Package 'YAPSA'

October 18, 2025

Type Package

Title Yet Another Package for Signature Analysis

Version 1.35.1

Date 2025-07-22

Imports limSolve, SomaticSignatures, VariantAnnotation, Seqinfo, reshape2, gridExtra, corrplot, dendextend, GetoptLong, circlize, gtrellis, doParallel, parallel, PMCMRplus, ggbeeswarm, ComplexHeatmap, KEGGREST, grDevices, Biostrings, BSgenome.Hsapiens.UCSC.hg19, magrittr, pracma, dplyr, utils

Depends R (>= 4.0.0), GenomicRanges, ggplot2, grid

Description This package provides functions and routines for supervised analyses of mutational signatures (i.e., the signatures have to be known, cf. L. Alexandrov et al., Nature 2013 and L. Alexandrov et al., Bioaxiv 2018). In particular, the family of functions LCD (LCD = linear combination decomposition) can use optimal signature-specific cutoffs which takes care of different detectability of the different signatures. Moreover, the package provides different sets of mutational signatures, including the COSMIC and PCAWG SNV signatures and the PCAWG Indel signatures; the latter infering that with YAPSA, the concept of supervised analysis of mutational signatures is extended to Indel signatures. YAPSA also provides confidence intervals as computed by profile likelihoods and can perform signature analysis on a stratified mutational catalogue (SMC = stratify mutational catalogue) in order to analyze enrichment and depletion patterns for the signatures in different strata.

License GPL-3

Suggests testthat, BiocStyle, knitr, rmarkdown

VignetteBuilder knitr

LazyLoad yes

biocViews Sequencing, DNASeq, SomaticMutation, Visualization, Clustering, GenomicVariation, StatisticalMethod, BiologicalQuestion

RoxygenNote 7.2.3

Encoding UTF-8

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

git_branch devel

git_last_commit d4187c4

git_last_commit_date 2025-07-22

Repository Bioconductor 3.22

Date/Publication 2025-10-17

Author Daniel Huebschmann [aut],
Lea Jopp-Saile [aut],
Carolin Andresen [aut],
Zuguang Gu [aut, cre],
Matthias Schlesner [aut]

Maintainer Zuguang Gu <z.gu@dkfz.de>

Contents

`add_annotation` *Add information to an annotation data structure*

Description

Function to iteratively add information to an annotation data structure as needed for `HeatmapAnnotation` and especially for `annotation_exposures_barplot`

Usage

```
add_annotation(
  in_annotation_col,
  in_annotation_df,
  in_attribution_vector,
  in_colour_vector,
  in_name
)
```

Arguments

`in_annotation_col`
: List, every element of which refers to one layer of annotation List elements are structures corresponding to named colour vectors

`in_annotation_df`
: Data frame, every column of which corresponds to a layer of annotation. It has as many rows as there are samples, every entry in a row corresponding to the attribute the samples has for the corresponding layer of annotation.The factor levels of a column of `in_annotation_df` correspond to the names of the corresponding element in `in_annotation_col`

`in_attribution_vector`
: A vector which is going to be cbinded to `in_annotatiin_df`, carrying the annotation information of the new layer to be added

`in_colour_vector`
: Named vector of colours to be attributed to the new annotation

`in_name`
: Name of the new layer of annotation

Value

A list with entries

- annotation_col: A list as in in_annotation_col but with one additional layer of annotation
- annotation_df: A data frame as in in_annotation_df but with one additional layer of annotation

Examples

```
NULL
```

`add_as_fist_to_list` *Add an element as first entry to a list*

Description

Works for all types of lists and inputs

Usage

```
add_as_fist_to_list(in_list, in_element)
```

Arguments

in_list	List to which an element is to be added
in_element	Element to be added

Value

List with input element as first entry.

Examples

```
NULL
```

`aggregate_exposures_by_category`
Aggregate exposures by category

Description

If a valid category (i.e. it matches to a category specified in in_sig_ind_df) is supplied, then the exposures are aggregated over this category.

Usage

```
aggregate_exposures_by_category(in_exposures_df, in_sig_ind_df, in_category)
```

Arguments

in_exposures_df	Input data frame of exposures.
in_sig_ind_df	Input data frame of meta information on the signatures. Has to match the signatures in in_exposures_df
in_category	Category to be aggregated over

Value

A list with entries:

- exposures: The exposures H, a numeric data frame with l rows and m columns, l being the number of aggregated signatures and m being the number of samples
- norm_exposures: The normalized exposures H, a numeric data frame with l rows and m columns, l being the number of aggregated signatures and m being the number of samples
- out_sig_ind_df: Data frame of the type signature_indices_df, i.e. indicating name, function and meta-information of the aggregated signatures..

See Also

LCD_complex_cutoff

Examples

```
NULL
```

---

annotate_intermut_dist_cohort

Annotate the intermutation distance of variants cohort-wide

Description

The function annotates the intermutational distance to a cohort wide data frame by applying annotate_intermut_dist_P to every PID-specific subfraction of the cohort wide data. Note that annotate_intermut_dist_PID calls rainfallTransform. If the PID information is missing, annotate_intermut_dist_PID is called directly for the whole input.

Usage

```
annotate_intermut_dist_cohort(
  in_dat,
  in_CHROM.field = "CHROM",
  in_POS.field = "POS",
  in_PID.field = NULL,
  in_mode = "min",
  in_verbose = FALSE
)
```

Arguments

in_dat	VRanges object, VRangesList, data frame or list of data frames which carries (at least) one column for the chromosome and one column for the position. Optionally, a column to specify the PID can be provided.
in_CHROM.field	String indicating which column of in_df carries the chromosome information
in_POS.field	String indicating which column of in_df carries the position information
in_PID.field	String indicating which column of in_df carries the PID information
in_mode	String passed through annotate_intermut_dist_PID to rainfallTransform indicating which method to choose for the computation of the intermutational distance.
in_verbose	Whether verbose or not.

Value

VRanges object, VRangesList, data frame or list of data frames identical to in_df (reordered by in_PID.field), but with the intermutation distance annotated as an additional column on the right named dist.

See Also

annotate_intermut_dist_PID

rainfallTransform

Examples

```
test_df <- data.frame(CHROM=c(1,1,1,2,2,2,3,3,3,4,4,4,5,5),
                      POS=c(1,2,4,4,6,9,1,4,8,10,20,40,100,200),
                      REF=c("C","C","C","T","T","T","A",
                            "A","A","G","G","G","N","A"),
                      ALT=c("A","G","T","A","C","G","C",
                            "G","T","A","C","T","A","N"),
                      PID=c(1,1,1,2,2,2,1,1,2,2,2,1,1,2))
test_df <- test_df[order(test_df$PID,test_df$CHROM,test_df$POS),]
min_dist_df <-
  annotate_intermut_dist_cohort(test_df,in_CHROM.field="CHROM",
                                in_POS.field="POS", in_PID.field="PID",
                                in_mode="min")
max_dist_df <-
  annotate_intermut_dist_cohort(test_df,in_CHROM.field="CHROM",
                                in_POS.field="POS", in_PID.field="PID",
                                in_mode="max")
min_dist_df
max_dist_df
```

annotate_intermut_dist_PID *Annotate the intermutation distance of variants per PID*

Description

The function annotates the intermutational distance to a PID wide data frame by applying rainfallTransform to every chromosome-specific subfraction of the PID wide data.

Usage

```
annotate_intermut_dist_PID(
  in_dat,
  in_CHROM.field = "CHROM",
  in_POS.field = "POS",
  in_mode = "min",
  in_verbose = FALSE
)
```

Arguments

in_dat	VRanges object or data frame which carries (at least) one column for the chromosome and one column for the position.
in_CHROM.field	String indicating which column of in_dat carries the chromosome information if dealing with data frames.
in_POS.field	String indicating which column of in_dat carries the position information if dealing with data frames.
in_mode	String passed to rainfallTransform indicating which method to choose for the computation of the intermutational distance.
in_verbose	Whether verbose or not.

Value

VRanges object or data frame identical to in_dat, but with the intermutation distance annotated as an additional column on the right named dist.

See Also

annotate_intermut_dist_cohort

rainfallTransform

Examples

```
test_df <- data.frame(
 CHROM=c(1,1,1,2,2,2,3,3,3,4,4,4,5,5),
 POS=c(1,2,4,4,6,9,1,4,8,10,20,40,100,200),
 REF=c("C","C","C","T","T","T","A","A","A","G","G","G","N","A"),
 ALT=c("A","G","T","A","C","G","C","G","T","A","C","T","A","N"))
min_dist_df <- annotate_intermut_dist_PID(test_df,in_CHROM.field="CHROM",
                                          in_POS.field="POS",
                                          in_mode="min")
```

```
max_dist_df <- annotate_intermut_dist_PID(test_df,in_CHROM.field="CHROM",
                                          in_POS.field="POS",
                                          in_mode="max")
min_dist_df
max_dist_df
```

---

annotation_exposures_barplot

Plot the exposures of a cohort with different layers of annotation

---

Description

The exposures H, determined by NMF or by LCD, are displayed as a stacked barplot by calling Heatmap. The x-axis displays the PIDs (patient identifier or sample), the y-axis the counts attributed to the different signatures with their respective colours per PID. It is analogous to plot_exposures. As many layers of information as desired can be added via an annotation data frame. The annotation data is handled in a way similar to annotation_heatmap_exposures. This function calls:

- rowAnnotation,
- HeatmapAnnotation and
- Heatmap

Usage

```
annotation_exposures_barplot(
  in_exposures_df,
  in_signatures_ind_df,
  in_subgroups_df,
  in_annotation_df = NULL,
  in_annotation_col = NULL,
  ylab = NULL,
  title = "",
  in_labels = FALSE,
  in_barplot_borders = TRUE,
  in_column_anno_borders = FALSE,
  in_annotation_legend_side = "right",
  in_padding = unit(c(2, 20, 2, 2), "mm"),
  in_annotation = NULL
)
```

Arguments

in_exposures_df
: Numerical data frame encoding the exposures H, i.e. which signature contributes how much to which PID (patient identifier or sample).

in_signatures_ind_df
: A data frame containing meta information about the signatures

in_subgroups_df
: A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup

in_annotation_df
: A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup for all layers of annotation

in_annotation_col
: A list indicating colour attributions for all layers of annotation

ylab
: String indicating the column name in `in_subgroups_df` to take the subgroup information from.

title
: Title for the plot to be created.

in_labels
: Whether or not to show the names of the samples.

in_barplot_borders
: Whether or not to show border lines in barplot

in_column_anno_borders
: Whether or not to draw separating lines between the fields in the annotation

in_annotation_legend_side
: Where to put the legends of the annotation df, default is right.

in_padding
: Parameter passed on to function `draw`

in_annotation
: A full annotation object may also be provided by the educated user.

Details

It might be necessary to install the newest version of the development branch of the packages **circlize** and **ComplexHeatmap** by Zuguang Gu: `devtools::install_github("jokergoo/circlize")` and `devtools::install_github("jokergoo/ComplexHeatmap")`

It might be necessary to install the newest version of the development branch of the packages **circlize** and **ComplexHeatmap** by Zuguang Gu: `devtools::install_github("jokergoo/circlize")` and `devtools::install_github("jokergoo/ComplexHeatmap")`

Value

The function doesn't return any value.

See Also

`HeatmapAnnotation`

`Heatmap`

`decorate_heatmap_body`

`annotation_heatmap_exposures`

`plot_exposures`

Examples

```
NULL
```

annotation_exposures_list_barplot *Plot the exposures of a cohort with different layers of annotation for SNV and INDEL signatures*

Description

The exposures H, determined by NMF or by LCD, are displayed as a stacked barplot by calling Heatmap. The x-axis displays the PIDs (patient identifier or sample), the y-axis the counts attributed to the different signatures with their respective colours per PID. It is analogous to plot_exposures. As many layers of information as desired can be added via an annotation data frame. The annotation data is handled in a way similar to annotation_heatmap_exposures. In comparison to annotation_exposures_barplot allows this function to deal with a list of differn signature and mutation types. This function calls:

- rowAnnotation,
- HeatmapAnnotation and
- Heatmap

Usage

```
annotation_exposures_list_barplot(
  in_exposures_list,
  in_signatures_ind_list,
  in_subgroups_list,
  in_annotation_list,
  ylab = NULL,
  title = "",
  in_indel_sigs = FALSE,
  in_labels = FALSE,
  in_barplot_borders = TRUE,
  in_column_anno_borders = FALSE,
  in_annotation_legend_side = "right",
  in_padding = unit(c(2, 20, 2, 2), "mm"),
  in_annotation = NULL
)
```

Arguments

in_exposures_list
: A list of numerical data frame encoding the exposures H of different signature types, i.e. which signature contributes how much to which PID (patient identifier or sample).

in_signatures_ind_list
: A list of data frame containing meta information about the each signature type individually

in_subgroups_list
: A list of data frame indicating of each siganture type which PID (patient or sample identifyier) belongs to which subgroup

`in_annotation_list`
: A list data frame indicating which PID (patient or sample identifyier) belongs to which subgroup for all layers of annotation and a list indicating colour attributions for all layers of annotation for each siganture type individually

`ylab`
: String indicating the column name in `in_subgroups_df` to take the subgroup information from.

`title`
: Title for the plot to be created.

`in_indel_sigs`
: Tag which is default FALSE when whole genome data are analysed the tag will be TRUE

`in_labels`
: Whether or not to show the names of the samples.

`in_barplot_borders`
: Whether or not to show border lines in barplot

`in_column_anno_borders`
: Whether or not to draw separating lines between the fields in the annotation

`in_annotation_legend_side`
: Where to put the legends of the annotation df, default is right.

`in_padding`
: Parameter passed on to function `draw`

`in_annotation`
: A full annotation object may also be provided by the educated user.

Details

It might be necessary to install the newest version of the development branch of the packages **circlize** and **ComplexHeatmap** by Zuguang Gu: `devtools::install_github("jokergoo/circlize")` and `devtools::install_github("jokergoo/ComplexHeatmap")`

It might be necessary to install the newest version of the development branch of the packages **circlize** and **ComplexHeatmap** by Zuguang Gu: `devtools::install_github("jokergoo/circlize")` and `devtools::install_github("jokergoo/ComplexHeatmap")`

Value

The function doesn't return any value.

See Also

`HeatmapAnnotation`

`Heatmap`

`decorate_heatmap_body`

`annotation_heatmap_exposures`

`plot_exposures`

Examples

```
NULL
```

annotation_heatmap_exposures *Heatmap to cluster the PIDs on their signature exposures (ComplexHeatmap)*

Description

The PIDs are clustered according to their signature exposures. The procedure is analogous to complex_heatmap_exposures, but enabling more than one annotation row for the PIDs. This function calls:

- rowAnnotation,
- HeatmapAnnotation and
- Heatmap

Usage

```
annotation_heatmap_exposures(
  in_exposures_df,
  in_annotation_df,
  in_annotation_col,
  in_signatures_ind_df,
  in_data_type = "norm exposures",
  in_method = "manhattan",
 in_palette = colorRamp2(c(0, 0.2, 0.4, 0.6), c("white", "yellow", "orange", "red")),
  in_cutoff = 0,
  in_filename = NULL,
  in_column_anno_borders = FALSE,
  in_row_anno_borders = FALSE,
  in_show_PIDs = TRUE,
  in_annotation_legend_side = "right"
)
```

Arguments

in_exposures_df	Numerical data frame encoding the exposures H, i.e. which signature contributes how much to which PID (patient identifier or sample).
in_annotation_df	A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup for all layers of annotation
in_annotation_col	A list indicating colour attributions for all layers of annotation
in_signatures_ind_df	A data frame containing meta information about the signatures, especially the asserted colour
in_data_type	Title in the figure
in_method	Method of the clustering to be supplied to dist. Can be either of: euclidean, maximum, manhattan, canberra, binary or minkowski

in_palette Palette with colours or colour codes for the heatmap. Default is `colorRamp2(c(0, 0.2, 0.4, 0.6),c('white','yellow','orange','red'))`

in_cutoff A numeric value less than 1. Signatures from within `W` with an overall exposure less than `in_cutoff` will be discarded for the clustering.

in_filename A path to save the heatmap. If none is specified, the figure will be plotted to the running environment.

in_column_anno_borders
Whether or not to draw separating lines between the fields in the annotation

in_row_anno_borders
Whether or not to draw separating lines between the fields in the annotation

in_show_PIDs Whether or not to show the PIDs on the x-axis

in_annotation_legend_side
Where to put the legends of the annotation df, default is right.

Details

One additional parameter, in_show_legend_bool_vector, indicating which legends to display, is planned but deactivated in this version of the package. In order to use this features, it will be necessary to install the newest version of the packages **circlize** and **ComplexHeatmap** by Zuguang Gu: `devtools::install_github("jokergoo/circlize")` and `devtools::install_github("jokergoo/ComplexHeatm`

Value

The function doesn't return any value.

See Also

`Heatmap`

`complex_heatmap_exposures`

Examples

```
NULL
```

---

attribute_nucleotide_exchanges

Attribute the nucleotide exchange for an SNV

Description

SNVs are grouped into 6 different categories (12/2 as reverse complements are summed over). This function defines the attribution.

Usage

```
attribute_nucleotide_exchanges(
  in_dat,
  in_REF.field = "REF",
  in_ALT.field = "ALT",
  in_verbose = FALSE
)
```

Arguments

in_dat	VRanges object or data frame which carries one column for the reference base and one column for the variant base
in_REF.field	String indicating which column of in_dat carries the reference base if dealing with data frames
in_ALT.field	String indicating which column of in_dat carries the variant base if dealing with data frames
in_verbose	Whether verbose or not.

Value

A character vector with as many rows as there are in in_dat which can be annotated (i.e. appended) to the input data frame.

Examples

```
test_df <- data.frame(
 CHROM=c(1,1,1,2,2,2,3,3,3,4,4,4,5,5),
 POS=c(1,2,3,4,5,6,1,2,3,4,5,6,7,8),
 REF=c("C","C","C","T","T","T","A","A","A","G","G","G","N","A"),
 ALT=c("A","G","T","A","C","G","C","G","T","A","C","T","A","N"))
test_df$change <- attribute_nucleotide_exchanges(
 test_df,in_REF.field = "REF",in_ALT.field = "ALT")
test_df
```

---

attribute_sequence_contex_indel

Attribution of sequence context and size for an INDEL

Description

The function is a wrapper and uses getSequenceContext to annotate the sequence context.

Usage

```
attribute_sequence_contex_indel(
  in_dat,
  in_REF.field = "REF",
  in_ALT.field = "ALT",
  in_verbose = FALSE,
  in_offsetL = 10,
  in_offsetR = 50
)
```

Arguments

in_dat	VRanges object or data frame which carries one column for the reference base and one column for the variant base
in_REF.field	String indicating which column of in_dat carries the reference base if dealing with data frames

in_ALT.field	String indicating which column of in_dat carries the variant base if dealing with data frames
in_verbose	Verbose if in_verbose=1
in_offsetL	Number of nucleotides which should be annotated downstream of the variant. Per default 10 bps are annotated
in_offsetR	Number of nucleotides which should be annotated upstream of the catiant. Per default 50 bps are annotated

Value

VRanges object or data frame with the same number rows and additional columns containing the type of INDEL (Ins = insertion and Del = deletion), the annotated sequence context of the defined length, the absolute number of exchanged nucleotides and the nucleotide exchange between in_REF.field and in_ALT.field.

Examples

```
data(GenomeOfNl_raw)
GenomeOfNl_context <- attribute_sequence_contex_indel(
                                    in_dat = head(GenomeOfNl_raw),
                                    in_REF.field = "REF",
                                    in_ALT.field = "ALT",
                                    in_verbose = FALSE,
                                    in_offsetL= 10, in_offsetR=50)
GenomeOfNl_context
```

---

attribution_of_indels *Attribution of variant into one onf the 83 INDEL categories*

Description

Each varaint is categorized into one of the 83 INDEL categories. The classification likewise to Alexandrov et al., 2018 (https://www.synapse.org/#!Synapse:syn11726616). The number of 83 features are classefied asfollowed:

1. Deletion of 1 bp C/(G) or T/(A) in a repetitive context. The context is classified into 1, 2, 3, 4, 5 or larger or equal to 6 times the same nucleotide(s).
2. Insertion of 1 bp C/(G) or T/(A) in a repetitive context. The context is classified into 0, 1, 2, 3, 4, or larger or equal to 5 times the same nucleotide(s).
3. Deletions of 2bps, 3bps, 4bps or more or equal to 5bps in a repetitive context. Each deletion is classified in a context of 1, 2, 3, 4, 5 or larger or equal to 6 times the same motif.
4. Insertion of 2 bps, 3 bps, 4 bps or more or equal to 5 bps in a repetitive context. Each deletion is classified in a context of 0, 1, 2, 3, 4 or larger or equal to 5 times the same motif.
5. Microhomology deletion of 2bps, 3bps, 4bps or more or equal to 5 bps in a partly repetitive context. The partly repetitive context is defined by motif length of minus 1 bp, 2 bps, 3 bps, 4 bps or more or equal to 5bps, which is located before and after the break-point junction of the deletion.

Usage

```
attribution_of_indels(in_dat_return = in_dat_return)
```

Arguments

in_dat_return	Data frame constucted form a vcf-like file of a whole cohort or a single-sample.The first columns are those of a standart vcf file, followed by an abitrary number of custom or defined columns. One of these can carry a PID (patient or sample identifyer) and subgroup information. Furthermore, the columns containing the sequence context and the absolute length of the INDEL as well as the INDEL type of the variant can be annotated to the vcf-like df with attribute_sequence_contex_indel. These columns are required to enable the constuction of a mutational catalog.

Value

Data frame with the same dimention as the input data frame plus an addional column with the INDEL classification number corrospondig to Alexandrov et al. 2018.

Examples

```
data(GenomeOfNl_raw)
GenomeOfNl_context <- attribute_sequence_contex_indel(in_dat =
head(GenomeOfNl_raw))
GenomeOfNl_classified <- attribution_of_indels(GenomeOfNl_context)
GenomeOfNl_classified
```

---

build_gene_list_for_pathway

Build a gene list for a given pathway name

Description

Build a gene list for a given pathway name

Usage

```
build_gene_list_for_pathway(in_string, in_organism)
```

Arguments

in_string	Name or description of the pathway
in_organism	Name of the taxon to be searched in

Value

A character vector of gene names

See Also

keggLink

keggFind

extract_names_from_gene_list

Examples

```
species <- "hsa"
gene_lists_meta_df <- data.frame(
  name=c("BER","NHEJ","MMR"),
  explanation=c("base excision repair",
                "non homologous end joining",
                "mismatch repair"))
number_of_pathways <- dim(gene_lists_meta_df)[1]
gene_lists_list <- list()
for (i in seq_len(number_of_pathways)) {
  temp_list <-
    build_gene_list_for_pathway(gene_lists_meta_df$explanation[i],
                                species)
  gene_lists_list <- c(gene_lists_list,list(temp_list))
}
gene_lists_list
```

`classify_indels` *INDEL function V1 - not compartible with AlexandrovSignatures*

Description

INDEL function V1 - not compartible with AlexandrovSignatures

Usage

```
classify_indels(
  in_df,
  in_ALT.field = "ALT",
  in_REF.field = "REF",
  in_breaks = c(-Inf, -10, -3, 0, 2, 9, Inf),
  in_labels = c("del3", "del2", "del1", "in1", "in2", "in3")
)
```

Arguments

`in_df`	Input data frame containing the variances in a vcf-like format
`in_ALT.field`	Column number for alternitve field
`in_REF.field`	Coloumn number for reference field
`in_breaks`	Handed over to function cut
`in_labels`	Handed over to function cut

Value

classVector, a factor vector of indel sizes

Examples

```
NULL
```

compare_exposures *Compares alternative exposures*

Description

Compares exposures computed by two alternative approaches for the same cohort

Usage

```
compare_exposures(in_exposures1_df, in_exposures2_df, deselect_flag = TRUE)
```

Arguments

in_exposures1_df	Numeric data frame with exposures, ideally the smaller exposure data is supplied first.
in_exposures2_df	Numeric data frame with exposures, ideally the bigger exposure data is supplied second.
deselect_flag	Wehther signatures absent in both exposure data frames should be removed.

Value

A list with entries merge_df, all_cor.coeff, all_p.value, cor.coeff_vector, p.value_vector, all_cor.test, and cor.test_list.

- merge_df: Merged molten input exposure data frames
- all_cor.coeff: Pearson correlation coefficient for all data points, i.e. taken all signatures together
- all_p.value: P-value of the Pearson test for all data points, i.e. taken all signatures together
- cor.coeff_vector: A vector of Pearson correlation coefficients evaluated for every signature independently
- p.value_vector: A vector of p-values of the Pearson tests evaluated for every signature independently
- all_cor.test: A data structure as returned by cor.test for all data points, i.e. taken all signatures together
- cor.test_list: A list of data structures as returned by cor.test, but evaluated for every signature independently

Examples

```
NULL
```

compare_expousre_sets *Compare two sets of exposures by cosine distance*

Description

Compare two sets of exposures, stored in numerical data frames H1 and H2, by computing the row-wise cosine distance

Usage

```
compare_expousre_sets(in_df_small, in_df_big, in_distance = cosineDist)
```

Arguments

in_df_small, in_df_big	Numerical data frames H1 and H2, ideally the bigger one first, both with l rows and m1 and m2 columns, l being the number of signatures and m1 and m2 being the respective numbers of samples or patient identefier of H1 and H2
in_distance	A function which computes the distance measure, default is cosineDist

Value

A list with entries distance, hierarchy_small and hierarchy_big.

- distance: A numerical data frame with the cosine distances between the columns of H1, indexing the rows, and H2, indexing the columns
- hierarchy_small: A data frame carrying the information of ranked similarity between the signatures in H2 with the signatures in H1
- hierarchy_big: A data frame carrying the information of ranked similarity between the signatures in H1 with the signatures in H2

See Also

cosineDist

Examples

```
sig_1_df <- data.frame(matrix(c(1,0,0,0,0,1,0,0,0,0,1,0),ncol=3))
names(sig_1_df) <- paste0("B",seq_len(dim(sig_1_df)[2]))
sig_2_df <- data.frame(matrix(c(1,1,0,0,0,0,1,1),ncol=2))
compare_expousre_sets(sig_1_df,sig_2_df)
```

compare_sets *Compare two sets of signatures by cosine distance*

Description

Compare two sets of signatures, stored in numerical data frames W1 and W2, by computing the column-wise cosine distance

Usage

```
compare_sets(in_df_small, in_df_big, in_distance = cosineDist)
```

Arguments

in_df_small, in_df_big	Numerical data frames W1 and W2, ideally the bigger one first, both with n rows and l1 and l2 columns, n being the number of features and l1 and l2 being the respective numbers of signatures of W1 and W2
in_distance	A function which computes the distance measure, default is cosineDist

Value

A list with entries distance, hierarchy_small and hierarchy_big.

- distance: A numerical data frame with the cosine distances between the columns of W1, indexing the rows, and W2, indexing the columns
- hierarchy_small: A data frame carrying the information of ranked similarity between the signatures in W2 with the signatures in W1
- hierarchy_big: A data frame carrying the information of ranked similarity between the signatures in W1 with the signatures in W2

See Also

cosineDist

Examples

```
sig_1_df <- data.frame(matrix(c(1,0,0,0,0,1,0,0,0,0,1,0),ncol=3))
names(sig_1_df) <- paste0("B",seq_len(dim(sig_1_df)[2]))
sig_2_df <- data.frame(matrix(c(1,1,0,0,0,0,1,1),ncol=2))
compare_sets(sig_1_df,sig_2_df)
```

compare_SMCs *Compare all strata from different stratifications*

Description

Compare all strata from different orthogonal stratification axes, i.e. othogonal SMCs by cosine similarity of signature exposures. First calls

- `make_strata_df`, then
- `plot_strata` and finally
- `make_comparison_matrix`

Usage

```
compare_SMCs(
  in_stratification_lists_list,
  in_signatures_ind_df,
  output_path,
  in_nrect = 5,
  in_attribute = ""
)
```

Arguments

`in_stratification_lists_list`	List of lists with entries from different (orthogonal) stratification axes or SMCs
`in_signatures_ind_df`	A data frame containing meta information about the signatures
`output_path`	Path to directory where the results, especially the figure produced by `corrplot` is going to be stored.
`in_nrect`	Number of clusters in the clustering procedure provided by `corrplot`
`in_attribute`	Additional string for the file name where the figure produced by `corrplot` is going to be stored.

Value

The comparison matrix of cosine similarities.

See Also

`plot_strata`

`make_comparison_matrix`

Examples

```
NULL
```

`compare_to_catalogues` *Compare one mutational catalogue to reference mutational catalogues*

Description

Compare one mutational catalogue (e.g. of one index patient) to a list of reference mutational catalogues (e.g. from the initial Alexandrov puplication) by cosine similarities

Usage

```
compare_to_catalogues(in_index_df, in_comparison_list)
```

Arguments

`in_index_df`	Data frame containing the mutational catalogue of interest
`in_comparison_list`	List of data frames (ideally named) containing the reference mutational catalogues

Value

A similarity dataframe

Examples

```
NULL
```

`complex_heatmap_exposures`
Heatmap to cluster the PIDs on their signature exposures (ComplexHeatmap)

Description

The PIDs are clustered according to their signature exposures. uses package **ComplexHeatmap** by Zuguang Gu. This function calls:

- `rowAnnotation`,
- `HeatmapAnnotation` and
- `Heatmap`

Usage

```
complex_heatmap_exposures(
  in_exposures_df,
  in_subgroups_df,
  in_signatures_ind_df,
  in_data_type = "norm exposures",
  in_method = "manhattan",
  in_subgroup_column = "subgroup",
  in_subgroup_colour_column = NULL,
 in_palette = colorRamp2(c(0, 0.2, 0.4, 0.6), c("white", "yellow", "orange", "red")),
  in_cutoff = 0,
  in_filename = NULL,
  in_column_anno_borders = FALSE,
  in_row_anno_borders = FALSE
)
```

Arguments

in_exposures_df	Numerical data frame encoding the exposures H, i.e. which signature contributes how much to which PID (patient identifier or sample).
in_subgroups_df	A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup
in_signatures_ind_df	A data frame containing meta information about the signatures, especially the asserted colour
in_data_type	Title in the figure
in_method	Method of the clustering to be supplied to dist. Can be either of: euclidean, maximum, manhattan, canberra, binary or minkowski
in_subgroup_column	Indicates the name of the column in which the subgroup information is encoded in in_subgroups_df
in_subgroup_colour_column	Indicates the name of the column in which the colour information for subgroups is encoded in in_subgroups_df. If NULL, a rainbow palette is used instead.
in_palette	Palette with colours for the heatmap. Default is colorRamp2(c(0, 0.2, 0.4, 0.6), c('white','yellow','orange','red'))
in_cutoff	A numeric value less than 1. Signatures from within W with an overall exposure less than in_cutoff will be discarded for the clustering.
in_filename	A path to save the heatmap. If none is specified, the figure will be plotted to the running environment.
in_column_anno_borders	Whether or not to draw separating lines between the fields in the annotation
in_row_anno_borders	Whether or not to draw separating lines between the fields in the annotation

Details

It might be necessary to install the newest version of the development branch of the packages **circlize** and **ComplexHeatmap** by Zuguang Gu: `devtools::install_github("jokergoo/circlize")` and `devtools::install_github("jokergoo/ComplexHeatmap")`

Value

The function doesn't return any value.

See Also

`Heatmap`

Examples

```
data(lymphoma_cohort_LCD_results)
complex_heatmap_exposures(
  rel_lymphoma_Nature2013_COSMIC_cutoff_exposures_df,
  COSMIC_subgroups_df,
  chosen_signatures_indices_df,
  in_data_type="norm exposures",
  in_subgroup_colour_column="col",
  in_method="manhattan",
  in_subgroup_column="subgroup")
```

---

computeLogLik *Compute the loglikelihood*

Description

Compute the loglikelihood

Usage

```
computeLogLik(in_vector, in_pdf = NULL, verbose = FALSE)
```

Arguments

`in_vector`	Numeric vector of input values of which the loglikelihood is computed.
`in_pdf`	Probability distribution function, if NULL a normal distribution is used.
`verbose`	Verbose if `in_verbose=1`

Value

A numeric value (sum of the logarithms of the likelihoods of the input vector)

Examples

```
NULL
```

compute_comparison_stat_df *Extract statistical measures for entity comparison*

Description

Compare one mutational catalogue (e.g. of one index patient) to a list of reference mutational catalogues (e.g. from the initial Alexandrov puplication) by cosine similarities

Usage

```
compute_comparison_stat_df(in_sim_df)
```

Arguments

in_sim_df	A similarity data frame as extracted by compare_to_catalogues

Value

A dataframe containing statistical measures, prepared for bar plot

Examples

```
NULL
```

confidence_indel_calulation *Wrapper to compute confidence intervals for SNV and INDEL signatures of a cohort or single-sample*

Description

Wrapper function around confIntExp, which is applies to every signature or sample pair in a cohort. The extracted lower bound of the confidence intervals are added to the input data which is reodered and melted in order to prepare for visualization with ggplot2. The calculates of confidence intervals is based on a profiling likelihood algorithm and the wrapper calculates the data for the exposure contubution identefied with SNV and INDEL signature decompositions and application of the following cutoffs:

1. CosmicValid_absCutoffVector
2. CosmicValid_normCutoffVector
3. CosmicArtif_absCutoffVector
4. CosmicArtif_normCutoffVector
5. PCAWGValidSNV_absCutoffVector
6. PCAWGValidID_absCutoffVector

The function makes use of differnet YAPSA functions. For each of the above stated cutoff vectors a per PID decompostion of the SNV and INDEL catalog is calulated respectivly using LCD_complex_cutoff_perPID. In a next step, variateExp wich is a wrapper around confIntExp to compute confidence intervals for a cohort is used. A dataframe is returend with the upper and lower bounds of the confidence intervals. In a last step plotExposuresConfidence_indel to plot the exposures to extracted signatures including confidence intervals computed with e.g. by variateExp.

Usage

```
confidence_indel_calulation(in_current_indel_df, in_current_snv_df)
```

Arguments

in_current_indel_df	A INDEL mutational catalog. Mutational catalog can be constucted with create_indel_mutation_c
in_current_snv_df	A SNV mutational catalog. Mutational catalog can be constuced with create_mutation_catalogue_

Value

A list is returned containing 12 objects. For each cutoff data frame two corrosponding object are present. First, the p gtable object which can be used for gaphically visualization, and second a dataframe containing the corrosponding upper and lower bounds of the confidence intervals.

Examples

```
data("GenomeOfNl_MutCat")
```

---

confidence_indel_only_calulation

Wrapper to compute confidence intervals for only INDEL signatures.

Description

Wrapper function around confIntExp, which is applies to every signature or sample pair in a cohort. The extracted lower bound of the confidence intervals are added to the input data which is reodered and melted in order to prepare for visualization with ggplot2. The calculates of confidence intervals is based on a profiling likelihood algorithm and the wrapper calculates the data for the exposure contubution identefied with INDEL singature decomposition and the usage of PCAWGValidID_absCutoffVector data frame.

Usage

```
confidence_indel_only_calulation(in_current_indel_df)
```

Arguments

in_current_indel_df	A INDEL mutational catalog. Mutational catalog can be constucted with create_indel_mutation_c

Details

The function makes use of differnet YAPSA functions. For each of the above stated cutoff vectors a per PID decompostion of the SNV and INDEL catalog is calulated respectivly using LCD_complex_cutoff_perPID. In a next step, variateExp which is a wrapper around confIntExp to compute confidenceintervals for a cohort is used. A dataframe is returend with the upper and lower bounds of the confidence intervals. In a last step plotExposuresConfidence_indel to plot the exposures to extracted signatures including confidence intervals computed with e.g. by variateExp.

Value

A list is returned containing two object. First, the p gtable object which can be used for gaphically visualization, and second a dataframe containing the corrosponding upper and lower bounds of the confidence intervals.

Examples

```
data("GenomeOfNl_MutCat")
temp_list <- confidence_indel_only_calulation(
                        in_current_indel_df=MutCat_indel_df)
plot(temp_list$p_complete_PCAWG_ID)
head(temp_list$complete_PCAWG_ID)
```

---

confIntExp *Compute confidence intervals*

Description

Compute confidence intervals using the (log-)likelihood ratio test, primarily for one input sample.

Usage

```
confIntExp(
  in_ind = 1,
  in_sigLevel = 0.05,
  in_delta = 1,
  in_exposure_vector = NULL,
  in_verbose = FALSE,
  ...
)
```

Arguments

in_ind	Index of the input signature to be variated.
in_sigLevel	Significance leve (one-sided)
in_delta	Inflation parameter for the alternative model.
in_exposure_vector	Exposure vector computed for the input sample.
in_verbose	Whether to run verbose (TRUE) or not (FALSE)
...	Input parameters passed on to variateExpSingle.

Value

A list with entries

- upper: Upper bound of the confidence interval
- lower: Lower bound of the confidence interval

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
data(lymphoma_test)
data(lymphoma_cohort_LCD_results)
data(sigs)
word_length <- 3
temp_list <- create_mutation_catalogue_from_df(
  lymphoma_test_df,this_seqnames.field = "CHROM",
  this_start.field = "POS",this_end.field = "POS",
  this_PID.field = "PID",this_subgroup.field = "SUBGROUP",
  this_refGenome = BSgenome.Hsapiens.UCSC.hg19,
  this_wordLength = word_length)
lymphoma_catalogue_df <- temp_list$matrix
lymphoma_PIDs <- colnames(lymphoma_catalogue_df)
data("lymphoma_cohort_LCD_results")
lymphoma_exposures_df <-
  lymphoma_Nature2013_COSMIC_cutoff_exposures_df[, lymphoma_PIDs]
lymphoma_sigs <- rownames(lymphoma_exposures_df)
lymphoma_sig_df <- AlexCosmicValid_sig_df[, lymphoma_sigs]
confIntExp(in_ind = 1, in_sigLevel = 0.05, in_delta = 0.4,
           in_exposure_vector = lymphoma_exposures_df[, 1],
           in_catalogue_vector = lymphoma_catalogue_df[, 1],
           in_sig_df = lymphoma_sig_df)
```

---

correct_rounded *Readjust the vector to it's original norm after rounding*

Description

After use of the function round_precision the norm of the input vector may have been altered by the rounding procedure. This function restores the norm by altering only the largest entry in the rounded vector (in order to create the least possible relative error).

Usage

```
correct_rounded(x, in_interval = c(0, 1))
```

Arguments

x	vector to be rounded
in_interval	Interval

Value

The adapted form of the input vector x.

Examples

```
NULL
```

`cosineDist` *Compute the cosine distance of two vectors*

Description

Compute the cosine distance of two vectors

Usage

```
cosineDist(a, b)
```

Arguments

a, b	Numerical vectors of same length

Value

The scalar product of the two input vectors divided by the product of the norms of the two input vectors

Examples

```
## 1. Orthogonal vectors:
cosineDist(c(1,0),c(0,1))
## 2. Non-orthogonal vectors:
cosineDist(c(1,0),c(1,1))
## Compare trigonometry:
1-cos(pi/4)
```

`cosineMatchDist` *Compute an altered cosine distance of two vectors*

Description

This is an altered cosine distance: it first reduced the dimension of the two input vectors to only those coordinates where both have non-zero entries. The cosine similarity is then computed on these reduced vectors, i.e. on a sub-vector space.

Usage

```
cosineMatchDist(a, b)
```

Arguments

a, b	Numerical vectors of same length

Value

The scalar product of the reduced input vectors divided by the product of the norms of the two reduced input vectors

Examples

```
## 1. Orthogonal vectors:
cosineMatchDist(c(1,0),c(0,1))
## 2. Non-orthogonal vectors:
cosineMatchDist(c(1,0),c(1,1))
```

`create_indel_mutation_catalogue_from_df` *Wrapper to create an INDEL mutational catalog from a vlf-like data frame*

Description

From data frame constucted from a vcf-file file the function `create_indel_mutation_catalogue_from_df` creates a mutational catalog V by squencially applying the `attribute_sequence_contex_indel`, `attribute_sequence_contex_indel` and then `attribution_of_indels`. The runtime of the function is about 1 sec per 6 variants as sequence context as well as INDEL calssification are timeconsuming to compute (optimization ongoing)

Usage

```
create_indel_mutation_catalogue_from_df(
  in_dat,
  in_signature_df,
  in_REF.field = "REF",
  in_ALT.field = "ALT",
  in_verbose = FALSE
)
```

Arguments

`in_dat`	A data frame constructed from a vcf-like file of a whole cohort or single-sample. The first columns are those of a standard vcf file (`CHROM`, `POS`, `REF` and `ALT`), followed by an arbitrary number of custom or used defined columns. One of these can carry a PID (patient or sample identifyier) and one can carry subgroup information.
`in_signature_df`	A numeric data frame `W` with `n` rows and `l` columns, `n` being the number of features and `l` being the number od signatures. Data frame containing INDEL signatures which should be used to create the mutational cataologe `V`.
`in_REF.field`	String indicating which column of `in_dat` carries the reference base if dealing with data frames
`in_ALT.field`	String indicating which column of `in_dat` carries the variant base if dealing with data frames
`in_verbose`	Verbose if `in_verbose=1`

Value

A dataframe in the format of a mutational catalog V, which can be used for LCD analysis

Examples

```
data(sigs_pcawg)
data(GenomeOfNl_raw)
temp_df <- translate_to_hg19(GenomeOfNl_raw[1:200,],"CHROM")
temp_df$PID <- sample(c("PID1","PID2","PID3","PID4","PID5"),200,replace=TRUE)
temp <- create_indel_mutation_catalogue_from_df(in_dat = temp_df,
   in_signature_df = PCAWG_SP_ID_sigs_df,
   in_REF.field = "REF",
   in_ALT.field = "ALT",
   in_verbose = FALSE)
dim(temp)
head(temp)
```

---

create_indel_mut_cat_from_df

Create a Mutational catalog from a data frame

Description

This function creates a mutational catalog from a data frame. It requires the returend data frame optainted with attribution_of_indels.

Usage

```
create_indel_mut_cat_from_df(in_df, in_signatures_df)
```

Arguments

in_df	A data frame constucted from a vcf-like file of a whole cohort or single-sample. The first coloums are those of a standart vcf file, followed by an arbitrary number of customs or used defined columns. One if these can carry a PID (patient or sample identefyier) and the subgroup information. Additionaly to consuct the the mutational catalog each variant needs to be characterize into one of the 83 INDEL feature classes, which can be perfomed with attribution_of_indels
in_signatures_df	A numeric data frame W with n rows and l columns, n being the number of features and l being the number of signatures.Data frame containing INDEL signatures which should be used to create the mutational cataolog V.

Value

A count dataframe, the mutational catalog V with rownames indicating the INDELs and colnames having the PIDs

Examples

```
data(GenomeOfNl_raw)
data(sigs_pcawg)
GenomeOfNl_context <- attribute_sequence_contex_indel(in_dat =
 head(GenomeOfNl_raw))
GenomeOfNl_classified <- attribution_of_indels(GenomeOfNl_context)
GenomeOfNl_mut_cat <- create_indel_mut_cat_from_df(GenomeOfNl_classified,
 in_signatures_df=PCAWG_SP_ID_sigs_df)
```

`create_mutation_catalogue_from_df` *Create a Mutational Catalogue from a data frame*

Description

This function creates a mutational catalogue from a data frame. It is a wrapper function for `create_mutation_catalogue_from_VR`: it first creates a VRanges object from the data frame by `makeVRangesFromDataFrame` and then passes this object on to the above mentioned custom function.

Usage

```
create_mutation_catalogue_from_df(
  this_df,
  this_refGenome_Seqinfo = NULL,
  this_seqnames.field = "X.CHROM",
  this_start.field = "POS",
  this_end.field = "POS",
  this_PID.field = "PID",
  this_subgroup.field = "subgroup",
  this_refGenome,
  this_wordLength,
  this_verbose = 1,
  this_rownames = c(),
  this_adapt_rownames = 1
)
```

Arguments

`this_df`	A data frame constructed from a vcf-like file of a whole cohort. The first columns are those of a standard vcf file, followed by an arbitrary number of custom or used defined columns. One of these can carry a PID (patient or sample identifyier) and one can carry subgroup information.
`this_refGenome_Seqinfo`	A seqInfo object, referring to the reference genome used. Argument passed on to `makeGRangesFromDataFrame` and thus indirectly to `makeGRangesFromDataFrame`.
`this_seqnames.field`	Indicates the name of the column in which the chromosome is encoded
`this_start.field`	Indicates the name of the column in which the start coordinate is encoded

`this_end.field` Indicates the name of the column in which the end coordinate is encoded

`this_PID.field` Indicates the name of the column in which the PID (patient or sample identifier) is encoded

`this_subgroup.field`
Indicates the name of the column in which the subgroup information is encoded

`this_refGenome` The reference genome handed over to `create_mutation_catalogue_from_VR` and indirectly to `mutationContext` and used to extract the motif context of the variants in `in_vr`.

`this_wordLength`
The size of the motifs to be extracted by `mutationContext`

`this_verbose` Verbose if `this_verbose=1`

`this_rownames` Optional parameter to specify rownames of the mutational catalogue `V` i.e. the names of the features.

`this_adapt_rownames`
Rownames of the output `matrix` will be adapted if `this_adapt_rownames=1`

Value

A list with entries `matrix` and `frame` obtained from `create_mutation_catalogue_from_VR`:

- `matrix`: The mutational catalogue `V`
- `frame`: Additional and meta information on rownames (features), colnames (PIDs) and subgroup attribution.

See Also

`makeVRangesFromDataFrame`

`create_mutation_catalogue_from_VR`

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
data(lymphoma_test)
word_length <- 3
temp_list <- create_mutation_catalogue_from_df(
  lymphoma_test_df,this_seqnames.field = "CHROM",
  this_start.field = "POS",this_end.field = "POS",
  this_PID.field = "PID",this_subgroup.field = "SUBGROUP",
  this_refGenome = BSgenome.Hsapiens.UCSC.hg19,
  this_wordLength = word_length)
  dim(temp_list$matrix)
  head(temp_list$matrix)
```

`create_mutation_catalogue_from_VR` *Create a Mutational Catalogue from a VRanges Object*

Description

This function creates a mutational catalogue from a VRanges Object by first calling `mutationContext` to establish the motif context of the variants in the input VRanges and then calling `motifMatrix` to build the mutational catalogue `V`.

Usage

```
create_mutation_catalogue_from_VR(
  in_vr,
  in_refGenome,
  in_wordLength,
  in_PID.field = "PID",
  in_verbose = 0,
  in_rownames = c(),
  adapt_rownames = 1
)
```

Arguments

`in_vr`	A VRanges object constructed from a vcf-like file of a whole cohort. The first columns are those of a standard vcf file, followed by an arbitrary number of custom or used defined columns. One of these can carry a PID (patient or sample identifyier) and one can carry subgroup information.
`in_refGenome`	The reference genome handed over to `mutationContext` and used to extract the motif context of the variants in `in_vr`.
`in_wordLength`	The size of the motifs to be extracted by `mutationContext`
`in_PID.field`	Indicates the name of the column in which the PID (patient or sample identifier) is encoded
`in_verbose`	Verbose if `in_verbose=1`
`in_rownames`	Optional parameter to specify rownames of the mutational catalogue `V` i.e. the names of the features.
`adapt_rownames`	Rownames of the output `matrix` will be adapted if `adapt_rownames=1`

Value

A list with entries `matrix`, `frame`,

- `matrix`: The mutational catalogue `V`
- `frame`: Additional and meta information on rownames (features), colnames (PIDs) and subgroup attribution.

See Also

`mutationContext`

`motifMatrix`

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
data(lymphoma_test)
data(sigs)
word_length <- 3
temp_vr <- makeVRangesFromDataFrame(
  lymphoma_test_df,in_seqnames.field="CHROM",
  in_subgroup.field="SUBGROUP",verbose_flag=1)
temp_list <- create_mutation_catalogue_from_VR(
  temp_vr,in_refGenome=BSgenome.Hsapiens.UCSC.hg19,
  in_wordLength=word_length,in_PID.field="PID",
  in_verbose=1)
dim(temp_list$matrix)
head(temp_list$matrix)
test_list <- split(lymphoma_test_df,f=lymphoma_test_df$PID)
other_list <- list()
for(i in seq_len(length(test_list))){
  other_list[[i]] <- test_list[[i]][c(1:80),]
}
other_df <- do.call(rbind,other_list)
other_vr <- makeVRangesFromDataFrame(
  other_df,in_seqnames.field="CHROM",
  in_subgroup.field="SUBGROUP",verbose_flag=1)
other_list <- create_mutation_catalogue_from_VR(
  other_vr,in_refGenome=BSgenome.Hsapiens.UCSC.hg19,
  in_wordLength=word_length,in_PID.field="PID",
  in_verbose=1,in_rownames=rownames(AlexCosmicValid_sig_df))
dim(other_list$matrix)
head(other_list$matrix)
```

`cutoffs` *Cutoffs for a supervised analysis of mutational signatures.*

Description

Series of data frames with signature-specific cutoffs. All values represent optimal cutoffs. The optimal cutoffs were determined for different choices of parameters in the cost function of the optimization. The row index is equivalent to the ratio between costs for false negative attribution and false positive attribution. The columns correspond to the different signatures. To be used with `LCD_complex_cutoff`. There are two different sets of cutoffs one for the signatures described by Alexandrov et al.(Natue 2013) and one for the signatures dokumented in Alexandriv et al. (biorxiv 2018). The calculation of the PCAWG signature specific cutoffs was perfomed in a single-sample resolution which are both valid for whole genome and whole exome sequencing data analysis.

`cutoffCosmicValid_rel_df`: Optimal cutoffs for `AlexCosmicValid_sig_df`, i.e. COSMIC signatures, only validated, trained on relative exposures.

`cutoffCosmicArtif_rel_df`: Optimal cutoffs for `AlexCosmicArtif_sig_df`, i.e. COSMIC signatures, including artifact signatures, trained on relative exposures.

`cutoffCosmicValid_abs_df`: Optimal cutoffs for `AlexCosmicValid_sig_df`, i.e. COSMIC signatures, only validated, trained on absolute exposures.

`cutoffCosmicArtif_abs_df`: Optimal cutoffs for `AlexCosmicArtif_sig_df`, i.e. COSMIC signatures, including artifact signatures, trained on absolute exposures.

cutoffInitialValid_rel_df: Optimal cutoffs for AlexInitialValid_sig_df, i.e. initially published signatures, only validated signatures, trained on relative exposures.

cutoffInitialArtif_rel_df: Optimal cutoffs for AlexInitialArtif_sig_df, i.e. initially published signatures, including artifact signatures, trained on relative exposures.

cutoffInitialValid_abs_df: Optimal cutoffs for AlexInitialValid_sig_df, i.e. initially published signatures, only validated signatures, trained on absolute exposures.

cutoffInitialArtif_abs_df: Optimal cutoffs for AlexInitialArtif_sig_df, i.e. initially published signatures, including artifact signatures, trained on absolute exposures.

Usage

```
data(cutoffs)
```

Author(s)

Daniel Huebschmann <huebschmann.daniel@googlemail.com>

---

cutoffs_pcawg *Opt. cutoffs, PCAWG SNV signatures, including artifacts*

Description

cutoffPCAWG_SBS_WGSWES_artifPid_df: Optimal cutoffs for PCAWG_SP_SBS_sigs_Artif_df, i.e. initially published signatures,including artifact signatures, trained in a single-sample resolution.

cutoffPCAWG_SBS_WGSWES_realPid_df: Optimal cutoffs for PCAWG_SP_SBS_sigs_Real_df, i.e. initially published signatures, only validated signatures, trained in a single-sample resolution.

cutoffPCAWG_ID_WGS_Pid_df: Optimal cutoffs for PCAWG_SP_ID_sigs_df, i.e. initially published signatures, signatures, trained in a single-sample resolution.

Usage

```
data(cutoffs_pcawg)
```

Author(s)

Lea Jopp-Saile <huebschmann.daniel@googlemail.com>

`cut_breaks_as_intervals` *Wrapper for cut*

Description

In this wrapper function for the known `cut` function, the `breaks` vector need not be supplied directly, instead, for every break, an interval is supplied and the function optimizes the choice of the breakpoint by chosing a local minimum of the distribution.

Usage

```
cut_breaks_as_intervals(
  in_vector,
  in_outlier_cutoffs = c(0, 3000),
  in_cutoff_ranges_list = list(c(60, 69), c(25, 32)),
  in_labels = c("late", "intermediate", "early"),
  in_name = "",
  output_path = NULL
)
```

Arguments

`in_vector`	Vector of numerical continuously distributed input
`in_outlier_cutoffs`	Interval specifyinf the upper and lower bounds of the range to be considered
`in_cutoff_ranges_list`	List if intervals in which the cutoffs for `cut` have to be optimized.
`in_labels`	Labels assigned to the strata or factors returned
`in_name`	String specifying the name of the quantity analyzed (and plotted on the x-axis of the figure to be created).
`output_path`	Path where the figure produced by the density function should be stored if non-NULL.

Value

A list with entries `category_vector`, and `density_plot` and `cutoffs`

- `category_vector`: Factor vector of the categories or strata, of the same length as `in_vector`
- `density_plot`: Density plot produced by the density function and indication of the chosen cutoffs.
- `cutoffs`: Vector of the computed optimal cutoffs

See Also

`cut`

`density`

Examples

```
data(lymphoma_test)
lymphoma_test_df$random_norm <- rnorm(dim(lymphoma_test_df)[1])
temp_list <- cut_breaks_as_intervals(
  lymphoma_test_df$random_norm,
  in_outlier_cutoffs=c(-4,4),
  in_cutoff_ranges_list=list(c(-2.5,-1.5),c(0.5,1.5)),
  in_labels=c("small","intermediate","big"))
 temp_list$density_plot
```

`deriveSigInd_df` *Derive a signature_indices_df object*

Description

Derive a data frame of type signature_indices_df (additional information for a set of signatures) from a set of given signatures for a set of new signatures.

Usage

```
deriveSigInd_df(querySigs, subjectSigs, querySigInd = NULL, in_sort = FALSE)
```

Arguments

`querySigs`	The signatures to compare to (given signatures).
`subjectSigs`	The signatures to be compared (new signatures). Alternatively this may be a complex object of type list and contain data from different deconvolutions, each of which having a set of signaturen to be compared.
`querySigInd`	The object of type signature_indices_df (additional informatio for a set of signatures) belonging to the set of known signatures.
`in_sort`	Whether to sort or not

Value

An object of type signature_indices_df (additional informatio for a set of signatures) belonging to the set of new signatures.

See Also

`relateSigs`

Examples

```
NULL
```

disambiguateVector *Disambiguate a vector*

Description

Add numbered suffixes to redundant entries in a vector

Usage

```
disambiguateVector(in_vector)
```

Arguments

in_vector	Input vector

Value

The disambiguated vector.

Examples

```
NULL
```

enrichSigs *Compare to background distribution*

Description

Compare exposures from an analysis of mutational signatures in a cohort of interest to exposures computed in a background (e.g. the set of WES and WGS samples from Alexandrov 2013).

Usage

```
enrichSigs(in_cohort_exposures_df, in_background_exposures_df, in_sig_df)
```

Arguments

in_cohort_exposures_df	Numerical data frame of the exposures of the cohort of interest.
in_background_exposures_df	Numerical data frame of the exposures of the background.
in_sig_df	Numerical data frame encoding the mutational signatures.

Value

A data frame with counts and p-values from Fisher tests.

Examples

```
NULL
```

exampleINDEL_YAPSA *Data structures used in examples, Indel tests and the Indel signature vignette of the YAPSA package.*

Description

Data structures used in examples, Indel tests and the Indel signature vignette of the YAPSA package.

Author(s)

Daniel Huebschmann <huebschmann.daniel@googlemail.com>

References

https://www.ncbi.nlm.nih.gov/pubmed/23945592

exampleYAPSA *Test and example data*

Description

Data structures used in examples, SNV tests and the SNV signature vignette of the YAPSA package.

lymphoma_PID_df: A data frame carrying subgroup information for a subcohort of samples used in the vignette. Data in the vignette is downloaded from ftp://ftp.sanger.ac.uk/pub/cancer/AlexandrovEtAl/somatic_mutation_data/Lymphoma%20B-cell/Lymphoma%20B-cell_clean_somatic_mutations_for_signature_analysis.txt. In the file available under that link somatic point mutation calls from several samples are listed in a vcf-like format. One column encodes the sample the variant was found in. In the vignette we want to restrict the analysis to only a fraction of these involved samples. The data frame lymphoma_PID_df carries the sample identifiers (PID) as rownames and the attributed subgroup in a column called subgroup.

lymphoma_test_df: A data frame carrying point mutation calls. It represents a subset of the data stored in ftp://ftp.sanger.ac.uk/pub/cancer/AlexandrovEtAl/somatic_mutation_data/Lymphoma%20B-cell/Lymphoma%20B-cell_clean_somatic_mutations_for_signature_analysis.txt. In the file available under that link somatic point mutation calls from several samples are listed in a vcf-like format. One column encodes the sample the variant was found in. The data frame lymphoma_test_df has only the variants occuring in the sample identifiers (PIDs) 4112512, 4194218 and 4121361.

lymphoma_Nature2013_raw_df: A data frame carrying point mutation calls. It represents a subset of the data stored in ftp://ftp.sanger.ac.uk/pub/cancer/AlexandrovEtAl/somatic_mutation_data/Lymphoma%20B-cell/Lymphoma%20B-cell_clean_somatic_mutations_for_signature_analysis.txt. In the file available under that link somatic point mutation calls from several samples are listed in a vcf-like format. One column encodes the sample the variant was found in.

lymphoma_Nature2013_COSMIC_cutoff_exposures_df: Data frame with exposures for testing the plot functions. Data taken from ftp://ftp.sanger.ac.uk/pub/cancer/AlexandrovEtAl/somatic_mutation_data/Lymphoma%20B-cell/Lymphoma%20B-cell_clean_somatic_mutations_for_signature_analysis.txt.

rel_lymphoma_Nature2013_COSMIC_cutoff_exposures_df: Data frame with normalized or relative exposures for testing the plot functions. Data taken from ftp://ftp.sanger.ac.uk/pub/

cancer/AlexandrovEtAl/somatic_mutation_data/Lymphoma%20B-cell/Lymphoma%20B-cell_clean_somatic_mutations_for_signature_analysis.txt.

COSMIC_subgroups_df: Subgroup information for the data stored in lymphoma_Nature2013_COSMIC_cutoff_exposur and rel_lymphoma_Nature2013_COSMIC_cutoff_exposures_df.

chosen_AlexInitialArtif_sigInd_df: Signature information for the data stored in lymphoma_Nature2013_COSMIC_ and rel_lymphoma_Nature2013_COSMIC_cutoff_exposures_df.

chosen_signatures_indices_df: Signature information for the data stored in lymphoma_Nature2013_COSMIC_cutof and rel_lymphoma_Nature2013_COSMIC_cutoff_exposures_df.

Usage

```
data(lymphoma_PID)

data(lymphoma_test)

data(lymphoma_Nature2013_raw)

data(lymphoma_cohort_LCD_results)

data(lymphoma_cohort_LCD_results)

data(lymphoma_cohort_LCD_results)

data(lymphoma_cohort_LCD_results)

data(lymphoma_cohort_LCD_results)
```

Author(s)

Daniel Huebschmann <huebschmann.daniel@googlemail.com>

References

https://www.ncbi.nlm.nih.gov/pubmed/23945592

Examples

```
data(lymphoma_test)
head(lymphoma_test_df)
dim(lymphoma_test_df)
table(lymphoma_test_df$PID)

data(lymphoma_Nature2013_raw)
head(lymphoma_Nature2013_raw_df)
dim(lymphoma_Nature2013_raw_df)
```

exchange_colour_vector *Colours codes for displaying SNVs*

Description

Vector attributing colours to nucleotide exchanges used when displaying SNV information, e.g. in a rainfall plot.

Usage

```
data(exchange_colour_vector)
```

Value

A named character vector

Author(s)

Daniel Huebschmann <huebschmann.daniel@googlemail.com>

exome_mutCatRaw_df *Example mutational catalog for the exome vignette*

Description

`exome_mutCatRaw_df`: A data frame in the format of a SNV mutation catalog. The mutational catalog contains SNV variants from a cohort of small-cell lung cancer published by Rudin et al. (Nature Genetics 2012) which was later used in the de novo discovery analysis of mutational signatures in human cancer by Alexandrov et al. (Nature 2013).

Usage

```
data(smallCellLungCancerMutCat_NatureGenetics2012)
```

Value

A data fame in the layout of a SNV mutational catalog

References

https://www.nature.com/articles/ng.2405

Examples

```
data(smallCellLungCancerMutCat_NatureGenetics2012)
head(exome_mutCatRaw_df)
dim(exome_mutCatRaw_df)
```

`exposures_barplot` *Wrapper for enhanced_barplot*

Description

Wrapper for enhanced_barplot

Usage

```
exposures_barplot(
  in_exposures_df,
  in_signatures_ind_df = NULL,
  in_subgroups_df = NULL,
  in_sum_ind = NULL,
  in_subgroups.field = "subgroup",
  in_title = "",
  in_labels = TRUE,
  in_show_subgroups = TRUE,
  ylab = NULL,
  in_barplot_borders = TRUE,
  in_column_anno_borders = FALSE
)
```

Arguments

`in_exposures_df`	Numerical data frame encoding the exposures `H`, i.e. which signature contributes how much to which PID (patient identifier or sample).
`in_signatures_ind_df`	A data frame containing meta information about the signatures. If NULL, the colour information for the signatures is taken from a rainbow palette.
`in_subgroups_df`	A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup. If NULL, it is assumed that all PIDs belong to one common subgroup. The colour coding for the default subgroup is red.
`in_sum_ind`	Index vector influencing the order in which the PIDs are going to be displayed
`in_subgroups.field`	String indicating the column name in `in_subgroups_df` to take the subgroup information from.
`in_title`	Title for the plot to be created.
`in_labels`	Flag, if `TRUE` the PIDs are displayed on the x-axis
`in_show_subgroups`	Flag, if `TRUE` then PIDs are grouped by subgroups
`ylab`	Label of the y-axis on the plot to be generate
`in_barplot_borders`	Whether or not to show border lines in barplot
`in_column_anno_borders`	Whether or not to draw separating lines between the fields in the annotation

Value

The generated barplot - a ggplot2 plot

Examples

```
data(lymphoma_cohort_LCD_results)
exposures_barplot(lymphoma_Nature2013_COSMIC_cutoff_exposures_df,
                  chosen_signatures_indices_df,
                  COSMIC_subgroups_df)
```

`extract_names_from_gene_list` *Return gene names from gene lists*

Description

Return gene names from gene lists

Usage

```
extract_names_from_gene_list(in_KEGG_gene_list, l)
```

Arguments

`in_KEGG_gene_list`	Gene list to extract names from
`l`	Index of the gene to be extracted

Value

The gene name.

See Also

`keggGet`

`build_gene_list_for_pathway`

Examples

```
NULL
```

find_affected_PIDs *Find samples affected*

Description

Find samples affected by SNVs in a certain pathway

Usage

```
find_affected_PIDs(in_gene_list, in_gene_vector, in_PID_vector)
```

Arguments

in_gene_list	List of genes in the pathway of interest.
in_gene_vector	Character vector for genes annotated to SNVs as in vcf_like_df.
in_PID_vector	Character vector for sample names annotated to SNVs as in vcf_like_df.

Value

A character vector of the names of the affected samples

Examples

```
NULL
```

GenomeOfNl_raw *Example data for the Indel vignette*

Description

GenomeOfNl_raw: A data frame contains the gemiline varinats of the dutch population. carrying point mutation calls. It represents a subset of the data stored in ftp://ftp.sanger.ac.uk/pub/cancer/AlexandrovEtAl/somatic_mutation_data/Lymphoma%20B-cell/Lymphoma%20B-cell_clean_somatic_mutations_for_signature_analysis.txt. In the file available under that link somatic point mutation calls from several samples are listed in a vcf-like format. One column encodes the sample the variant was found in.

Usage

```
data(GenomeOfNl_raw)
```

Value

A data frame in a vcf-like format

References

release version 5 https://www.nlgenome.nl/menu/main/app-go-nl/?page_id=9

Examples

```
data(GenomeOfNl_raw)
head(GenomeOfNl_raw)
dim(GenomeOfNl_raw)
```

`getSequenceContext` *Extracts the sequence context up and downstream of a nucleotide position*

Description

Extracts the sequence context up and downstream of a nucleotide position

Usage

```
getSequenceContext(position, chr, offsetL = 10, offsetR = 50)
```

Arguments

`position`	Start position of the considered INDEL
`chr`	Chromosome of the considered INDEL
`offsetL`	Number of nucleotides downstream of `position`
`offsetR`	Number of nucleotides upstream of `position`

Value

Returns a character string containing the defined seqeunce context

Examples

```
library(Biostrings)
library(BSgenome.Hsapiens.UCSC.hg19)

sequence_context <- getSequenceContext(position = 123456789, chr = "chr12",
                                       offsetL= 10, offsetR=50)
sequence_context
```

get_extreme_PIDs — *Return those PIDs which have an extreme pattern for signature exposure*

Description

For all signatures found in a project, this function returns the sample identifiers (PIDs) with extremely high or extremely low exposures of the respective signatures.

Usage

```
get_extreme_PIDs(in_exposures_df, in_quantile = 0.03)
```

Arguments

in_exposures_df	Data frame with the signature exposures
in_quantile	Quantile for the amount of extreme PIDs to be selected.

Value

A data frame with 4 rows per signature (high PIDs, high exposures, low PIDs, low exposures); the number of columns depends on the quantile chosen.

Examples

```
data(lymphoma_cohort_LCD_results)
get_extreme_PIDs(lymphoma_Nature2013_COSMIC_cutoff_exposures_df,0.05)
```

hclust_exposures — *Cluster the PIDs according to their signature exposures*

Description

The PIDs are clustered according to their signature exposures by calling first creating a distance matrix:

- dist, then
- hclust and then
- labels_colors to colour the labels (the text) of the leaves in the dendrogram.

Typically one colour per subgroup.

Usage

```
hclust_exposures(
  in_exposures_df,
  in_subgroups_df,
  in_method = "manhattan",
  in_subgroup_column = "subgroup",
  in_palette = NULL,
  in_cutoff = 0,
  in_filename = NULL,
  in_shift_factor = 0.3,
  in_cex = 0.2,
  in_title = "",
  in_plot_flag = FALSE
)
```

Arguments

in_exposures_df	Numerical data frame encoding the exposures H, i.e. which signature contributes how much to which PID (patient identifier or sample).
in_subgroups_df	A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup
in_method	Method of the clustering to be supplied to dist. Can be either of: euclidean, maximum, manhattan, canberra, binary or minkowski
in_subgroup_column	Indicates the name of the column in which the subgroup information is encoded in in_subgroups_df
in_palette	Palette with colours or colour codes for the labels (the text) of the leaves in the dendrogram. Typically one colour per subgroup. If none is specified, a rainbow palette of the length of the number of subgroups will be used as default.
in_cutoff	A numeric value less than 1. Signatures from within W with an overall exposure less than in_cutoff will be discarded for the clustering.
in_filename	A path to save the dendrogram. If none is specified, the figure will be plotted to the running environment.
in_shift_factor	Graphical parameter to adjust figure to be created
in_cex	Graphical parameter to adjust figure to be created
in_title	Title in the figure to be created under in_filename
in_plot_flag	Whether or not to display the dendrogram

Value

A list with entries hclust and dendrogram.

- hclust: The object created by hclust
- dendrogram: The above object wrapped in as.dendrogram

See Also

hclust

dist

labels_colors

Examples

```
data(lymphoma_cohort_LCD_results)
hclust_exposures(rel_lymphoma_Nature2013_COSMIC_cutoff_exposures_df,
                 COSMIC_subgroups_df,
                 in_method="manhattan",
                 in_subgroup_column="subgroup")
```

LCD *Linear Combination Decomposition*

Description

LCD performs a mutational signatures decomposition of a given mutational catalogue V with known signatures W by solving the minimization problem $min(||W * H - V||)$ with additional constraints of non-negativity on H where W and V are known

Usage

```
LCD(in_mutation_catalogue_df, in_signatures_df, in_per_sample_cutoff = 0)
```

Arguments

in_mutation_catalogue_df	A numeric data frame V with n rows and m columns, n being the number of features and m being the number of samples
in_signatures_df	A numeric data frame W with n rows and l columns, n being the number of features and l being the number of signatures
in_per_sample_cutoff	A numeric value less than 1. Signatures from within W with an exposure per sample less than in_cutoff will be discarded.

Value

The exposures H, a numeric data frame with l rows and m columns, l being the number of signatures and m being the number of samples

See Also

lsei

Examples

```
## define raw data
W_prim <- matrix(c(1,2,3,4,5,6),ncol=2)
W_prim_df <- as.data.frame(W_prim)
W_df <- YAPSA:::normalize_df_per_dim(W_prim_df,2) # corresponds to the sigs
W <- as.matrix(W_df)
## 1. Simple case: non-negativity already in raw data
H <- matrix(c(2,5,3,6,1,9,1,2),ncol=4)
H_df <- as.data.frame(H) # corresponds to the exposures
V <- W %*% H # matrix multiplication
V_df <- as.data.frame(V) # corresponds to the mutational catalogue
exposures_df <- YAPSA:::LCD(V_df,W_df)
## 2. more complicated: raw data already contains negative elements
## define indices where sign is going to be swapped
sign_ind <- c(5,7)
## now compute the indices of the other fields in the columns affected
## by the sign change
row_ind <- sign_ind %% dim(H)[1]
temp_ind <- 2*row_ind -1
other_ind <- sign_ind + temp_ind
## alter the matrix H to yield a new mutational catalogue
H_compl <- H
H_compl[sign_ind] <- (-1)*H[sign_ind]
H_compl_df <- as.data.frame(H_compl) # corresponds to the exposures
V_compl <- W %*% H_compl # matrix multiplication
V_compl_df <- as.data.frame(V_compl) # corresponds to the mutational catalog
exposures_df <- YAPSA:::LCD(V_compl_df,W_df)
exposures <- as.matrix(exposures_df)
```

LCD_complex_cutoff *LCD with a signature-specific cutoff on exposures*

Description

LCD_cutoff performs a mutational signatures decomposition by Linear Combination Decomposition (LCD) of a given mutational catalogue V with known signatures W by solving the minimization problem $min(||W * H - V||)$ with additional constraints of non-negativity on H where W and V are known, but excludes signatures with an overall contribution less than a given signature-specific cutoff (and thereby accounting for a background model) over the whole cohort.

LCD_complex_cutoff_perPID is a wrapper for LCD_complex_cutoff and runs individually for every PID.

LCD_extractCohort_callPerPID runs LCD_complex_cutoff and takes the identified signatures as input for LCD_complex_cutoff_perPID.

LCD_complex_cutoff_consensus calls LCD_complex_cutoff_combined AND LCD_complex_cutoff_perPID and makes a consensus signature call set.

LCD_complex_cutoff_combined is a wrapper for LCD_complex_cutoff, LCD_complex_cutoff_perPID, LCD_complex_cutoff_consensus AND LCD_extractCohort_callPerPID.

Usage

```
LCD_complex_cutoff(
  in_mutation_catalogue_df,
  in_signatures_df,
  in_cutoff_vector = NULL,
  in_filename = NULL,
  in_method = "abs",
  in_per_sample_cutoff = 0,
  in_rescale = TRUE,
  in_sig_ind_df = NULL,
  in_cat_list = NULL
)

LCD_complex_cutoff_perPID(
  in_mutation_catalogue_df,
  in_signatures_df,
  in_cutoff_vector = NULL,
  in_filename = NULL,
  in_method = "abs",
  in_rescale = TRUE,
  in_sig_ind_df = NULL,
  in_cat_list = NULL,
  minimumNumberOfAlterations = 25
)

LCD_extractCohort_callPerPID(
  in_mutation_catalogue_df,
  in_signatures_df,
  in_cutoff_vector = NULL,
  in_filename = NULL,
  in_method = "abs",
  in_rescale = TRUE,
  in_sig_ind_df = NULL,
  in_cat_list = NULL,
  in_verbose = FALSE,
  minimumNumberOfAlterations = 25,
  cutoff_type = "adaptive"
)

LCD_complex_cutoff_consensus(
  in_mutation_catalogue_df = NULL,
  in_signatures_df = NULL,
  in_cutoff_vector = NULL,
  in_filename = NULL,
  in_method = "abs",
  in_rescale = TRUE,
  in_sig_ind_df = NULL,
  in_cat_list = NULL,
  in_cohort_LCDlist = NULL,
  in_perPID_LCDlist = NULL,
  addSigs_cohort_cutoff = 0.25,
  addSigs_perPID_cutoff = 0.25,
```

```
  addSigs_relAbs_cutoff = 0.01,
  keep.unassigned = FALSE,
  keep.all.cohort.sigs = TRUE,
  in_verbose = FALSE,
  minimumNumberOfAlterations = 25
)

LCD_complex_cutoff_combined(
  in_mutation_catalogue_df = NULL,
  in_signatures_df = NULL,
  in_cutoff_vector = NULL,
  in_filename = NULL,
  in_method = "abs",
  in_rescale = TRUE,
  in_sig_ind_df = NULL,
  in_cat_list = NULL,
  addSigs_cohort_cutoff = 0.25,
  addSigs_perPID_cutoff = 0.25,
  addSigs_relAbs_cutoff = 0.01,
  keep.all.cohort.sigs = TRUE,
  in_verbose = FALSE,
  minimumNumberOfAlterations = 25,
  cutoff_type = "adaptive"
)
```

Arguments

in_mutation_catalogue_df	A numeric data frame V with n rows and m columns, n being the number of features and m being the number of samples
in_signatures_df	A numeric data frame W with n rows and l columns, n being the number of features and l being the number of signatures
in_cutoff_vector	A numeric vector of values less than 1. Signatures from within W with an overall exposure less than the respective value in in_cutoff_vector will be discarded.
in_filename	A path to generate a histogram of the signature exposures if non-NULL
in_method	Indicate to which data the cutoff shall be applied: absolute exposures, relative exposures
in_per_sample_cutoff	A numeric value less than 1. Signatures from within W with an exposure per sample less than in_cutoff will be discarded.
in_rescale	Boolean, if TRUE (default) the exposures are rescaled such that colSums over exposures match colSums over mutational catalogue
in_sig_ind_df	Data frame of type signature_indices_df, i.e. indicating name, function and meta-information of the signatures. Default is NULL.
in_cat_list	List of categories for aggregation. Have to be among the column names of in_sig_ind_df. Default is NULL.
minimumNumberOfAlterations	The perPID part of the analysis issues a warning if one sample has less mutations than this minimum cutoff.

in_verbose
: Verbose if in_verbose=1

cutoff_type
: If chosen to be "adaptive", the default, then signature-specific cutoffs are used for the the per-PID analysis in LCD_extractCohort_callPerPID, otherwise, no cutoffs are used.

in_cohort_LCDlist
: Optional, if not provided, the cohort-wide exposures are recalculated by calling LCD_complex_cutoff

in_perPID_LCDlist
: Optional, if not provided, the per sample exposures are recalculated by calling LCD_complex_cutoff_perPID

addSigs_cohort_cutoff
: Numeric value for a cutoff: signatures which are detected in a fraction of the samples of the cohort greater than this cutoff are kept for the consensus set of signatures

addSigs_perPID_cutoff
: Numeric value for a cutoff: signatures which are detected in one sample with exposure greater than this cutoff are kept for the consensus set of signatures

addSigs_relAbs_cutoff
: Numeric value for a cutoff: signatures which are detected with at least this fraction of all variants cohort wide are kept for the consensus set of signatures

keep.unassigned
: Boolean, if TRUE the exposures from the signatures which don't fulfill the criteria to be kept will be added and stored in the exposures as "unassigned", otherwise the exposures are rescaled.

keep.all.cohort.sigs
: If TRUE (default), all signatures extracted cohort wide are kept, if FALSE, the function reevaluates whether the signatures extracted cohort wide still fulfill their criteria (i.e. exposures > cutoff) after perPID extraction.

Value

A list with entries:

- exposures: The exposures H, a numeric data frame with l rows and m columns, l being the number of signatures and m being the number of samples
- norm_exposures: The normalized exposures H, a numeric data frame with l rows and m columns, l being the number of signatures and m being the number of samples
- signatures: The reduced signatures that have exposures bigger than in_cutoff
- choice: Index vector of the reduced signatures in the input signatures
- order: Order vector of the signatures by exposure
- residual_catalogue: Numerical data frame (matrix) of the difference between fit (product of signatures and exposures) and input mutational catalogue
- rss: Residual sum of squares (i.e. sum of squares of the residual catalogue)
- cosDist_fit_orig_per_matrix: Cosine distance between the fit (product of signatures and exposures) and input mutational catalogue computed after putting the matrix into vector format (i.e. one scaler product for the whole matrix)
- cosDist_fit_orig_per_col: Cosine distance between the fit (product of signatures and exposures) and input mutational catalogue computed per column (i.e. per sample, i.e. as many scaler products as there are samples in the cohort)

- sum_ind: Decreasing order of mutational loads based on the input mutational catalogue
- out_sig_ind_df: Data frame of the type signature_indices_df, i.e. indicating name, function and meta-information of the signatures. Default is NULL, non-NULL only if in_sig_ind_df is non-NULL.
- aggregate_exposures_list: List of exposure data frames aggregated over different categories. Default is NULL, non-NULL only if in_sig_ind_df and in_cat_list are non-NULL and if the categories specified in in_cat_list are among the column names of in_sig_ind_df.

See Also

LCD

aggregate_exposures_by_category

lsei

Examples

```
NULL
```

LCD_SMC *CD stratification analysis*

Description

CD stratification analysis

Usage

```
LCD_SMC(in_mutation_sub_catalogue_list, in_signatures_df, in_F_df = NULL)
```

Arguments

in_mutation_sub_catalogue_list	A list of s stratified mutational catalogues Vi \(numeric data frames\) with n rows and m columns each, n being the number of features and m being the number of samples. This list is naturally provided in run_SMC.
in_signatures_df	A numeric data frame W with n rows and l columns, n being the number of features and l being the number of signatures
in_F_df	Default NULL

Value

Returns a list with all exposures and the stratified ones

`logLikelihood` *Compute a loglikelihood ratio test*

Description

Compute a likelihood ratio test based on the loglikelihoods of the residuals of two different models of the same data.

Usage

```
logLikelihood(
  in_1,
  in_2,
  df_1 = NULL,
  df_2 = NULL,
  in_pdf = NULL,
  verbose = FALSE
)
```

Arguments

`in_1`	Residuals of model 1 of the input data.
`in_2`	Residuals of model 2 of the input data.
`df_1`	Degrees of freedom of the input model 1. If either `df_1` or `df_2` is NULL, the difference between the degrees of freedom of the two models is assumed to be 1.
`df_2`	Degrees of freedom of the input model 2. If either `df_1` or `df_2` is NULL, the difference between the degrees of freedom of the two models is assumed to be 1.
`in_pdf`	Probability distribution function, passed on to computeLogLik, if NULL a normal distribution is used.
`verbose`	Verbose if `in_verbose=1`

Value

A list with entries

- `statistic`: The test statistic
- `delta_df`: The difference in degrees of freedom between input model 1 and 2
- `p.value`: p value of the statistical test.

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
data(lymphoma_test)
data(sigs)
data(cutoffs)
word_length <- 3
temp_list <- create_mutation_catalogue_from_df(
  lymphoma_test_df,this_seqnames.field = "CHROM",
```

```
  this_start.field = "POS",this_end.field = "POS",
  this_PID.field = "PID",this_subgroup.field = "SUBGROUP",
  this_refGenome = BSgenome.Hsapiens.UCSC.hg19,
  this_wordLength = word_length)
lymphoma_catalogue_df <- temp_list$matrix
lymphoma_PIDs <- colnames(lymphoma_catalogue_df)
current_sig_df <- AlexCosmicValid_sig_df
current_sigInd_df <- AlexCosmicValid_sigInd_df
current_cutoff_vector <- cutoffCosmicValid_rel_df[6, ]
iniLCDList <- LCD_complex_cutoff(
  in_mutation_catalogue_df = lymphoma_catalogue_df[, 1, drop = FALSE],
  in_signatures_df = current_sig_df,
  in_cutoff_vector = current_cutoff_vector,
  in_method = "relative", in_rescale = TRUE,
  in_sig_ind_df = current_sigInd_df)
current_sig_df <- AlexCosmicValid_sig_df[, -9]
current_sigInd_df <- AlexCosmicValid_sigInd_df[-9,]
current_cutoff_vector <- cutoffCosmicValid_rel_df[6, -9]
redLCDList <- LCD_complex_cutoff(
  in_mutation_catalogue_df = lymphoma_catalogue_df[, 1, drop = FALSE],
  in_signatures_df = current_sig_df,
  in_cutoff_vector = current_cutoff_vector,
  in_method = "relative", in_rescale = TRUE,
  in_sig_ind_df = current_sigInd_df)
logLikelihood(iniLCDList, redLCDList)
```

`lymphomaNature2013_mutCat_df`

Example mutational catalog for the SNV vignette

Description

`lymphomaNature2013_mutCat_df`: A data frame in the format of a SNV mutation catalog. The mutational catalog contains SNV variants from the `lymphoma_Nature2013_raw_df` data. Mutational catalog was created with `create_mutation_catalogue_from_df` function.

Usage

```
data(lymphomaNature2013_mutCat_df)
```

Value

A data fame in the layout of a SNV mutational catalog

References

ftp://ftp.sanger.ac.uk/pub/cancer/AlexandrovEtAl/somatic_mutation_data/Lymphoma%20B-cell/Lymphoma%20B-cell_clean_somatic_mutations_for_signature_analysis.txt

Examples

```
data(lymphomaNature2013_mutCat_df)
head(lymphomaNature2013_mutCat_df)
dim(lymphomaNature2013_mutCat_df)
```

`makeVRangesFromDataFrame` *Construct a VRanges Object from a data frame*

Description

In this package, big data frames are generated from cohort wide vcf-like files. This function constructs a VRanges object from such a data frame by using `makeGRangesFromDataFrame` from the package `GenomicRanges`

Usage

```
makeVRangesFromDataFrame(
  in_df,
  in_keep.extra.columns = TRUE,
  in_seqinfo = NULL,
  in_seqnames.field = "X.CHROM",
  in_start.field = "POS",
  in_end.field = "POS",
  in_PID.field = "PID",
  in_subgroup.field = "subgroup",
  in_strand.field = "strand",
  verbose_flag = 1
)
```

Arguments

`in_df`	A big dataframe constructed from a vcf-like file of a whole cohort. The first columns are those of a standard vcf file, followed by an arbitrary number of custom or user defined columns. One of these can carry a PID (patient or sample identifyier) and one can carry subgroup information.
`in_keep.extra.columns`	in_seqinfo Argument passed on to `makeGRangesFromDataFrame`
`in_seqinfo`	A seqInfo object, referring to the reference genome used. Argument passed on to `makeGRangesFromDataFrame`
`in_seqnames.field`	Indicates the name of the column in which the chromosome is encoded
`in_start.field`	Indicates the name of the column in which the start coordinate is encoded
`in_end.field`	Indicates the name of the column in which the end coordinate is encoded
`in_PID.field`	Indicates the name of the column in which the PID (patient or sample identifier) is encoded
`in_subgroup.field`	Indicates the name of the column in which the subgroup information is encoded

in_strand.field
: Indicates the name of the column in which the strandedness is encoded

verbose_flag
: Verbose if 1

Value

The constructed VRanges object

See Also

makeGRangesFromDataFrame

Examples

```
data(lymphoma_test)
temp_vr <- makeVRangesFromDataFrame(lymphoma_test_df,
                                    in_seqnames.field="CHROM",
                                    in_subgroup.field="SUBGROUP",
                                    verbose_flag=1)
```

---

make_catalogue_strata_df

Group strata from different stratification axes

Description

For a comparison of the strata from different orthogonal stratification axes, i.e. othogonal SMCs, the strata have to be grouped and reformatted. This function does this task for the comparison by cosine similarity of mutational catalogues. Output of this function is the basis for applying make_comparison_matrix. It is called by the wrapper function run_comparison_catalogues.

Usage

```
make_catalogue_strata_df(
  in_stratification_lists_list,
  in_additional_stratum = NULL
)
```

Arguments

in_stratification_lists_list
: List of lists with entries from different (orthogonal) stratification axes or SMCs

in_additional_stratum
: Include an additionally supplied stratum in comparison in non-NULL.

Value

A list with entries strata_df, number_of_SMCs, number_of_strata.

- strata_df: Pasted numerical data frame of all strata (these are going to be compared e.g. by make_comparison_matrix).
- number_of_SMCs: Number of orthogonal stratifications in in_stratification_lists_list and additional ones.
- number_of_strata: Cumulative number of strata (sum over the numbers of strata of the different stratifications in in_stratification_lists_list) and additional ones.

See Also

plot_strata

make_comparison_matrix

run_comparison_catalogues

Examples

```
NULL
```

---

make_comparison_matrix

Compute a similarity matrix for different strata

Description

Compute and plot a similarity matrix for different strata from different stratification axes together. First, compare_sets is called on in_strata_df with itself, yielding a distance matrix (a numerical data frame) dist_df of the strata. The corresponding similarity matrix 1-dif_df is then passed to corrplot.

Usage

```
make_comparison_matrix(
  in_strata_df,
  output_path = NULL,
  in_nrect = 5,
  in_attribute = "",
  in_palette = NULL
)
```

Arguments

in_strata_df	Numerical data frame of all strata to be compared.
output_path	Path to directory where the results, especially the figure produced by corrplot is going to be stored.
in_nrect	Number of clusters in the clustering procedure provided by corrplot
in_attribute	Additional string for the file name where the figure produced by corrplot is going to be stored.
in_palette	Colour palette for the matrix

Value

The comparison matrix of cosine similarities.

See Also

compare_SMCs

Examples

```
data(sigs)
make_comparison_matrix(
 AlexCosmicValid_sig_df,in_nrect=9,
 in_palette=colorRampPalette(c("blue","green","red"))(n=100))
```

---

make_strata_df *Group strata from different stratification axes*

Description

For a comparison of the strata from different orthogonal stratification axes, i.e. othogonal SMCs, the strata have to be grouped and reformatted. This function does this task for the comparison by cosine similarity of signature exposures. Output of this function is the basis for applying plot_strata and make_comparison_matrix. It is called by the wrapper functions compare_SMCs, run_plot_strata_general or run_comparison_general.

Usage

```
make_strata_df(
  in_stratification_lists_list,
  in_remove_signature_ind = NULL,
  in_additional_stratum = NULL
)
```

Arguments

in_stratification_lists_list	List of lists with entries from different (orthogonal) stratification axes or SMCs
in_remove_signature_ind	Omit one of the signatures in in_signatures_ind_df for the comparison if non-NULL. The parameter specifies the index of the signature to be removed.
in_additional_stratum	Include an additionally supplied stratum in comparison in non-NULL.

Value

A list with entries strata_df, number_of_SMCs, number_of_strata.

- strata_df: Pasted numerical data frame of all strata (these are going to be compared e.g. by make_comparison_matrix).
- number_of_SMCs: Number of orthogonal stratifications in in_stratification_lists_list and additional ones.

- number_of_strata: Cumulative number of strata (sum over the numbers of strata of the different stratifications in in_stratification_lists_list) and additional ones.

See Also

plot_strata

make_comparison_matrix

compare_SMCs

run_plot_strata_general

run_comparison_general

Examples

```
NULL
```

---

make_subgroups_df *Make a custom data structure for subgroups*

Description

Creates a data frame carrying the subgroup information and the order in which the PIDs have to be displayed. Calls aggregate on in_vcf_like_df.

Usage

```
make_subgroups_df(
  in_vcf_like_df,
  in_exposures_df = NULL,
  in_palette = NULL,
  in_subgroup.field = "SUBGROUP",
  in_PID.field = "PID",
  in_verbose = FALSE
)
```

Arguments

in_vcf_like_df	vcf-like data frame with point mutation calls
in_exposures_df	Data frame with the signature exposures
in_palette	Palette for colour attribution to the subgroups if nun-NULL
in_subgroup.field	String indicating which column of in_vcf_like_df carries the subgroup information
in_PID.field	String indicating which column of in_vcf_like_df and of in_exposures_df carries the PID information
in_verbose	Whether verbose or not.

Value

subgroups_df: A data frame carrying the subgroup and rank information.

See Also

aggregate

Examples

```
data(lymphoma_test)
data(lymphoma_cohort_LCD_results)
choice_ind <- (names(lymphoma_Nature2013_COSMIC_cutoff_exposures_df)
               %in% unique(lymphoma_test_df$PID))
lymphoma_test_exposures_df <-
  lymphoma_Nature2013_COSMIC_cutoff_exposures_df[,choice_ind]
make_subgroups_df(lymphoma_test_df,lymphoma_test_exposures_df)
```

---

melt_exposures *Generically melts exposure data frames*

Description

Melt an exposure data frame with signatures as ID variables.

Usage

```
melt_exposures(in_df)
```

Arguments

in_df	Numeric data frame with exposures.

Value

A data frame with the molten exposures.

Examples

```
NULL
```

merge_exposures *Merge exposure data frames*

Description

Merges with the special feature of preserving the signatures and signature order.

Usage

```
merge_exposures(in_exposures_list, in_signatures_df)
```

Arguments

in_exposures_list	List of data frames (carrying information on exposures).
in_signatures_df	Data frame W in which the columns represent the signatures.

Value

A data frame with the merged exposures.

Examples

```
NULL
```

MutCat_indel_df *Example mutational catalog for the Indel vignette*

Description

MutCat_indel_df: A data frame in the format of a mutation catalog. The mutational catalog contains Indel variants from the GenomeOfNl_raw data. Variants were random sampled for 15 artificial patient for the purpose to have a Indel mutational catalog and have to show the functionality of the package. The results of the mutational catalog should not be interpreted fot they biological relevance. Mutational catalog was created with create_indel_mutation_catalogue_from_df function.

Usage

```
data(GenomeOfNl_MutCat)
```

Value

A data fame in the layout of a Indel mutational catalog

References

Mutational catalog created form release version 5 of the Genome of NL https://www.nlgenome.nl/menu/main/app-go-nl/?page_id=9

Examples

```
data(GenomeOfNl_MutCat)
head(MutCat_indel_df)
dim(MutCat_indel_df)
```

`normalizeMotifs_otherRownames` *Normalize Somatic Motifs with different rownames*

Description

This is a wrapper function to `normalizeMotifs`. The rownames are first transformed to fit the convention of the `SomaticSignatures` package and then passed on to the above mentioned function.

Usage

```
normalizeMotifs_otherRownames(in_matrix, in_norms, adjust_counts = TRUE)
```

Arguments

`in_matrix`, `in_norms`	Arguments to `normalizeMotifs`
`adjust_counts`	Whether to rescale the counts after adaption or not. Default is true.

Value

The matrix returned by `normalizeMotifs`, but with rownames transformed back to the convention of the input

Examples

```
NULL
```

`normalize_df_per_dim` *Useful functions on data frames*

Description

`normalize_df_per_dim`: Normalization is carried out by dividing by `rowSums` or `colSums`; for rows with `rowSums=0` or columns with `colSums=0`, the normalization is left out.

`average_over_present`: If averaging over columns, zero rows (i.e. those with `rowSums=0`) are left out, if averaging over rows, zero columns (i.e. those with `colSums=0`) are left out.

`sd_over_present`: If computing the standard deviation over columns, zero rows (i.e. those with `rowSums=0`) are left out, if computing the standard deviation over rows, zero columns (i.e. those with `colSums=0`) are left out.

`stderrmean_over_present`: If computing the standard error of the mean over columns, zero rows (i.e. those with `rowSums=0`) are left out, if computing the standard error of the mean over rows, zero columns (i.e. those with `colSums=0`) are left out. Uses the function `stderrmean`

Usage

```
normalize_df_per_dim(in_df, in_dimension)

average_over_present(in_df, in_dimension)

sd_over_present(in_df, in_dimension)

stderrmean_over_present(in_df, in_dimension)
```

Arguments

in_df	Data frame to be normalized
in_dimension	Dimension along which the operation will be carried out

Value

The normalized numerical data frame (normalize_df_per_dim)

A vector of the means (average_over_present)

A vector of the standard deviations (sd_over_present)

A vector of the standard errors of the mean (stderrmean_over_present)

See Also

stderrmean

Examples

```
test_df <- data.frame(matrix(c(1,2,3,0,5,2,3,4,0,6,0,0,0,0,0,4,5,6,0,7),
                             ncol=4))
## 1. Normalize over rows:
normalize_df_per_dim(test_df,1)
## 2. Normalize over columns:
normalize_df_per_dim(test_df,2)

test_df <- data.frame(matrix(c(1,2,3,0,5,2,3,4,0,6,0,0,0,0,0,4,5,6,0,7),
                             ncol=4))
## 1. Average over non-zero rows:
average_over_present(test_df,1)
## 2. Average over non-zero columns:
average_over_present(test_df,2)

test_df <- data.frame(matrix(c(1,2,3,0,5,2,3,4,0,6,0,0,0,0,0,4,5,6,0,7),
                             ncol=4))
## 1. Compute standard deviation over non-zero rows:
sd_over_present(test_df,1)
## 2. Compute standard deviation over non-zero columns:
sd_over_present(test_df,2)

test_df <- data.frame(matrix(c(1,2,3,0,5,2,3,4,0,6,0,0,0,0,0,4,5,6,0,7),
                             ncol=4))
## 1. Compute standard deviation over non-zero rows:
stderrmean_over_present(test_df,1)
## 2. Compute standard deviation over non-zero columns:
stderrmean_over_present(test_df,2)
```

plotExchangeSpectra *Plot the spectra of nucleotide exchanges*

Description

Plots the spectra of nucleotide exchanges in their triplet contexts. If several columns are present in the input data frame, the spectra are plotted for every column separately.

Usage

```
plotExchangeSpectra(
  in_catalogue_df,
  in_colour_vector = NULL,
  in_show_triplets = FALSE,
  in_show_axis_title = FALSE,
  in_scales = "free_x",
  in_refLine = NULL,
  in_refAlpha = 0.5,
  in_background = NULL
)
```

Arguments

in_catalogue_df	Numerical data frame encoding the exchange spectra to be displayed, either a mutational catalogue V or a signatures matrix W.
in_colour_vector	Specifies the colours of the 6 nucleotide exchanges if non-null.
in_show_triplets	Whether or not to show the triplets on the x-axis
in_show_axis_title	Whether or not to show the name of the y-axis
in_scales	Argument passed on to facet_grid
in_refLine	If non-null, value on the y-axis at which a horizontal line is to be drawn
in_refAlpha	Transparency of the horizontal line if it is to be drawn
in_background	Option to provide a background theme, e.g. theme_grey

Value

The generated barplot - a ggplot2 plot

See Also

geom_bar

facet_grid

Examples

```
NULL
```

plotExchangeSpectra_indel *Plot the spectra of nucleotide exchanges of INDELs*

Description

Plots the spectra of nucelotides in their triplet contexts. If several columns are present in the input data frame, the spectra are ploted for every column seperatly. The function is only suitable for a INDEL spectra and for SNV representation the funtion `plotExchangeSpectra` should be used.

Usage

```
plotExchangeSpectra_indel(
  in_catalogue_df,
  in_colour_vector = NULL,
  in_show_indel = FALSE,
  in_show_axis_title = FALSE,
  in_scales = "free_x",
  in_refLine = NULL,
  in_refAlpha = 0.5,
  in_background = NULL
)
```

Arguments

`in_catalogue_df`	Numerical data frame encoding the exchange spectra to be displayed, either a mutational catalogue `V` or a signatures matrix `W`
`in_colour_vector`	Specifies the colours of the INDELs if non-null
`in_show_indel`	Whether or not to show the INDEL names on the x-axis
`in_show_axis_title`	Whether or not to show the name of the y-axis
`in_scales`	Argument passed on to `facet_grid`
`in_refLine`	If non-null, value on the y-axis at which a horizontal line is to be drawn
`in_refAlpha`	Transparency of the horizontal line if it is to be drawn
`in_background`	Option to provide a background theme, e.g. `theme_grey`

Value

The generated barplot - a ggplot2 plot

Examples

```
data(sigs_pcawg)
plotExchangeSpectra_indel(PCAWG_SP_ID_sigs_df[,c(6,8)])
```

plotExposuresConfidence

Plot exposures including confidence intervals

Description

Plot the exposures to extracted signatures including confidence intervals computed e.g. by variateExp.

Usage

```
plotExposuresConfidence(in_complete_df, in_subgroups_df, in_sigInd_df)
```

Arguments

in_complete_df	Melted numeric input data frame e.g. as computed by variateExp
in_subgroups_df	Data frame containing meta information on subgroup attribution of the samples in the cohort of interest.
in_sigInd_df	Data frame with meta information on the signatures used in the analysis.

Value

The function doesn't return any value but plots instead.

Examples

```
NULL
```

plotExposuresConfidence_indel

Plot exposures including confidence intervals for exposures of SNVs and INDELs

Description

Plot the exposures to extracted signatures including the confidence intervals computed e.g. by variateExp

Usage

```
plotExposuresConfidence_indel(in_complete_df, in_subgroups_df, in_sigInd_df)
```

Arguments

in_complete_df	Melted numeric input data frame e.g. as computed by variateExp
in_subgroups_df	Data frame containing meta information on subgroup attribution of the samples in the cohort of interest.
in_sigInd_df	Data frame with meta information on the signatures used in the analysis.

Value

The function returns a gtable object which can be plotted with `plot` or `grid.draw`

Examples

```
NULL
```

`plot_exposures` *Plot the exposures of a cohort*

Description

`plot_exposures`: The exposures `H`, determined by NMF or by `LCD`, are displayed as a stacked barplot by calling

- `geom_bar` and optionally
- `geom_text`.

The x-axis displays the PIDs (patient identifier or sample), the y-axis the counts attributed to the different signatures with their respective colours per PID. Is called by `plot_relative_exposures`.

`plot_relative_exposures`: Plot the relative or normalized exposures of a cohort. This function first normalizes its input and then sends the normalized data to `plot_exposures`.

Usage

```
plot_exposures(
  in_exposures_df,
  in_signatures_ind_df,
  in_subgroups_df = NULL,
  in_sum_ind = NULL,
  in_subgroups.field = "subgroup",
  in_title = "",
  in_labels = TRUE,
  in_show_subgroups = TRUE,
  legend_height = 10
)

plot_relative_exposures(
  in_exposures_df,
  in_signatures_ind_df,
  in_subgroups_df,
  in_sum_ind = NULL,
  in_subgroups.field = "subgroup",
  in_title = "",
  in_labels = TRUE,
  in_show_subgroups = TRUE
)
```

Arguments

in_exposures_df	Numerical data frame encoding the exposures H, i.e. which signature contributes how much to which PID (patient identifier or sample).
in_signatures_ind_df	A data frame containing meta information about the signatures
in_subgroups_df	A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup
in_sum_ind	Index vector influencing the order in which the PIDs are going to be displayed
in_subgroups.field	String indicating the column name in in_subgroups_df to take the subgroup information from.
in_title	Title for the plot to be created.
in_labels	Flag, if TRUE the PIDs are displayed on the x-axis
in_show_subgroups	Flag, if TRUE then PIDs are grouped by subgroups
legend_height	How many signatures should be displayed in one column together at most.

Value

The generated barplot - a ggplot2 plot

See Also

LCD

geom_bar

geom_text

Examples

```
data(lymphoma_cohort_LCD_results)
plot_exposures(lymphoma_Nature2013_COSMIC_cutoff_exposures_df,
               chosen_signatures_indices_df,
               COSMIC_subgroups_df)

data(lymphoma_cohort_LCD_results)
plot_relative_exposures(lymphoma_Nature2013_COSMIC_cutoff_exposures_df,
                        chosen_signatures_indices_df,
                        COSMIC_subgroups_df)
```

plot_SMC *Plot results of the Stratification of a Mutational Catalogue*

Description

Plot a big composite figure with 3 columns: in the left column the per-PID absolute exposures will be shown, in the middle column the per_PID relative or normalized exposures will be shown, in the right column the cohort-wide exposures are shown (averaged over PIDs).

Usage

```
plot_SMC(
  number_of_strata,
  output_path,
  decomposition_method,
  number_of_sigs,
  name_list,
  exposures_strata_list,
  this_signatures_ind_df,
  this_subgroups_df,
  in_strata_order_ind,
  exposures_both_rel_df_list,
  cohort_method_flag,
  fig_width = 1200,
  fig_height = 900,
  fig_type = "png",
  in_label_orientation = "turn",
  this_sum_ind = NULL
)
```

Arguments

`number_of_strata`
: Number of strata as deduced from `link{SMC}`

`output_path`
: Path to file where the results are going to be stored. If NULL, the results will be plotted to the running environment.

`decomposition_method`
: String for the filename of the generated barplot.

`number_of_sigs`
: Number of signatures

`name_list`
: Names of the contructed strata.

`exposures_strata_list`
: The list of `s` strata specific exposures Hi, all are numerical data frames with `l` rows and `m` columns, `l` being the number of signatures and `m` being the number of samples

`this_signatures_ind_df`
: A data frame containing meta information about the signatures

`this_subgroups_df`
: A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup

in_strata_order_ind
: Index vector defining reordering of the strata

exposures_both_rel_df_list
: A list of s strata specific cohortwide (i.e. averaged over cohort) normalized exposures

cohort_method_flag
: Either or several of c("all_PIDs","cohort","norm_PIDs"), representing alternative ways to average over the cohort.

fig_width
: Width of the figure to be plotted

fig_height
: Height of the figure to be plotted

fig_type
: png or pdf

in_label_orientation
: Whether or not to turn the labels on the x-axis.

this_sum_ind
: Optional set of indices for reordering the PIDs

Value

The function doesn't return any value.

Examples

```
NULL
```

---

plot_strata *Plot all strata from different stratification axes together*

Description

Plot the cohort wide signature exposures of all strata from different stratification axes together. Naturally called by compare_SMCs.

Usage

```
plot_strata(
  in_strata_list,
  in_signatures_ind_df,
  output_path = NULL,
  in_attribute = ""
)
```

Arguments

in_strata_list
: Data structure created by make_strata_df or make_catalogue_strata_df in which the strata from different orthogonal stratification axes are reorganized in a consistent structure.

in_signatures_ind_df
: A data frame containing meta information about the signatures

output_path
: Path to directory where the results, especially the figure produced, are going to be stored.

in_attribute
: Additional string for the file name where the figure output is going to be stored.

Value

The function doesn't return any value.

See Also

compare_SMCs

Examples

```
NULL
```

---

read_entry *Read a single vcf-like file into a single data frame*

Description

Note: this function uses read.csv to read vcf-like files into data frames for single samples. As it uses read.csv, the default value for comment.char is "" and not "#" as it would have been for read.table.

Usage

```
read_entry(
  current_ind,
  in_list,
  header = TRUE,
  in_header = NULL,
  variant_type = "SNV",
  delete.char = NULL,
  ...
)

read_list(in_list, in_parallel = FALSE, header = TRUE, in_header = NULL, ...)
```

Arguments

current_ind	Index of the file to read from the list provided below.
in_list	List of paths to vcf-like file to be read. The list may be named.
header	Boolean whether a header information should be read (as in read.table)
in_header	Vector of column names to be substituted if non-NULL.
variant_type	Default is "SNV" and provides additional plausibility and checks, omitted if other string
delete.char	Character to be deleted, e.g. in order to discriminate between comment lines and header lines, if non-NULL
...	Parameters passed on to read.table
in_parallel	If multicore functionality is provided on a compute cluster, this option may be set to TRUE in order to enhance speed.

Value

A vcf-like data frame

A list with entries:

- `vcf_like_df_list`: List of the read data frames
- `readVcf_time`: Object of class `proc_time`, which stores the time needed for reading in the data

Examples

```
NULL
```

```
NULL
```

`relateSigs` *Make unique assignments between sets of signatures*

Description

Make unique assignments between a set of given signatures and a set of new signatures.

Usage

```
relateSigs(querySigs, subjectSigs)
```

Arguments

`querySigs`	The signatures to compare to (given signatures).
`subjectSigs`	The signatures to be compared (new signatures).

Value

A list of comparison vectors

See Also

`compare_sets`

`disambiguateVector`

Examples

```
NULL
```

repeat_df *Create a data frame with default values*

Description

Create a data frame with default values

Usage

```
repeat_df(in_value, in_rows, in_cols)
```

Arguments

in_value	Default entry to be repeated in the data frame
in_rows, in_cols	Dimensions of the data frame to be created

Value

The created data frame

Examples

```
## 1. Initialize with numeric value:
repeat_df(1,2,3)
## 2. Initialize with NA value:
repeat_df(NA,3,2)
## 3. Initialize with character:
repeat_df("a",4,3)
```

round_precision *Round to a defined precision*

Description

This function is an extension with regard to the function round from base R as it allows not only digits as precision, but can also round to a user-specified precision. The interval in which the rounding operation is to be carried out also can be specified by the user (default is the unit interval). Alternatively, breaks can be provided.

Usage

```
round_precision(x, breaks = NULL, in_precision = 0.05, in_interval = c(0, 1))
```

Arguments

x	Vector to be rounded
breaks	The breaks used for rounding. Default NULL
in_precision	Precition default 0.05
in_interval	Interval needs to be larger than the precision value

Value

A list with two entries:

- `values`: the rounded vector
- `breaks`: the breaks used for rounding

Examples

```
NULL
```

`run_annotate_vcf_pl` *Wrapper function to annotate addition information*

Description

Wrapper function to the perl script annotate_vcf.pl which annotates data of a track stored in file_B (may be different formats) to called variants stored in a vcf-like file_A.

Usage

```
run_annotate_vcf_pl(
  in_data_file,
  in_anno_track_file,
  in_new_column_name,
  out_file,
  in_data_file_type = "custom",
  in_anno_track_file_type = "bed",
  in_data_CHROM.field = "CHROM",
  in_data_POS.field = "POS",
  in_data_END.field = "POS"
)
```

Arguments

`in_data_file`	Path to the input vcf-like file to be annotated
`in_anno_track_file`	Path to the input file containing the annotation track
`in_new_column_name`	String indicating the name of the column to be created for annotation.
`out_file`	Path where the created files can be stored.
`in_data_file_type`	`custom` for vcf-like
`in_anno_track_file_type`	Type of the file `in_anno_track_file` containing the annotation track.
`in_data_CHROM.field`	String indicating which column of `in_data_file` contains the chromosome information.

in_data_POS.field
: String indicating which column of in_data_file contains the position information.

in_data_END.field
: String indicating which column of in_data_file contains the end information if regions are considered.

Value

Return zero if no problems occur.

Examples

```
NULL
```

---

run_comparison_catalogues

Compare all strata from different stratifications

Description

Compare all strata from different orthogonal stratification axes, i.e. othogonal SMCs by cosine similarity of mutational catalogues. Function similar to run_comparison_general. First calls

- make_catalogue_strata_df, then
- make_comparison_matrix

Usage

```
run_comparison_catalogues(
  in_stratification_lists_list,
  output_path = NULL,
  in_nrect = 5,
  in_attribute = ""
)
```

Arguments

in_stratification_lists_list
: List of lists with entries from different (orthogonal) stratification axes or SMCs

output_path
: Path to directory where the results, especially the figure produced by corrplot is going to be stored.

in_nrect
: Number of clusters in the clustering procedure provided by corrplot

in_attribute
: Additional string for the file name where the figure produced by

Value

The comparison matrix of cosine similarities.

See Also

make_comparison_matrix

run_comparison_general

Examples

```
NULL
```

`run_comparison_general` *Compare all strata from different stratifications*

Description

Compare all strata from different orthogonal stratification axes, i.e. othogonal SMCs by cosine similarity of signature exposures. Function similar to `compare_SMCs`, but without calling `plot_strata`. First calls

- `make_strata_df`, then
- `make_comparison_matrix`

Usage

```
run_comparison_general(
  in_stratification_lists_list,
  output_path = NULL,
  in_nrect = 5,
  in_attribute = "",
  in_remove_signature_ind = NULL,
  in_additional_stratum = NULL
)
```

Arguments

`in_stratification_lists_list`	List of lists with entries from different (orthogonal) stratification axes or SMCs
`output_path`	Path to directory where the results, especially the figure produced by `corrplot` is going to be stored.
`in_nrect`	Number of clusters in the clustering procedure provided by `corrplot`
`in_attribute`	Additional string for the file name where the figure produced by `corrplot` is going to be stored.
`in_remove_signature_ind`	Omit one of the signatures in `in_signatures_ind_df` for the comparison if non-NULL. The parameter specifies the index of the signature to be removed.
`in_additional_stratum`	Include an additionally supplied stratum in comparison in non-NULL.

Value

The comparison matrix of cosine similarities.

See Also

make_comparison_matrix

compare_SMCs

run_comparison_catalogues

Examples

```
NULL
```

---

run_kmer_frequency_correction

Provide comprehensive correction factors for kmer content

Description

This function is analogous to normalizeMotifs. If an analysis of mutational signatures is performed on e.g. Whole Exome Sequencing (WES) data, the signatures and exposures have to be adapted to the potentially different kmer (trinucleotide) content of the target capture. The present function takes as arguments paths to the used reference genome and target capture file. It the extracts the sequence of the target capture by calling `bedtools getfasta` on the system command prompt. `run_kmer_frequency_normalization` then calls a custom made perl script `kmer_frequencies.pl` also included in this package to count the occurences of the tripletts in both the whole reference genome and the created target capture sequence. These counts are used for normalization as in normalizeMotifs. Note that kmerFrequency provides a solution to approximate kmer frequencies by random sampling. As opposed to that approach, the function described here deterministically counts all occurences of the kmers in the respective genome.

Usage

```
run_kmer_frequency_correction(
  in_ref_genome_fasta,
  in_target_capture_bed,
  in_word_length,
  project_folder,
  target_capture_fasta = "targetCapture.fa",
  in_verbose = 1
)
```

Arguments

`in_ref_genome_fasta`
: Path to the reference genome fasta file used.

`in_target_capture_bed`
: Path to a bed file containing the information on the used target capture. May also be a compressed bed.

in_word_length	Integer number defining the length of the features or motifs, e.g. 3 for tripletts or 5 for pentamers
project_folder	Path where the created files, especially the fasta file with the sequence of the target capture and the count matrices, can be stored.
target_capture_fasta	Name of the fasta file of the target capture to be created if not yet existent.
in_verbose	Verbose if in_verbose=1

Value

A list with 2 entries:

- rel_cor: The correction factors after normalization as in run_kmer_frequency_normalization
- abs_cor: The correction factors without normalization.

See Also

normalizeMotifs

Examples

```
NULL
```

---

run_kmer_frequency_normalization

Provide normalized correction factors for kmer content

Description

This function is analogous to normalizeMotifs. If an analysis of mutational signatures is performed on e.g. Whole Exome Sequencing (WES) data, the signatures and exposures have to be adapted to the potentially different kmer (trinucleotide) content of the target capture. The present function takes as arguments paths to the used reference genome and target capture file. It the extracts the sequence of the target capture by calling bedtools getfasta on the system command prompt. run_kmer_frequency_normalization then calls a custom made perl script kmer_frequencies.pl also included in this package to count the occurences of the tripletts in both the whole reference genome and the created target capture sequence. These counts are used for normalization as in normalizeMotifs. Note that kmerFrequency provides a solution to approximate kmer frequencies by random sampling. As opposed to that approach, the function described here deterministically counts all occurences of the kmers in the respective genome.

Usage

```
run_kmer_frequency_normalization(
  in_ref_genome_fasta,
  in_target_capture_bed,
  in_word_length,
  project_folder,
  in_verbose = 1
)
```

Arguments

in_ref_genome_fasta	Path to the reference genome fasta file used.
in_target_capture_bed	Path to a bed file containing the information on the used target capture. May also be a compressed bed.
in_word_length	Integer number defining the length of the features or motifs, e.g. 3 for tripletts or 5 for pentamers
project_folder	Path where the created files, especially the fasta file with the sequence of the target capture and the count matrices, can be stored.
in_verbose	Verbose if in_verbose=1

Value

A numeric vector with correction factors

See Also

normalizeMotifs

Examples

```
NULL
```

---

run_plot_strata_general

Wrapper function for plot_strata

Description

First calls

- make_strata_df, then
- plot_strata

Usage

```
run_plot_strata_general(
  in_stratification_lists_list,
  in_signatures_ind_df,
  output_path = NULL,
  in_attribute = "",
  in_remove_signature_ind = NULL,
  in_additional_stratum = NULL
)
```

Arguments

in_stratification_lists_list	List of lists with entries from different (orthogonal) stratification axes or SMCs
in_signatures_ind_df	A data frame containing meta information about the signatures
output_path	Path to directory where the results, especially the figure produced by plot_strata is going to be stored.
in_attribute	Additional string for the file name where the figure produced by plot_strata is going to be stored.
in_remove_signature_ind	Omit one of the signatures in in_signatures_ind_df for the comparison if non-NULL. The parameter specifies the index of the signature to be removed.
in_additional_stratum	Include an additionally supplied stratum in comparison in non-NULL.

Value

The function doesn't return any value.

See Also

plot_strata

Examples

```
NULL
```

---

run_SMC *Wrapper function for the Stratification of a Mutational Catalogue*

Description

run_SMC takes as input a big dataframe constructed from a vcf-like file of a whole cohort. This wrapper function calls custom functions to construct a mutational catalogue and stratify it according to categories indicated by a special column in the input dataframe:

- create_mutation_catalogue_from_df
- adjust_number_of_columns_in_list_of_catalogues

This stratification yields a collection of stratified mutational catalogues, these are reformatted and sent to the custom function SMC and thus indirectly to LCD_SMC to perform a signature analysis of the stratified mutational catalogues. The result is then handed over to plot_SMC for visualization.

Usage

```
run_SMC(
  my_table,
  this_signatures_df,
  this_signatures_ind_df,
  this_subgroups_df,
  column_name,
  refGenome,
  cohort_method_flag = "all_PIDs",
  in_strata_order_ind = seq_len(length(unique(my_table[, column_name]))),
  wordLength = 3,
  verbose_flag = 1,
  target_dir = NULL,
  strata_dir = NULL,
  output_path = NULL,
  in_all_exposures_df = NULL,
  in_rownames = c(),
  in_norms = NULL,
  in_label_orientation = "turn",
  this_sum_ind = NULL
)
```

Arguments

`my_table`	A big dataframe constructed from a vcf-like file of a whole cohort. The first columns are those of a standard vcf file, followed by an arbitrary number of custom or user defined columns. One of these must carry a PID (patient or sample identifyier) and one must be the category used for stratification.
`this_signatures_df`	A numeric data frame `W` in with `n` rows and `l` columns, `n` being the number of features and `l` being the number of signatures
`this_signatures_ind_df`	A data frame containing meta information about the signatures
`this_subgroups_df`	A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup
`column_name`	Name of the column in `my_table` which is going to be used for stratification
`refGenome`	FaFile of the reference genome to extract the motif context of the variants in `my_table`
`cohort_method_flag`	Either or several of `c("all_PIDs","cohort","norm_PIDs")`, representing alternative ways to average over the cohort.
`in_strata_order_ind`	Index vector defining reordering of the strata
`wordLength`	Integer number defining the length of the features or motifs, e.g. 3 for tripletts or 5 for pentamers
`verbose_flag`	Verbose if `verbose_flag=1`
`target_dir`	Path to directory where the results of the stratification procedure are going to be stored if non-NULL.

strata_dir — Path to directory where the mutational catalogues of the different strata are going to be stored if non-NULL

output_path — Path to directory where the results, especially the figures produced by plot_SMC are going to be stored.

in_all_exposures_df — Optional argument, if specified, H, i.e. the overall exposures without stratification, is set to equal in_all_exposures_df. This is equivalent to forcing the LCD_SMC procedure to use e.g. the exposures of a previously performed NMF decomposition.

in_rownames — Optional parameter to specify rownames of the mutational catalogue V i.e. the names of the features.

in_norms — If specified, vector of the correction factors for every motif due to differing trinucleotide content. If null, no correction is applied.

in_label_orientation — Whether or not to turn the labels on the x-axis.

this_sum_ind — Optional set of indices for reordering the PIDs

Value

A list with entries exposures_list, catalogues_list, cohort and name_list.

- exposures_list: The list of s strata specific exposures Hi, all are numerical data frames with l rows and m columns, l being the number of signatures and m being the number of samples
- catalogues_list: A list of s strata specific cohortwide (i.e. averaged over cohort) normalized exposures
- cohort: subgroups_df adjusted for plotting
- name_list: Names of the contructed strata.

See Also

create_mutation_catalogue_from_df

normalizeMotifs_otherRownames

plot_SMC

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
data(sigs)
data(lymphoma_test)
data(lymphoma_cohort_LCD_results)
strata_list <-
  cut_breaks_as_intervals(lymphoma_test_df$random_norm,
                          in_outlier_cutoffs=c(-4,4),
                          in_cutoff_ranges_list=list(c(-2.5,-1.5),
                                                     c(0.5,1.5)),
                          in_labels=c("small","intermediate","big"))
lymphoma_test_df$random_cat <- strata_list$category_vector
choice_ind <- (names(lymphoma_Nature2013_COSMIC_cutoff_exposures_df)
               %in% unique(lymphoma_test_df$PID))
lymphoma_test_exposures_df <-
  lymphoma_Nature2013_COSMIC_cutoff_exposures_df[,choice_ind]
temp_subgroups_df <- make_subgroups_df(lymphoma_test_df,
```

```
                                        lymphoma_test_exposures_df)
mut_density_list <- run_SMC(lymphoma_test_df,
                            AlexCosmicValid_sig_df,
                            AlexCosmicValid_sigInd_df,
                            temp_subgroups_df,
                            column_name="random_cat",
                            refGenome=BSgenome.Hsapiens.UCSC.hg19,
                            cohort_method_flag="norm_PIDs",
                            in_rownames = rownames(AlexCosmicValid_sig_df))
```

`shapiro_if_possible` *Wrapper for Shapiro test but allow for all identical values*

Description

Wrapper for Shapiro test but allow for all identical values

Usage

```
shapiro_if_possible(in_vector)
```

Arguments

`in_vector`	Numerical vector the Shapiro-Wilk test is computed on

Value

p-value of the Shapiro-Wilk test, zero if all entries in the input vector `in_vector` are identical.

See Also

`shapiro.test`

Examples

```
shapiro_if_possible(runif(100,min=2,max=4))
shapiro_if_possible(rnorm(100,mean=5,sd=3))
shapiro_if_possible(rep(4.3,100))
shapiro_if_possible(c("Hello","World"))
```

sigs *Data for mutational signatures*

Description

The numerical data of the mutational signatures published initially by Alexandrov et al. (Nature 2013) and Alexandrov et al., (Bioaxiv 2018) is stored in data frames with endings `_sig_df`, the associated meta-information is stored in data frames with endings `_sigInd_df`. There are several instances of `_sig_df` and `_sigInd_df`, corresponding to results and data obtained at different times and with different raw data. There always is a one-to-one correspondence between a `_sig_df` and a `_sigInd_df`. The data frames of type `_sig_df` have as many rows as there are features, i.e. 96 if analyzing mutational signatures of SNVs in a triplet context, and as many columns as there are signatures. Data frames of type `_sigInd_df` have as many rows as there are signatures in the corresponding `_sig_df` and several columns:

- `sig`: signature name
- `index`: corresponding to the row index of the signature
- `colour`: colour for visualization in stacked barplots
- `process`: asserted biological process
- `cat.coarse`: categorization of the signatures according to the asserted biological processes at low level of detail
- `cat.medium`: categorization of the signatures according to the asserted biological processes at intermediate level of detail
- `cat.high`: categorization of the signatures according to the asserted biological processes at high level of detail
- `cat.putative`: categorization of the signatures according to the asserted biological processes based on clustering and inference

Please note, that categorization columns are only present for the data frames corrosponding to the data from Alexandorv et al. (Nature 2013).

`AlexInitialArtif_sig_df`: Data frame of the signatures published initially by Alexandrov et al. (Nature 2013). There are 27 signatures which constitute the columns, 22 of which were validated by an orhtogonal sequencing technology. These 22 are in the first 22 columns of the data frame. The column names are *A* pasted to the number of the signature, e.g. *A5*. The nonvalidated signatures have an additional letter in their naming convention: either *AR1 - AR3* or *AU1 - AU2*. The rownames are the features, i.e. an encoding of the nucleotide exchanges in their trinucleotide context, e.g. *C>A ACA*. In total there are 96 different features and therefore 96 rows when dealing with a trinucleotide context.

`AlexInitialArtif_sigInd_df`: Meta-information for `AlexInitialArtif_sig_df`

`AlexInitialValid_sig_df`: Data frame of only the validated signatures published initially by Alexandrov et al. (Nature 2013), corresponding to the first 22 columns of `AlexInitialArtif_sig_df`

`AlexInitialValid_sigInd_df`: Meta-information for `AlexInitialValid_sig_df`

`AlexCosmicValid_sig_df`: Data frame of the updated signatures list maintained by Ludmil Alexandrov at https://cancer.sanger.ac.uk/cosmic/signatures. The column names are *AC* pasted to the number of the signature, e.g. *AC5*. The naming convention for the rows is as described for `AlexInitialArtif_sig_df`.

`AlexCosmicValid_sigInd_df`: Meta-information for `AlexCosmicValid_sig_df`

AlexCosmicArtif_sig_df: Data frame of the updated signatures list maintained by Ludmil Alexandrov at https://cancer.sanger.ac.uk/cosmic/signatures and complemented by the artifact signatures from the initial publication, i.e. the last 5 columns of AlexInitialArtif_sig_df. The column names are *AC* pasted to the number of the signature, e.g. *AC5*. The naming convention for the rows is as described for AlexInitialArtif_sig_df.

AlexCosmicArtif_sigInd_df: Meta-information for AlexCosmicArtif_sig_df

Usage

```
data(sigs)
```

Author(s)

Daniel Huebschmann <huebschmann.daniel@googlemail.com>

Source

AlexInitial: ftp://ftp.sanger.ac.uk/pub/cancer/AlexandrovEtAl/signatures.txt

AlexCosmic: https://cancer.sanger.ac.uk/cancergenome/assets/signatures_probabilities.txt

References

Alexandrov et al. (Nature 2013)

---

sigs_pcawg

Data for PCAWG SNV signatures (COSMIC v3), including artifacts PCAWG_SP_SBS_sigs_Artif_df*: Data frame of the signatures published by Alexandrov et al. (Biorxiv 2013) which were decomposed with the method SigProfiler. SNV signatures are labeled with SBS, single base signature. There are 67 signatures which constitute the columns, 47 of which were validated by a bayesian NFM mehtod, SignatureAnayzer. Validated signatures are SBS1-SBS26,SBS28-SBS42 and SBS44. SBS7 is split up into 7 a/b/c and d. SBS10 ans SBS17 are both split up into a and b. Resulting in a 47 validated sigantures. Please note, unlike the paper by Alexandrov et al. (Biorxiv 2018) the data sets do not contain a SBS84 and SBS85 as not all were availiablt to perfom supervised signature analysis. In total there are 96 different features and therefore 96 rows when dealing with a trinucleotide context.*

Description

PCAWG_SP_SBS_sigInd_Artif_df: Meta-information for PCAWG_SP_SBS_sigs_Artif_df

PCAWG_SP_SBS_sigs_Real_df: Data frame of only the validated signatures published by Alexandrov et al. (Biorxiv 2018), corresponding to the column 1-26, 28-42 and 44 of the PCAWG_SP_SBS_sigs_Artif_df data frame

PCAWG_SP_SBS_sigInd_Real_df: Meta-information for PCAWG_SP_SBS_sigs_Real_df

PCAWG_SP_ID_sigs_df: Data frame with Indel signatures published by Alexandrov et al. (Biorxiv 2018) which were decomposed with the method SigProfiler. There are 17 Sigantures reported but as

supervised signatures are only valid for whole genome sequencing data analysis. In whole genome sequencing data the Indel signature ID15 was not discribed and thus is not part of this data set. In total 83 features are described. The categorization consideres the size of the insertion and delition, the motif, and the sequence context. Hereby the number of repetition or patial repetition of the motif is determined.

PCAWG_SP_ID_sigInd_df: Meta-information for PCAWG_SP_ID_sigs_df

Usage

```
data(sigs_pcawg)
```

Author(s)

Lea Jopp-Saile <huebschmann.daniel@googlemail.com>

Source

PCAWG_SNV: https://www.synapse.org/#!Synapse:syn11738319

PCAWG_INDEL: https://cancer.sanger.ac.uk/cosmic/signatures/ID

References

Alexandrov et al. (Biorxiv 2018)

SMC *Stratification of a Mutational Catalogue*

Description

SMC takes a given collection of stratified mutational catalogues Vi, sends them to perform a mutational signatures decomposition by Linear Combination Decomposition (LCD) with the functions LCD_SMC with known signatures W. It subsequently performs some useful statistics and preparation for plotting with the function plot_SMC. SMC is naturally called by run_SMC.

Usage

```
SMC(
  df_list,
  this_signatures_df,
  in_all_exposures_df,
  number_of_strata,
  number_of_sigs,
  name_list,
  this_subgroups_df,
  mutation_catalogue_all_df,
  cohort_method_flag,
  in_verbose = 1
)
```

Arguments

df_list	A list of s stratified mutational catalogues Vi \(numeric data frames\) with n rows and m columns each, n being the number of features and m being the number of samples.This list is naturally provided in run_SMC.
this_signatures_df	A numeric data frame W in with n rows and l columns, n being the number of features and l being the number of signatures
in_all_exposures_df	The overall exposures H without stratification, a numeric data frame with l rows and m columns, l being the number of signatures and m being the number of samples
number_of_strata	The length of the list df_list
number_of_sigs	The number of signatures used in the current decomposition.
name_list	A list of names of the different strata
this_subgroups_df	A data frame indicating which PID (patient or sample identifyier) belongs to which subgroup
mutation_catalogue_all_df	The overall mutational catalogue V without stratification.
cohort_method_flag	Either or several of c("all_PIDs","cohort","norm_PIDs"), representing alternative ways to average over the cohort.
in_verbose	Verbose if in_verbose=1

Value

A list with entries exposures_strata_list, exposures_both_rel_df_list, this_subgroups_df, subgroup_ind and decomposition_method.

- exposures_strata_list: The list of s strata specific exposures Hi, all are numerical data frames with l rows and m columns, l being the number of signatures and m being the number of samples
- exposures_both_rel_df_list: A list of s strata specific cohortwide (i.e. averaged over cohort) normalized exposures
- this_subgroups_df: subgroups_df adjusted for plotting
- subgroup_ind: Index of the subgroups chosen and relevant for plotting.
- decomposition_method: String telling whether LCD or NMF was used, relevant only for handing over to plot_SMC.

See Also

run_SMC

plot_SMC

LCD_SMC

Examples

NULL

SMC_perPID *Run SMC at a per sample level*

Description

Run an SMC analysis (stratification of the mutational catalogue) at per sample / per-PID level, corresponding to a divide and conquer strategy. For every single PID, only those signatures actually present in this PID will be provided for the SMC analysis.

Usage

```
SMC_perPID(
  in_dfList,
  in_LCDlist,
  in_subgroups_df,
  in_save_plot = TRUE,
  in_save_dir = NULL,
  in_save_name = "KataegisSMCs.pdf",
  in_verbose_flag = 0,
  ...
)
```

Arguments

in_dfList	Named list of vcf-like data frames, one entry per sample/PID of a cohort.
in_LCDlist	Output of an LCD list perfomed on the above cohort, carrying notably information on the exposures (in_LCDlist$exposures), the present signatures (in_LCDlist$signatures) and meta information about the signatures(in_LCDlist$out_sig_ind_df).
in_subgroups_df	Data frame with subgroup information about the PIDs in the above mentioned cohort.
in_save_plot	Boolean flag to indicate whether per-PID plots should be saved.
in_save_dir	If per-PID plots are to be saved, this is the path where to save them.
in_save_name	Suffix to be appended to the sample name to generate the name of the saved per-PID plots.
in_verbose_flag	Whether to run verbose (1) or not (0).
...	Data passed on to run_SMC.

Value

A list of lists. The top level is a named per-PID list, each entry is of type SMClist (cf. run_SMC).

Examples

```
NULL
```

split_exposures_by_subgroups *Split an exposures data frame by subgroups*

Description

If a cohort consists of different subgroups, this function enables to split the data frame storing the signature exposures into a list of data frames with signature exposures, one per subgroup. This functionality is needed for stat_test_subgroups and stat_plot_subgroups

Usage

```
split_exposures_by_subgroups(
  in_exposures_df,
  in_subgroups_df,
  in_subgroups.field = "subgroup",
  in_PID.field = "PID"
)
```

Arguments

in_exposures_df	Numerical data frame of the exposures (i.e. contributions of the different signatures to the number of point mutations per PID)
in_subgroups_df	Data frame indicating which PID belongs to which subgroup
in_subgroups.field	Name indicating which column in in_subgroups_df contains the subgroup information
in_PID.field	Name indicating which column in in_subgroups_df contains the PID information

Value

List of data frames with the subgroup specific signature exposures.

See Also

stat_test_subgroups

stat_plot_subgroups

Examples

```
NULL
```

`stat_plot_subgroups` *Plot averaged signature exposures per subgroup*

Description

Plot one averaged signature exposure pattern per subgroup. Uses `split_exposures_by_subgroups`.

Usage

```
stat_plot_subgroups(
  in_exposures_df,
  in_subgroups_df,
  in_signatures_ind_df,
  in_subgroups.field = "subgroup",
  in_PID.field = "PID",
  in_colour_vector = NULL
)
```

Arguments

`in_exposures_df`	Numerical data frame of the exposures (i.e. contributions of the different signatures to the number of point mutations per PID)
`in_subgroups_df`	Data frame indicating which PID belongs to which subgroup
`in_signatures_ind_df`	Data frame carrying additional information on the signatures
`in_subgroups.field`	Name indicating which column in `in_subgroups_df` contains the subgroup information
`in_PID.field`	Name indicating which column in `in_subgroups_df` contains the PID information
`in_colour_vector`	If non-null, specifies the colours attributed to the subgroups

Value

The function doesn't return any value, it plots instead.

See Also

`split_exposures_by_subgroups`

Examples

```
NULL
```

stat_test_SMC *Apply statistical tests to a stratification (SMC)*

Description

stat_test_SMC tests for enrichment or depletion in the different strata of a stratification of the mutational catalogue for every signature independently by applying Kruskal Wallis tests. For those signatures where the Kruskal Wallis test gives a significant p-value, pairwise posthoc tests are carried out by calling kwAllPairsNemenyiTest. Additionally all data is tested for normality by Shapiro Wilk tests, so that the user may apply ANOVA and pairwise posthoc t-test where allowed.

Usage

```
stat_test_SMC(in_strat_list, in_flag = "norm")
```

Arguments

in_strat_list	A list with entries exposures_list, catalogues_list, cohort and name_list as in the output of run_SMC. • exposures_list: The list of s strata specific exposures Hi, all are numerical data frames with l rows and m columns, l being the number of signatures and m being the number of samples • catalogues_list: A list of s strata specific cohortwide (i.e. averaged over cohort) normalized exposures • cohort: subgroups_df adjusted for plotting • name_list: Names of the contructed strata.
in_flag	If "norm", all tests are performed on normalized exposures, otherwise the absolute exposures are taken.

Value

A list with entries kruskal_df, shapiro_df, kruskal_posthoc_list,

- kruskal_df: A data frame containing results (statistic and p values) of the Kruskal Wallis tests (tests for enrichment or depletion in the different strata for every signature independently).
- shapiro_df: A data frame containing results (p values) of the Shapiro Wilk tests (tests for normal distribution in the different strata for every signature independently).
- kruskal_posthoc_list: A list of results of pairwise posthoc tests carried out for those signatures where the Kruskal Wallis test yielded a significant p-value (carried out by kwAllPairsNemenyiTest).

See Also

run_SMC

kwAllPairsNemenyiTest

kruskal.test

shapiro_if_possible

shapiro.test

Examples

```
NULL
```

`stat_test_subgroups` *Test for differences in average signature exposures between subgroups*

Description

Apply Kruskal-Wallis tests to detect differences in the signature exposures between different subgroups. Uses `split_exposures_by_subgroups`. Algorithm analogous to `stat_test_SMC`.

Usage

```
stat_test_subgroups(
  in_exposures_df,
  in_subgroups_df,
  in_subgroups.field = "subgroup",
  in_PID.field = "PID"
)
```

Arguments

`in_exposures_df`	Numerical data frame of the exposures (i.e. contributions of the different signatures to the number of point mutations per PID)
`in_subgroups_df`	Data frame indicating which PID belongs to which subgroup
`in_subgroups.field`	Name indicating which column in `in_subgroups_df` contains the subgroup information
`in_PID.field`	Name indicating which column in `in_subgroups_df` contains the PID information

Value

A list with entries `kruskal_df`, `kruskal_posthoc_list`,

- `kruskal_df`: A data frame containing results (statistic and p values) of the Kruskal Wallis tests (tests for enrichment or depletion in the different strata for every signature independently).
- `kruskal_posthoc_list`: A list of results of pairwise posthoc tests carried out for those signatures where the Kruskal Wallis test yielded a significant p-value (carried out by `kwAllPairsNemenyiTest`).

See Also

`split_exposures_by_subgroups`

`stat_test_SMC`

`kwAllPairsNemenyiTest`

`kruskal.test`

Examples

```
NULL
```

`stderrmean` *Compute the standard error of the mean*

Description

This function returns the standard deviation of an input numerical vector divided by the square root of the length of the input vector

Usage

```
stderrmean(x)
```

Arguments

`x`	A numerical vector

Value

Standard deviation of an input numerical vector divided by the square root of the length of the input vector

Examples

```
A <- c(1,2,3)
sd(A)
stderrmean(A)
```

`sum_over_list_of_df` *Elementwise sum over a list of (numerical) data frames*

Description

Elementwise sum over a list of (numerical) data frames

Usage

```
sum_over_list_of_df(in_df_list)
```

Arguments

`in_df_list`	List of (numerical) data frames

Value

A numerical data frame with the same dimensions as the entries of `in_df_list` with elementwise sums

Examples

```
A <- data.frame(matrix(c(1,1,1,2,2,2),ncol=2))
B <- data.frame(matrix(c(3,3,3,4,4,4),ncol=2))
df_list <- list(A=A,B=B)
sum_over_list_of_df(df_list)
```

---

targetCapture_cor_factors

Correction factors for different target capture kits

Description

List of lists with correction factors for different target capture kits. The elements of the overall list are lists, every one carrying information for one target capture kit (and namend after it). The elements of these sublists are 64 dimensional vectors with correction factors for all triplets. They were computed using counts of occurence of the respective triplets in the target capture and in the reference genome and making ratios (either for the counts themselves as in `abs_cor` or for the relative occurences in `rel_cor`). The information in this data structure may be used as input to `normalizeMotifs_otherRownames`.

Usage

```
data(targetCapture_cor_factors)
```

Value

A list of lists of data frames

Author(s)

Daniel Huebschmann <huebschmann.daniel@googlemail.com>

---

testSigs

Test for significance of alternative models cohort wide

Description

Wrapper function for variateExpSingle for application cohort wide.

Usage

```
testSigs(
  in_catalogue_df,
  in_sig_df,
  in_exposures_df,
  in_factor = 0,
  in_pdf = NULL
)
```

Arguments

in_catalogue_df	Input numerical data frame of the mutational catalog of the cohort to be analyzed
in_sig_df	Numerical data frame of the signatures used for analysis.
in_exposures_df	Input numerical data frame of the exposures computed for the cohort to be analyzed
in_factor	Deviation factor of the altered alternative model.
in_pdf	Probability distribution function, parameter passed on to confIntExp if NULL assumed to be normal distribution.

Value

Returns a data frame

Examples

```
NULL
```

---

test_exposureAffected *Test significance of association*

Description

Test significance of association between a vector of exposures and a selection of samples, e.g. those affected by mutations in a pathway as returned by `find_affected_PIDs`

Usage

```
test_exposureAffected(
  in_exposure_vector,
  in_affected_PIDs,
  in_mutation_label = NULL,
  in_exposure_label = NULL
)
```

Arguments

in_exposure_vector	Named vector of a phenotype (e.g. exposures to a specific signature)
in_affected_PIDs	Character vector of samples affected by some criterion, e.g. mutations in a pathway as returned by `find_affected_PIDs`
in_mutation_label	If non-NULL, prefix to the mutation status (x-axis label) in the produced boxplot
in_exposure_label	If non-NULL, prefix to the exposures (y-axis label) in the produced boxplot

Value

A list with entries:

- `current_kruskal`: Kruskal test object from testing phenotype against affection
- `current_boxplot`: Boxplot of phenotype against affection

Examples

```
NULL
```

`test_gene_list_in_exposures` *Test if mutated PIDs are enriched in signatures*

Description

For all signatures found in a project, this function tests whether PIDs having mutations in a specified list of genes of interest have significantly higher exposures.

Usage

```
test_gene_list_in_exposures(
  in_gene_list,
  in_exposure_df,
  in_mut_table,
  in_gene.field = "GENE_short",
  in_p_cutoff = 0.05
)
```

Arguments

`in_gene_list`	List with genes of interest
`in_exposure_df`	Data frame with the signature exposures
`in_mut_table`	Data frame or table of mutations (derived from vcf-format)
`in_gene.field`	Name of the column in which the gene names are to be looked up
`in_p_cutoff`	Significance threshold

Value

A list with entries `pvals`, `exposure_df`, `number_of_mutated`,

- `pvals`: p-values of the t-tests performed on mutated vs. unmutated PIDs
- `exposure_df`: Transposed input exposures data frame with additional annotations for mutation status
- `number_of_mutated`: Number of PIDs carrying a mutation

Examples

```
NULL
```

transform_rownames_R_to_MATLAB
Change rownames from one naming convention to another

Description

Rownames or names of the features used differ between the different contexts a signature analysis is carried out in. The function transform_rownames_R_to_MATLAB changes from the convention used in the YAPSA pacakge to the one used by Alexandrov et al. in the MATLAB framework.

The function transform_rownames_MATLAB_to_R changes from the convention used in Alexandrov et al. in the MATLAB framework to the one used by the YAPSA pacakge.

The function transform_rownames_MATLAB_to_R changes from the convention used in stored mutational catalogues by Alexandrov et al. to the one used by the YAPSA pacakge.

The function transform_rownames_YAPSA_to_deconstructSigs changes from the convention used in the YAPSA package to the one used by the deconstructSigs package.

The function transform_rownames_YAPSA_to_deconstructSigs changes from the convention used in the deconstructSigs package to the one used by the YAPSA pacakge.

Usage

```
transform_rownames_R_to_MATLAB(in_rownames, wordLength = 3)

transform_rownames_MATLAB_to_R(in_rownames, wordLength = 3)

transform_rownames_nature_to_R(in_rownames, wordLength = 3)

transform_rownames_YAPSA_to_deconstructSigs(in_rownames, wordLength = 3)

transform_rownames_deconstructSigs_to_YAPSA(in_rownames, wordLength = 3)
```

Arguments

in_rownames	Character vector of input rownames
wordLength	Size of the considered motif context

Value

A character vector of the translated rownames.

Examples

```
NULL
```

translate_to_hg19 *Translate chromosome names to the hg19 naming convention*

Description

translate_to_hg19: In hg19 naming convention, chromosome names start with the prefix *chr* and the gonosomes are called *X* and *Y*. If data analysis is performed e.g. with BSgenome.Hsapiens.UCSC.hg19, this naming convention is needed. The inverse transform is done with translate_to_1kG.

translate_to_1kG: In 1kG, i.e. 1000 genomes naming convention, chromosome names have no prefix *chr* and the gonosomes are called *23* for *X* and *24* for *Y*. If data analysis is performed e.g. with hs37d5.fa, this naming convention is needed. The inverse transform is done with translate_to_hg19.

Usage

```
translate_to_hg19(in_dat, in_CHROM.field = "CHROM", in_verbose = FALSE)

translate_to_1kG(in_dat, in_CHROM.field = "chr", in_verbose = FALSE)
```

Arguments

in_dat	GRanges object, VRanges object or data frame which carries one column with chromosome information to be reformatted.
in_CHROM.field	String indicating which column of in_dat carries the chromosome information
in_verbose	Whether verbose or not.

Value

GRanges object, VRanges object or data frame identical to in_dat, but with the names in the chromosome column replaced (if dealing with data frames) or alternatively the seqlevels replaced (if dealing with GRanges or VRanges objects).

Examples

```
test_df <- data.frame(CHROM=c(1,2,23,24),POS=c(100,120000000,300000,25000),
                      dummy=c("a","b","c","d"))
hg19_df <- translate_to_hg19(test_df, in_CHROM.field = "CHROM")
hg19_df

test_df <- data.frame(CHROM=c(1,2,23,24),POS=c(100,120000000,300000,25000),
                      dummy=c("a","b","c","d"))
hg19_df <- translate_to_hg19(test_df, in_CHROM.field = "CHROM")
onekG_df <- translate_to_1kG(hg19_df, in_CHROM.field = "CHROM")
onekG_df
```

`trellis_rainfall_plot` *Create a rainfall plot in a trellis structure*

Description

A trellis is a plot structure which allows space optimized multi-panel multi track plots. This function uses the package **gtrellis** developed by Zuguang Gu, also available at https://www.bioconductor.org/packages/release/bioc/html/gtrellis.html. The graphics in the tracks within a gtrellis plot are mostly drawn with functions from the package **grid**. Note that for technical reasons, the column indicating the chromosome MUST have the name *chr* and be the first column in the data frame supplied to the gtrellis functions. Therefore reformatting is performed in this function before calling gtrellis functions.

Usage

```
trellis_rainfall_plot(
  in_rainfall_dat,
  in_point_size = unit(1, "mm"),
  in_rect_list = NULL,
  in_title = "",
  in_CHROM.field = "CHROM",
  in_POS.field = "POS",
  in_dist.field = "dist",
  in_col.field = "col"
)
```

Arguments

`in_rainfall_dat`	Data frame which has to contain at least columns for chromosome, position, intermutational distance and colour information
`in_point_size`	size of the points in the rainfall plot to be created has to be provided with appropriate units, e.g. in_point_size=unit(0.5,"mm")
`in_rect_list`	Optional argument, if present, will lead to highlighting of specified regions by coloured but transparent rectangles
`in_title`	Title in the figure to be created.
`in_CHROM.field`	String indicating which column of `in_rainfall_dat` carries the chromosome information
`in_POS.field`	String indicating which column of `in_rainfall_dat` carries the position information
`in_dist.field`	String indicating which column of `in_rainfall_dat` carries the intermutational distance information
`in_col.field`	String indicating which column of `in_rainfall_dat` carries the colour information encoding the nucleotide exchange

Value

The function doesn't return any value.

The function doesn't return any value.

See Also

gtrellis_layout

add_track

grid.points

Examples

```
data(lymphoma_test)
choice_PID <- "4121361"
PID_df <- subset(lymphoma_test_df,PID==choice_PID)
trellis_rainfall_plot(PID_df,in_point_size=unit(0.5,"mm"))
```

`variateExp` *Wrapper to compute confidence intervals for a cohort*

Description

Wrapper function around confIntExp, which is applied to every signature/sample pair in a cohort. The extracted upper and lower bounds of the confidence intervals are added to the input data which is reordered and melted in order to prepare for visualization with ggplot2.

Usage

```
variateExp(
  in_catalogue_df,
  in_sig_df,
  in_exposures_df,
  in_sigLevel = 0.05,
  in_delta = 0.4,
  in_pdf = NULL
)
```

Arguments

`in_catalogue_df`	Input numerical data frame of the mutational catalog of the cohort to be analyzed.
`in_sig_df`	Numerical data frame of the signatures used for analysis.
`in_exposures_df`	Input numerical data frame of the exposures computed for the cohort to be analyzed.
`in_sigLevel`	Significance level, parameter passed to confIntExp.
`in_delta`	Inflation parameter for the alternative model, parameter passed on to confIntExp
`in_pdf`	Probability distribution function, parameter passed on to confIntExp, if NULL assumed to be normal distribution.

Value

A melted data frame.

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
data(lymphoma_test)
data(lymphoma_cohort_LCD_results)
data(sigs)
word_length <- 3
temp_list <- create_mutation_catalogue_from_df(
  lymphoma_test_df,this_seqnames.field = "CHROM",
  this_start.field = "POS",this_end.field = "POS",
  this_PID.field = "PID",this_subgroup.field = "SUBGROUP",
  this_refGenome = BSgenome.Hsapiens.UCSC.hg19,
  this_wordLength = word_length)
lymphoma_catalogue_df <- temp_list$matrix
lymphoma_PIDs <- colnames(lymphoma_catalogue_df)
data("lymphoma_cohort_LCD_results")
lymphoma_exposures_df <-
  lymphoma_Nature2013_COSMIC_cutoff_exposures_df[,lymphoma_PIDs]
lymphoma_sigs <- rownames(lymphoma_exposures_df)
lymphoma_sig_df <- AlexCosmicValid_sig_df[,lymphoma_sigs]
lymphoma_complete_df <- variateExp(in_catalogue_df = lymphoma_catalogue_df,
                                   in_sig_df = lymphoma_sig_df,
                                   in_exposures_df = lymphoma_exposures_df,
                                   in_sigLevel = 0.025, in_delta = 0.4)
head(lymphoma_complete_df)
lymphoma_complete_df$sample <-
  factor(lymphoma_complete_df$sample,
         levels = colnames(lymphoma_exposures_df)[
           order(colSums(lymphoma_exposures_df), decreasing = TRUE)])
sig_colour_vector <- c("black", AlexCosmicValid_sigInd_df$colour)
names(sig_colour_vector) <-
  c("total", as.character(AlexCosmicValid_sigInd_df$sig))
ggplot(data = lymphoma_complete_df,
       aes(x = sample, y = exposure, fill = sig)) +
  geom_bar(stat = "identity") +
  geom_errorbar(aes(ymin = lower, ymax = upper), width = 0.2) +
  facet_wrap(~sig, nrow = nrow(lymphoma_exposures_df) + 1) +
  theme_grey() +
  theme(panel.border = element_rect(fill = NA, colour = "black"),
        strip.background = element_rect(colour = "black"),
        legend.position = "none") +
  scale_fill_manual(values = sig_colour_vector)
```

---

variateExpSingle *Wrapper for the likelihood ratio test*

Description

Application of the likelihood ratio test to mutational signatures, primarily for one single sample.

Usage

```
variateExpSingle(
  in_catalogue_vector,
  in_sig_df,
  in_exposure_vector,
  in_ind,
  in_factor = 1,
  in_pdf = NULL,
  verbose = FALSE
)
```

Arguments

in_catalogue_vector	Mutational catalog of the input sample.
in_sig_df	Data frame encoding the signatures used for the analysis.
in_exposure_vector	Exposure vector computed for the input sample.
in_ind	Index specifying which signature among in_sig_df is to be tested.
in_factor	Deviation factor of the altered alternative model.
in_pdf	Probability distibution function, parameter passed on to logLikelihood and later to computeLogLik
verbose	Verbose if in_verbose=1

Value

Returns a list

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
data(lymphoma_test)
data(lymphoma_cohort_LCD_results)
data(sigs)
word_length <- 3
temp_list <- create_mutation_catalogue_from_df(
  lymphoma_test_df,this_seqnames.field = "CHROM",
  this_start.field = "POS",this_end.field = "POS",
  this_PID.field = "PID",this_subgroup.field = "SUBGROUP",
  this_refGenome = BSgenome.Hsapiens.UCSC.hg19,
  this_wordLength = word_length)
lymphoma_catalogue_df <- temp_list$matrix
lymphoma_PIDs <- colnames(lymphoma_catalogue_df)
data("lymphoma_cohort_LCD_results")
lymphoma_exposures_df <-
  lymphoma_Nature2013_COSMIC_cutoff_exposures_df[, lymphoma_PIDs]
lymphoma_sigs <- rownames(lymphoma_exposures_df)
lymphoma_sig_df <- AlexCosmicValid_sig_df[, lymphoma_sigs]
variateExpSingle(
  in_ind = 1,
  in_factor = 1.5,
  in_catalogue_vector = lymphoma_catalogue_df[, 1],
  in_sig_df = lymphoma_sig_df,
```

```
in_exposure_vector = lymphoma_exposures_df[, 1])
```

YAPSA *Generate R documentation from inline comments.*

Description

Yet Another Package for mutational Signature analysis

Details

This package provides functions and routines useful in the analysis of mutational signatures (cf. L. Alexandrov et al., Nature 2013). In particular, functions to perform a signature analysis with known signatures (`LCD` = linear combination decomposition) and a signature analysis on stratified mutational catalogue (`run_SMC` = stratify mutational catalogue) are provided.

Index