `pch` will be used once for the highest p-value interval and multiple times (e.g. "*", "**", "***") for each lower p-value interval.
`...`	other arguments pass to `corrplot::corrplot()` function.

Author(s)

Shixiang Wang

See Also

Other signature plot: `draw_sig_activity`, `draw_sig_profile`, `draw_subtypes_comparison`

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
sig_activity <- sig_get_activity(res$nmfObj)
sig_cor <- sig_get_correlation(sig_activity)
draw_sig_corrplot(sig_cor)
```

`draw_sig_profile` *Plot signature profile*

Description

Currently support copy number signatures and mutation signatures.

Usage

```
draw_sig_profile(nmfObj, mode = c("copynumber", "mutation"),
  y_scale = c("relative", "absolute"), font_scale = 1)
```

Arguments

`nmfObj`	a `NMF` result object which is an element return from sig_extract or run results of **NMF** package.
`mode`	variation type to decompose, currently support "copynumber" or "mutation".
`y_scale`	one of 'relative' or 'absolute', if choose 'relative', signature columns will be scaled to sum as 1.
`font_scale`	a number used to set font scale.

Value

a `ggplot` object

Author(s)

Shixiang Wang

See Also

Other signature plot: `draw_sig_activity`, `draw_sig_corrplot`, `draw_subtypes_comparison`

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
draw_sig_profile(res$nmfObj)
```

`draw_subtypes_comparison` *Plot comparison between signature subtypes*

Description

Using result data from sig_summarize_subtypes function, this function plot genotypes/phenotypes comparison between signature subtypes 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()`.

Usage

```
draw_subtypes_comparison(subtype_summary, xlab = "subtype",
  ylab_co = NA, legend_title_ca = NA, show_pvalue = TRUE, ...)
```

Arguments

`subtype_summary`	a `list` from result of sig_summarize_subtypes function.
`xlab`	lab name of x axis for all plots.
`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 `subtype_summary`, the location for categorical type data should mark with `NA`.
`legend_title_ca`	legend title for plots of categorical type data. Of note, this argument should be a character vector has same length as `subtype_summary`, the location for continuous type data should mark with `NA`.
`show_pvalue`	if `TRUE`, show p value for comparison of continuous data types.
`...`	other paramters pass to `ggpubr::stat_compare_means()`.

Value

a `list` of `ggplot` objects.

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

Other signature plot: draw_sig_activity, draw_sig_corrplot, draw_sig_profile

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
# Assign samples to clusters
subtypes <- sig_assign_samples(res$nmfObj, type = "samples")

set.seed(1234)
# Add custom groups
subtypes$new_group <- sample(c("1", "2", "3", "4"), size = nrow(subtypes), replace = TRUE)
# Summarize subtypes
subtypes.sum <- sig_summarize_subtypes(subtypes[, -1],
  col_subtype = "nmf_subtypes",
  cols_to_summary = colnames(subtypes[, -1])[c(-1, -2)],
  type = c("co", "ca"), verbose = TRUE
)
draw_subtypes_comparison(subtypes.sum)
```

GenomicVariation-class

Class GenomicVariation

Description

S4 class for storing summarized genomic variation profile.

Slots

`CopyNumber` a `CopyNumber` object.

`MAF` a `MAF` object.

`clinical.data` data associated with each sample in copy number profile and MAF. This slot is intersection of `clinical.data` slot in both `CopyNumber` object and `MAF` object.

get_ArmLocation *Get chromosome arm location*

Description

Get chromosome arm location

Usage

```
get_ArmLocation(genome_build = c("hg19", "hg38"))
```

Arguments

genome_build	genome build version, should be 'hg19' or 'hg38'.

See Also

Other internal calculation function series: get_LengthFraction, get_cnlist, get_cnsummary_sample, get_components, get_features, get_matrix

Examples

```
hg19_arm <- get_ArmLocation("hg19")
hg38_arm <- get_ArmLocation("hg38")
```

get_cnlist *Extract copy number profile as list from CopyNumber object*

Description

Extract copy number profile as list from CopyNumber object

Usage

```
get_cnlist(CopyNumber)
```

Arguments

CopyNumber	a CopyNumber object.

Value

a list

Author(s)

Shixiang Wang

See Also

Other internal calculation function series: get_ArmLocation, get_LengthFraction, get_cnsummary_sample, get_components, get_features, get_matrix

Examples

```
extdata_dir <- system.file("extdata", package = "sigminer", mustWork = TRUE)
cp <- read_copynumber(extdata_dir, pattern = "txt", genome_build = "hg19")
cn_list <- get_cnlist(cp)
```

---

get_cnsummary_sample *Get summary of copy number variation per sample*

Description

Include number of CNV segments, CNA burden, number of CNV amplification segments, number of CNV deletion segments etc..

Usage

```
get_cnsummary_sample(segTab, genome_build = c("hg19", "hg38"),
  genome_measure = c("called", "wg"), min_seg_len = 1000L)
```

Arguments

segTab	a data.frame with 'chromosome', 'start', 'end' and 'segVal' and 'sample' these five ordered columns. 'chromosome' should have prefix "chr".
genome_build	genome build version, should be 'hg19' or 'hg38'.
genome_measure	default is 'called', can be 'wg' or 'called'. 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 use autosome size from genome build, this option is useful for WGS, SNP etc..
min_seg_len	minimal length of CNV segment for CNA burden calculation, default is 1000. (!NOT implement NOW!)

Details

CNA burden, a simple metric of CNA level defined as the percent of the autosomal tumor genome bearing CNAs, could be used as an informative measure of CNA.

Value

a data table

Author(s)

Shixiang Wang w_shixiang@163.com

References

Hieronymus, Haley, et al. "Copy number alteration burden predicts prostate cancer relapse." Proceedings of the National Academy of Sciences 111.30 (2014): 11139-11144.

See Also

Other internal calculation function series: get_ArmLocation, get_LengthFraction, get_cnlist, get_components, get_features, get_matrix

Examples

```
load(system.file("extdata", "example_cn_list.RData",
  package = "sigminer", mustWork = TRUE
))
segTabs <- data.table::rbindlist(tcga_segTabs, idcol = "sample")
segTabs$chromosome <- paste0("chr", segTabs$chromosome)
samp_sum <- get_cnsummary_sample(segTabs[, c(2:5, 1)])
```

---

get_components *Fit optimal number of mixture model components*

Description

Apply mixture modelling to breakdown each feature distribution into mixtures of Gaussian or mixtures of Poison distributions using **flexmix** package. The order of features is 'Segment size', 'Breakpoint count per 10 Mb', 'Length of oscillating copy-number chain', 'Breakpoint count per arm', 'Copy number change', 'Absolute copy number'.

Usage

```
get_components(CN_features, seed = 123456, min_comp = 2,
  max_comp = 10, min_prior = 0.001, model_selection = "BIC",
  nrep = 1, niter = 1000)
```

Arguments

CN_features	a list generate from get_features() function.
seed	seed number.
min_comp	minimal number of components to fit, default is 2. Can also be a vector with length 6, which apply to each feature.
max_comp	maximal number of components to fit, default is 10. Can also be a vector with length 6, which apply to each feature.
min_prior	minimal prior value, default is 0.001. Details about custom setting please refer to **flexmix** package.

model_selection
: model selection strategy, default is 'BIC'. Details about custom setting please refer to **flexmix** package.

nrep
: number of run times for each value of component, keep only the solution with maximum likelihood.

niter
: maximal number of iteration to achive converge.

Value

a `list` contain `flexmix` object of copy-number features.

Author(s)

Geoffrey Macintyre, Shixiang Wang

See Also

Other internal calculation function series: `get_ArmLocation`, `get_LengthFraction`, `get_cnlist`, `get_cnsummary_sample`, `get_features`, `get_matrix`

Examples

```
# Load copy number features
load(system.file("extdata", "toy_cn_features.RData",
  package = "sigminer", mustWork = TRUE
))
cn_components <- get_components(cn_features)
```

---

get_features *Derive copy number feature distributions*

Description

This function summarise each copy-number profile using a number of different feature distributions: sigment size, breakpoint number (per ten megabase), change-point copy-number, segment copy-number, breakpoint number (per chromosome arm), length of segments with oscilating copy-number.

Usage

```
get_features(CN_data, cores = 1, genome_build = c("hg19", "hg38"))
```

Arguments

CN_data	a list contains multiple data.frames (recommended), each data.frame stores copy-number profile for one sample with 'chromosome', 'start', 'end' and 'segVal' these four necessary columns. Of note, 'segVal' column shoule be absolute copy number values.
cores	number of compute cores to run this task. You can use parallel::detectCores() function to check how many cores you can use.
genome_build	genome build version, should be 'hg19' or 'hg38'.

Value

a list contains six copy number feature distributions.

Author(s)

Geoffrey Macintyre, Shixiang Wang

See Also

Other internal calculation function series: get_ArmLocation, get_LengthFraction, get_cnlist, get_cnsummary_sample, get_components, get_matrix

Examples

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

cn_features <- get_features(cn_list, cores = 1)
```

---

get_LengthFraction *Calculate length fraction profile of copy number*

Description

Calculate length fraction profile of copy number

Usage

```
get_LengthFraction(CN_data, genome_build = c("hg19", "hg38"),
  seg_cols = c("Chromosome", "Start.bp", "End.bp", "modal_cn"),
  samp_col = "sample")
```

Arguments

CN_data	a data.frame with 'chromosome', 'start', 'end' and 'segVal' (optinal) and 'sample' these five columns (specify column names using seg_cols and samp_cols options) or a list contains multiple data.frames, each data.frame stores copy-number profile for one sample with 'chromosome', 'start', 'end' and 'segVal' (optional) these four columns. If 'sample' column is not specified, will try using name of each data.frame.
genome_build	genome build version, should be 'hg19' or 'hg38'.
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.

Value

a data table

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

Other internal calculation function series: get_ArmLocation, get_cnlist, get_cnsummary_sample, get_components, get_features, get_matrix

Examples

```
# Load copy number list
load(system.file("extdata", "toy_cnlist.RData",
  package = "sigminer", mustWork = TRUE
))
annot <- get_LengthFraction(cn_list, seg_cols = c("chromosome", "start", "end", "segVal"))
```

---

get_matrix *Generate a sample-by-component matrix*

Description

Generate a sample-by-component matrix representing the sum of posterior probabilities of each copy-number event being assigned to each component.

Usage

```
get_matrix(CN_features, all_components = NULL, cores = 1,
  rowIter = 1000)
```

Arguments

CN_features	a list contains six copy number feature distributions, obtain this from get_features() function.
all_components	a list contain flexmix object of copy-number features, obtain this from get_components function or use pre-compiled components data which come from CNV signature paper https://www.nature.com/articles/s41588-018-0179-8 (set this parameter as NULL).
cores	number of compute cores to run this task. You can use parallel::detectCores() function to check how many cores you can use.
rowIter	step size of iteration for rows of ech CNV feature.

Value

a numeric sample-by-component matrix

Author(s)

Geoffrey Macintyre, Shixiang Wang

See Also

Other internal calculation function series: get_ArmLocation, get_LengthFraction, get_cnlist, get_cnsummary_sample, get_components, get_features

Examples

```
# Load copy number components
load(system.file("extdata", "toy_cn_components.RData",
  package = "sigminer", mustWork = TRUE
))
# Load copy number features
load(system.file("extdata", "toy_cn_features.RData",
  package = "sigminer", mustWork = TRUE
))

cn_matrix <- get_matrix(cn_features, cn_components)
```

---

hello *Say hello to users*

Description

Say hello to users

Usage

```
hello()
```

Examples

```
hello()
```

`MAF-class` *Class MAF*

Description

S4 class for storing summarized MAF.

Slots

`data` data.table of MAF file containing all non-synonymous variants.

`variants.per.sample` table containing variants per sample

`variant.type.summary` table containing variant types per sample

`variant.classification.summary` table containing variant classification per sample

`gene.summary` table containing variant classification per gene

`summary` table with basic MAF summary stats

`maf.silent` subset of main MAF containing only silent variants

`clinical.data` clinical data associated with each sample/Tumor_Sample_Barcode in MAF.

`prepare_copynumber` *Prepare nmf input matrix for copy number signature analysis*

Description

Prepare nmf input matrix for copy number signature analysis

Usage

```
prepare_copynumber(CopyNumber, reference_components = FALSE, cores = 1,
  seed = 123456, min_comp = 2, max_comp = 10, min_prior = 0.001,
  model_selection = "BIC", nrep = 1, niter = 1000, rowIter = 1000)
```

Arguments

CopyNumber	a CopyNumber object.
reference_components	default is FALSE, calculate mixture components from CopyNumber object. If set it to NULL, use pre-compiled components data which come from CNV signature paper. If can also be a list contain flexmix object of copy-number features, obtain this from get_components function.
cores	number of compute cores to run this task. You can use parallel::detectCores() function to check how many cores you can use.
seed	seed number.
min_comp	minimal number of components to fit, default is 2. Can also be a vector with length 6, which apply to each feature.
max_comp	maximal number of components to fit, default is 10. Can also be a vector with length 6, which apply to each feature.
min_prior	minimal prior value, default is 0.001. Details about custom setting please refer to **flexmix** package.
model_selection	model selection strategy, default is 'BIC'. Details about custom setting please refer to **flexmix** package.
nrep	number of run times for each value of component, keep only the solution with maximum likelihood.
niter	maximal number of iteration to achive converge.
rowIter	step size of iteration for rows of ech CNV feature.

Value

a list contains matrix for NMF input, copy number features and components.

Author(s)

Shixiang Wang

See Also

Other signature analysis prepare function series: prepare_maf

Examples

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

prepare_maf *Prepare nmf input matrix for mutational signature analysis*

Description

NMF input matrix here is trinucletiode matrix. This function calls `trinucleotideMatirx` provided by **maftools** to extract 96 mutation motifs.

Usage

```
prepare_maf(maf, ref_genome = NULL, prefix = NULL, add = TRUE,
  ignoreChr = NULL, useSyn = TRUE, fn = NULL)
```

Arguments

`maf`	an `MAF` object generated by `read.maf`
`ref_genome`	BSgenome object or name of the installed BSgenome package. Example: BSgenome.Hsapiens.UCSC.hg1 Default NULL, tries to auto-detect from installed genomes.
`prefix`	Prefix to add or remove from contig names in MAF file.
`add`	If prefix is used, default is to add prefix to contig names in MAF file. If false prefix will be removed from contig names.
`ignoreChr`	Chromsomes to ignore from analysis. e.g. chrM
`useSyn`	Logical. Whether to include synonymous variants in analysis. Defaults to TRUE
`fn`	If given writes APOBEC results to an output file with basename fn. Default NULL.

Details

Extracts immediate 5' and 3' bases flanking the mutated site and classifies them into 96 substitution classes. Requires BSgenome data packages for sequence extraction.

APOBEC Enrichment: Enrichment score is calculated using the same method described by Roberts et al.

E = (n_tcw * background_c) / (n_C * background_tcw)

where, n_tcw = number of mutations within T[C>T]W and T[C>G]W context. (W -> A or T)

n_C = number of mutated C and G

background_C and background_tcw motifs are number of C and TCW motifs occuring around +/- 20bp of each mutation.

One-sided Fisher's Exact test is performed to determine the enrichment of APOBEC tcw mutations over background.

Value

list of 2. A matrix of dimension nx96, where n is the number of samples in the MAF and a table describing APOBEC enrichment per sample.

References

Roberts SA, Lawrence MS, Klimczak LJ, et al. An APOBEC Cytidine Deaminase Mutagenesis Pattern is Widespread in Human Cancers. Nature genetics. 2013;45(9):970-976. doi:10.1038/ng.2702.

See Also

Other signature analysis prepare function series: prepare_copynumber

Examples

```
laml.maf <- system.file("extdata", "tcga_laml.maf.gz", package = "maftools")
laml <- read_maf(maf = laml.maf)
library(BSgenome.Hsapiens.UCSC.hg19)
laml.tnm <- prepare_maf(
  maf = laml, ref_genome = "BSgenome.Hsapiens.UCSC.hg19",
  prefix = "chr", add = TRUE, useSyn = TRUE
)
```

---

read_copynumber *Read absolute copy number profile*

Description

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

Usage

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

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).

use_all	default is FALSE. If True, use all columns from raw input.
min_segnum	minimal number of copy number segments within a sample.
genome_build	genome build version, should be 'hg19' or 'hg38'.
genome_measure	default is 'called', can be 'wg' or 'called'. 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 use autosome size from genome build, this option is useful for WGS, SNP etc..
clinical_data	a data.frame representing clinical data associated with each sample in copy number profile.
complement	if TRUE, complement chromosome does not show in input data with normal copy 2 and force use_all to FALSE (no matter what user input).
verbose	print extra messages.
...	other parameters pass to data.table::fread()

Value

a CopyNumber object

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

Other read genomic variation data function series: read_maf, read_variation

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, verbose = TRUE
)
```

---

read_maf *Read MAF files.*

Description

Takes tab delimited MAF (can be plain text or gz compressed) file as an input and summarizes it in various ways. Also creates oncomatrix - helpful for visualization.

Usage

```
read_maf(maf, clinicalData = NULL, removeDuplicatedVariants = TRUE,
  useAll = TRUE, gisticAllLesionsFile = NULL,
  gisticAmpGenesFile = NULL, gisticDelGenesFile = NULL,
  gisticScoresFile = NULL, cnLevel = "all", cnTable = NULL,
  isTCGA = FALSE, vc_nonSyn = NULL, 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.
clinicalData	Clinical data associated with each sample/Tumor_Sample_Barcode in MAF. Could be a text file or a data.frame. Default NULL.
removeDuplicatedVariants	removes repeated variants in a particuar sample, mapped to multiple transcripts of same Gene. See Description. Default TRUE.
useAll	logical. Whether to use all variants irrespective of values in Mutation_Status. Defaults to TRUE. If FALSE, only uses with values Somatic.
gisticAllLesionsFile	All Lesions file generated by gistic. e.g; all_lesions.conf_XX.txt, where XX is the confidence level. Default NULL.
gisticAmpGenesFile	Amplification Genes file generated by gistic. e.g; amp_genes.conf_XX.txt, where XX is the confidence level. Default NULL.
gisticDelGenesFile	Deletion Genes file generated by gistic. e.g; del_genes.conf_XX.txt, where XX is the confidence level. Default NULL.
gisticScoresFile	scores.gistic file generated by gistic. Default NULL
cnLevel	level of CN changes to use. Can be 'all', 'deep' or 'shallow'. Default uses all i.e, genes with both 'shallow' or 'deep' CN changes
cnTable	Custom copynumber data if gistic results are not available. Input file or a data.frame should contain three columns in aforementioned order with gene name, Sample name and copy number status (either 'Amp' or 'Del'). Default NULL.
isTCGA	Is input MAF file from TCGA source. If TRUE uses only first 12 characters from Tumor_Sample_Barcode.
vc_nonSyn	NULL. Provide manual list of variant classifications to be considered as non-synonymous. Rest will be considered as silent variants. Default uses Variant Classifications with High/Moderate variant consequences. http://asia.ensembl.org/Help/Glossary?id=535 "Frame_Shift_Del", "Frame_Shift_Ins", "Splice_Site", "Translation_Start_Site","Nonsense_Mutation", "Nonstop_Mutation", "In_Frame_Del","In_Frame_Ins", "Missense_Mutation"
verbose	TRUE logical. Default to be talkative and prints summary.

Details

This function takes MAF file as input and summarizes them. If copy number data is available, e.g from GISTIC, it can be provided too via arguments gisticAllLesionsFile, gisticAmpGenesFile, and gisticDelGenesFile. Copy number data can also be provided as a custom table containing Gene name, Sample name and Copy Number status.

Note that if input MAF file contains multiple affected transcripts of a variant, this function by default removes them as duplicates, while keeping single unique entry per variant per sample. If you wish to keep all of them, set removeDuplicatedVariants to FALSE.

FLAGS - If you get a note on possible FLAGS while reading MAF, it means some of the top mutated genes are fishy. These genes are often non-pathogenic and passengers, but are frequently mutated in most of the public exome studies. Examples of such genes include TTN, MUC16, etc. This note can be ignored without any harm, it's only generated as to make user aware of such genes. See references for details on FLAGS.

Value

An object of class MAF.

References

Shyr C, Tarailo-Graovac M, Gottlieb M, Lee JJ, van Karnebeek C, Wasserman WW. FLAGS, frequently mutated genes in public exomes. BMC Med Genomics 2014; 7: 64.

See Also

Other read genomic variation data function series: `read_copynumber`, `read_variation`

Other read genomic variation data function series: `read_copynumber`, `read_variation`

Examples

```
laml.maf <- system.file("extdata", "tcga_laml.maf.gz", package = "maftools")
laml <- read_maf(maf = laml.maf)
```

`read_variation` *Read genomic variation profile*

Description

Read CopyNumber and MAF object as a new S4 object GenomicVariation for uniform variation analysis. **The function is initialized to construct structure of sigminer, please dont use it for now**.

Usage

```
read_variation(copynumber, maf, clinical_data = NULL)
```

Arguments

copynumber	a CopyNumber object
maf	a MAF object
clinical_data	clinical.data data associated with each sample in copy number profile and MAF.

Value

a GenomicVariation object

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

Other read genomic variation data function series: read_copynumber, read_maf

Examples

```
# Read MAF
laml.maf <- system.file("extdata", "tcga_laml.maf.gz", package = "maftools")
laml <- read_maf(maf = laml.maf)
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData",
  package = "sigminer", mustWork = TRUE
))
# Combine as GenomicVariation object
gv <- read_variation(cn, laml)
```

---

sigminer *sigminer: Capture Genomic Variation Signatures using Non-Negative Matrix Factorization*

Description

sigminer provides functions for identification of copy number signatures (Geoffrey et al. (2018) doi: 10.1038/s4158801801798) and mutational signatures (Alexandrov et al. (2018) doi: 10.1038/nature12477) by non-negative matrix factorization, signature analysis and visualization. It can be used to capture signatures of genomic variation, compare genotype or phenotype features of different signatures and thus uncover the relationship between the mechanism of genomic variation and phenotypes in cancer.

read_ functions

Read data into R objects:

- `read_copynumber()` - read `data.frame`/files into CopyNumber object.
- `read_maf()` - read `data.frame`/file into MAF object. This is powered by `read.maf` function in **maftools** package.
- `read_variation()` - combine a CopyNumber object and a MAF object as a GenomicVariation object. **It is not useful for now**.

sig_ functions

Signature analysis for genomic variations:

- `sig_prepare()` - do preparation step of signature analysis for CopyNumber or MAF object.
- `sig_estimate()` - provides survey plot and consensus map to user for selecting the best signature number.
- `sig_extract()` - extract signatures based on specified signature number.
- `sig_assign_samples()` - assign samples to signatures. This classify samples into different subgroups based on the dominant signature.
- `sig_get_activity()` - obtain signature activity in samples.
- `sig_get_correlation()` - obtain correlation matrix between signatures activity.
- `sig_get_similarity()` - get similarity between signatures.
- `sig_summarize_subtypes()` - get summary of signature subtypes.

draw_ functions

Result visualization for copy number data:

- `draw_cn_distribution()` - plot copy number distribution either by length or chromosome.
- `draw_cn_features()` - plot copy number feature distribution.
- `draw_cn_components()` - plot mixture fit model components.

Result visualization for signature analysis:

- `draw_sig_profile()` - plot signature profile.
- `draw_sig_activity()` - plot signature activity.
- `draw_sig_corrplot()` - plot correlation between signature activities.
- `draw_subtypes_comparison()` - plot comparison between signature subtypes.

Result visualization for MAF is provide by **maftools** package, please read its vignette.

sig_assign_samples *Return sample clustering from NMF run results*

Description

One of key results from NMF decomposition is to cluster samples into different groups. This function takes NMF result (a NMF object) as input and return the membership in each cluster.

Usage

```
sig_assign_samples(nmfObj, type = "consensus", matchConseOrder = F)
```

Arguments

nmfObj	a NMF result object which is an element return from sig_extract or run results of **NMF** package.
type	cluster type, could be 'consensus' or 'samples'.
matchConseOrder	if TRUE, the result will match order as shown in consensus map when type argument is 'consensus'.

Details

X = W x H

W is the feature matrix, H is the sample matrix After NMF run, use this function to select import features and assign groups for the two matrix.

More detail please see NMF::predict().

Value

a data.table object

See Also

Other signature analysis series function: sig_estimate, sig_extract, sig_get_activity, sig_get_correlation, sig_get_similarity, sig_prepare, sig_summarize_subtypes

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
# Assign samples to clusters
subtypes <- sig_assign_samples(res$nmfObj, type = "samples")
```

`sig_estimate` *Estimate signature number*

Description

Use **NMF** package to evaluate the optimal number of signatures. Users should `library(NMF)` firstly.

Usage

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

Arguments

`nmf_matrix`	a `matrix` used for NMF decomposition (with rownames and colnames), generate from sig_prepare function.
`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.
`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).
`cores`	number of cpu cores to run NMF.
`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.
`use_random`	Should generate random data from input to test measurements. Default is `TRUE`.
`save_plots`	if `TRUE`, save plots to local machine.
`plot_basename`	when save plots, set custom basename for file path.
`method`	specification of the NMF algorithm. Use 'brunet' as default. Available methods for nmf decompositions are 'brunet', 'lee', 'ls-nmf', 'nsNMF', 'offset'.
`pConstant`	A small positive value 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

See Also

Other signature analysis series function: `sig_assign_samples`, `sig_extract`, `sig_get_activity`, `sig_get_correlation`, `sig_get_similarity`, `sig_prepare`, `sig_summarize_subtypes`

Examples

```
# Load copy number prepare object
load(system.file("extdata", "toy_copynumber_prepare.RData",
  package = "sigminer", mustWork = TRUE
))
library(NMF)
cn_estimate <- sig_estimate(cn_prepare$nmf_matrix,
  cores = 1, nrun = 5,
  verbose = TRUE
)
```

---

sig_extract *Extract variation signatures*

Description

Do NMF de-composition and then extract signatures.

Usage

```
sig_extract(nmf_matrix, n_sig, mode = c("copynumber", "mutation"),
  nrun = 10, cores = 1, method = "brunet", pConstant = NULL,
  seed = 123456)
```

Arguments

nmf_matrix	a matrix used for NMF decomposition (with rownames and colnames), generate from sig_prepare function.
n_sig	number of signature. Please run sig_prepare to select a suitable value.
mode	variation type to decompose, currently support "copynumber" or "mutation".
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.
method	specification of the NMF algorithm. Use 'brunet' as default. Available methods for nmf decompositions are 'brunet', 'lee', 'ls-nmf', 'nsNMF', 'offset'.
pConstant	A small positive value 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.

Value

a list contains NMF object, signature matrix and activity matrix.

Author(s)

Shixiang Wang

See Also

Other signature analysis series function: sig_assign_samples, sig_estimate, sig_get_activity, sig_get_correlation, sig_get_similarity, sig_prepare, sig_summarize_subtypes

Examples

```
# Load copy number prepare object
load(system.file("extdata", "toy_copynumber_prepare.RData",
  package = "sigminer", mustWork = TRUE
))
# Extract copy number signatures
res <- sig_extract(cn_prepare$nmf_matrix, 2, mode = "copynumber", nrun = 1)
```

sig_get_activity *Get signature activity*

Description

Get signature activity

Usage

```
sig_get_activity(nmfObj)
```

Arguments

nmfObj	a NMF result object which is an element return from sig_extract or run results of **NMF** package.

Value

a list

See Also

Other signature analysis series function: sig_assign_samples, sig_estimate, sig_extract, sig_get_correlation, sig_get_similarity, sig_prepare, sig_summarize_subtypes

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
# Get activity of signatures
sig_activity <- sig_get_activity(res$nmfObj)
```

sig_get_correlation *Get correlation matrix between signature activities*

Description

Compute correlation matrix and corresponding statistical test values between signature activies.

Usage

```
sig_get_correlation(cn_activity = NULL, snv_activity = NULL,
  type = c("absolute", "relative"), ...)
```

Arguments

cn_activity	activity of copy number signature, a list, obtain it from sig_get_activity function.
snv_activity	activity of mutational signature, a list, obtain it from sig_get_activity function.
type	one of 'absolute' and 'relative'.
...	other arguments pass to corrplot::cor.mtest().

Value

a list.

Author(s)

Shixiang Wang

See Also

Other signature analysis series function: sig_assign_samples, sig_estimate, sig_extract, sig_get_activity, sig_get_similarity, sig_prepare, sig_summarize_subtypes

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
# Get activity of signatures
sig_activity <- sig_get_activity(res$nmfObj)
# Get correlation matrix between signature activities
sig_cor <- sig_get_correlation(sig_activity)
```

---

sig_get_similarity *Get similarity between signatures*

Description

Obtain similarity for two signature matrix, their rows (features) must match.

Usage

```
sig_get_similarity(sig1, sig2, type = c("cos", "cor"))
```

Arguments

sig1	signature 1, result of sig_extract function or a data.frame which rows are features and columns are signatures (colnames are necessary).
sig2	signature 2, result of sig_extract function or a data.frame which rows are features and columns are signatures (colnames are necessary).
type	could be "cos" (default, cosine similarity) or "cor" (correlation).

Value

a matrix which rownames from sig1 and colnames from sig2.

Author(s)

Shixiang Wang

See Also

Other signature analysis series function: sig_assign_samples, sig_estimate, sig_extract, sig_get_activity, sig_get_correlation, sig_prepare, sig_summarize_subtypes

---

sig_prepare *Prepare variation signature analysis*

Description

Generate a matrix for NMF de-composition.

Usage

```
sig_prepare(object, ...)

## S3 method for class 'CopyNumber'
sig_prepare(object, reference_components = FALSE,
  cores = 1, seed = 123456, min_comp = 2, max_comp = 10,
  min_prior = 0.001, model_selection = "BIC", nrep = 1,
  niter = 1000, rowIter = 1000, ...)

## S3 method for class 'MAF'
sig_prepare(object, ref_genome = NULL, prefix = NULL,
  add = TRUE, ignoreChr = NULL, useSyn = TRUE, ...)

## S3 method for class 'GenomicVariation'
sig_prepare(object, ...)
```

Arguments

object	a CopyNumber object or MAF object or GenomicVariation (not support for now) object.
...	custom setting for operating object. Detail see S3 method for corresponding class (e.g. CopyNumber).
reference_components	default is FALSE, calculate mixture components from CopyNumber object. If set it to NULL, use pre-compiled components data which come from CNV signature paper. If can also be a list contain flexmix object of copy-number features, obtain this from get_components function.

cores	number of compute cores to run this task. You can use parallel::detectCores() function to check how many cores you can use.
seed	seed number.
min_comp	minimal number of components to fit, default is 2. Can also be a vector with length 6, which apply to each feature.
max_comp	maximal number of components to fit, default is 10. Can also be a vector with length 6, which apply to each feature.
min_prior	minimal prior value, default is 0.001. Details about custom setting please refer to **flexmix** package.
model_selection	model selection strategy, default is 'BIC'. Details about custom setting please refer to **flexmix** package.
nrep	number of run times for each value of component, keep only the solution with maximum likelihood.
niter	maximal number of iteration to achive converge.
rowIter	step size of iteration for rows of ech CNV feature.
ref_genome	BSgenome object or name of the installed BSgenome package. Example: BSgenome.Hsapiens.UCSC.hg1 Default NULL, tries to auto-detect from installed genomes.
prefix	Prefix to add or remove from contig names in MAF file.
add	If prefix is used, default is to add prefix to contig names in MAF file. If false prefix will be removed from contig names.
ignoreChr	Chromsomes to ignore from analysis. e.g. chrM
useSyn	Logical. Whether to include synonymous variants in analysis. Defaults to TRUE

Details

The result matrix generated further need to transpose before calling NMF if user use NMF::nmf by hand.

Value

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

Methods (by class)

- `CopyNumber`: Signature analysis prepare for CopyNumber object
- `MAF`: Signature analysis prepare for CopyNumber object
- `GenomicVariation`: Signature analysis prepare for GenomicVariation object

Author(s)

Shixiang Wang

See Also

Other signature analysis series function: sig_assign_samples, sig_estimate, sig_extract, sig_get_activity, sig_get_correlation, sig_get_similarity, sig_summarize_subtypes

Examples

```
# Load copy number object
load(system.file("extdata", "toy_copynumber.RData",
  package = "sigminer", mustWork = TRUE
))
# Prepare copy number signature analysis
cn_prepare <- sig_prepare(cn)
```

`sig_summarize_subtypes`

Get summary of signature subtypes

Description

Summarize genotypes/phenotypes based on signature subtypes. For categorical type, calculate fisher p value (using stats::fisher.test) and count table. 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 `draw_subtypes_comparison()`.

Usage

```
sig_summarize_subtypes(data, col_subtype, cols_to_summary, type = "ca",
  verbose = FALSE)
```

Arguments

`data`	a `data.frame` contains signature subtypes and genotypes/phenotypes (including categorical and continuous type data) want to analyze. User need to construct this `data.frame` by him/herself.
`col_subtype`	column name of signature subtypes.
`cols_to_summary`	column names of genotypes/phenotypes want to summarize based on subtypes.
`type`	a characater vector with length same as `cols_to_summary`, 'ca' for categorical type and 'co' for continuous type.
`verbose`	if `TRUE`, print extra information.

Value

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

Author(s)

Shixiang Wang w_shixiang@163.com

See Also

Other signature analysis series function: sig_assign_samples, sig_estimate, sig_extract, sig_get_activity, sig_get_correlation, sig_get_similarity, sig_prepare

Examples

```
# Load copy number signature
load(system.file("extdata", "toy_copynumber_signature.RData",
  package = "sigminer", mustWork = TRUE
))
# Assign samples to clusters
subtypes <- sig_assign_samples(res$nmfObj, type = "samples")

set.seed(1234)
# Add custom groups
subtypes$new_group <- sample(c("1", "2", "3", "4"), size = nrow(subtypes), replace = TRUE)
# Summarize subtypes
subtypes.sum <- sig_summarize_subtypes(subtypes[, -1],
  col_subtype = "nmf_subtypes",
  cols_to_summary = colnames(subtypes[, -1])[c(-1, -2)],
  type = c("co", "ca"), verbose = TRUE
)
```

---

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

```
## 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

`test_run_components` *Test running status for getting components*

Description

This is used when user find it is hard to determine component number by get_components function.

Usage

```
test_run_components(CN_features, feature_name = c("segsize", "bp10MB",
  "osCN", "bpchrarm", "changepoint", "copynumber"), seed = 123456,
  min_comp = 2, max_comp = 10, min_prior = 0.001,
  model_selection = "BIC", nrep = 1, niter = 1000)
```

Arguments

`CN_features`	a `list` generate from `get_features()` function.
`feature_name`	feature name to test.
`seed`	seed number.
`min_comp`	minimal number of components to fit, default is 2.
`max_comp`	maximal number of components to fit, default is 10.
`min_prior`	minimal prior value, default is 0.001. Details about custom setting please refer to **flexmix** package.
`model_selection`	model selection strategy, default is 'BIC'. Details about custom setting please refer to **flexmix** package.
`nrep`	number of run times for each value of component, keep only the solution with maximum likelihood.
`niter`	maximal number of iteration to achive converge.

Value

a `list` contain `flexmix` object of copy-number features.

Author(s)

Shixiang Wang

Examples

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

# Test component number from 2 to 3 for feature 'sigment size'
test_run_components(cn_features, feature_name = "segsize", max_comp = 3)
```

Index