Package 'MutSeqR'

September 27, 2024

Title Analysis of error-corrected Next-Generation Sequencing Data for Mutation Detection

Version 0.0.0.9000

Description Standard methods for analysis of mutation data following ecNGS for the purpose of mutagencity assessment. Functions include importing the mutation lists provided by a variant caller, and a set of analytical tools for statistical testing and visualization of mutation data; comparison to COSMIC and/or germline signatures; etc.

License MIT + file LICENSE

Depends R (>= 3.4.0)

Imports binom,
BiocManager,
Biostrings,
BSgenome,
car,
data.table,
doBy,
dplyr,
GenomeInfoDb,
GenomicRanges,
ggplot2,
here,
httr,
IRanges,
lme4,
magrittr,
openxlsx,
patchwork,
plyranges,
reticulate,
rlang,
S4Vectors,
stringr,
SummarizedExperiment,
tidyr,
VariantAnnotation,
xml2

Suggests BSgenome.Hsapiens.UCSC.hg38,
BSgenome.Mmusculus.UCSC.mm10,

GenVisR,
knitr,
packcircles,
rmarkdown,
sigminer,
SigProfilerMatrixGeneratorR,
testthat (>= 3.0.0),
ToxicR,
trackViewer,
fs

Config/reticulate list(packages = list(list(package = ``SigProfilerAssignment''), list(package = ``SigProfilerExtractor''), list(package = ``SigProfilerMatrixGenerator'')))

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Contents

`add_binom_conf_intervals` *Add confidence intervals to mutation frequencies*

Description

Uses the binomial distribution to create confidence intervals for mutation frequencies calculated from a single point estimate using DNA sequencing

Usage

```
add_binom_conf_intervals(df, x, n, conf.level = 0.95, method = "wilson")
```

Arguments

`df`	The summary data frame containing the mutation frequencies
`x`	Column name that specifies the mutation count (e.g., mut_depth)
`n`	Column name that specifies the sequencing depth (e.g., total_depth)
`conf.level`	Confidence interval to calculate, default 95% (0.95)
`method`	The method used by binom::binom.confint to calculate intervals. Default is "wilson".

Value

A data frame with added columns indicating the confidence intervals.

annotate_CpG_sites *Annotate CpG sites*

Description

A simple method to test whether your trinucleotide context contains a CpG site. Vectorized version of Biostrings::vcountPattern is used.

Usage

```
annotate_CpG_sites(mut_data, motif = "CG", column_query = "context", ...)
```

Arguments

mut_data	A dataframe or GRanges object containing the genomic regions of interest in which to look for CpG sites.
motif	Default "CG", which returns CpG sites. You could in theory use an arbitrary string to look at different motifs. Use with caution. In this case the pattern being searched must be a column in the mutation data.
column_query	Default "context" but can be any column in the mutation data that you wish to look for a motif in.
...	Additional arguments to vcountPattern()

Value

A data frame with the same number of rows as there were ranges in the input, but with an additional metadata column indicating CpG sites in the target sequence of the mutation.

bmd_ma *Fit a model averaged continuous BMD model*

Description

This function fits a model averaged continuous BMD model to dose-response data for mutation frequency.

Usage

```
bmd_ma(
  mf_data,
  data_type = "individual",
  dose_col = "dose",
  response_cols = c("sample_MF_min", "sample_MF_max"),
  sd_col = NULL,
  n_col = NULL,
  bmr_type = "rel",
  bmr = 0.5,
  fit = "laplace",
  a = 0.025,
  ...
)
```

Arguments

`mf_data`	A data frame containing the dose-response data. Data may be individual for each sample or averaged over dose groups. Required columns for individual data are "dose" and "response". Multiple response columns are allowed. Required columns for summarised data are "dose", "mean response", "sample size", and "standard deviation". Only one response column is allowed for summarised data.
`data_type`	A string specifying the type of response data. Data may be response per individual or summarised across dose groups. ("individual", "summary").
`dose_col`	A character string specifying the column in mf_data containing the dose data.
`response_cols`	A character vector specifying the columns in mf_data containing the response data. For summarised data types, this should be the mean response for each dose group.
`sd_col`	A character string specifying the column in mf_data containing the standard deviation of the response data. This is only required for summarised data types.
`n_col`	A character string specifying the column in mf_data containing the sample size of each dose group. This is only required for summarised data types.
`bmr_type`	A string specifying the type of benchmark response. For continuous models, there are four types of BMD definitions that are commonly used: • Relative deviation (default; 'BMR_TYPE = "rel"'). This defines the BMD as the dose that changes the control mean/median a certain percentage from the background dose, i.e. it is the dose, BMD that solves $\mid f(dose) - f(0) \mid = (1 \pm BMR)f(0)$ • Standard deviation ('BMR_TYPE = "sd"'). This defines the BMD as the dose associated with the mean/median changing a specified number of standard deviations from the mean at the control dose., i.e., it is the dose, BMD, that solves $\mid f(dose) - f(0) \mid = BMR \times \sigma$ • Hybrid deviation ('BMR_TYPE = "hybrid"'). This defines the BMD that changes the probability of an adverse event by a stated amount relative to no exposure (i.e 0). That is, it is the dose, BMD, that solves $\frac{Pr(X>x\|dose)-Pr(X>x\|0)}{Pr(X<x\|0)} = BMR$. For this definition, $Pr(X < x\|0) = 1 - Pr(X > X\|0) = \pi_0$, where $0 \leq \pi_0 < 1$ is defined by the user as "point_p," and it defaults to 0.01. Note: this discussion assumed increasing data. The fitter determines the direction of the data and inverts the probability statements for decreasing data. • Absolute deviation ('BMR_TYPE="abs"'). This defines the BMD as an absolute change from the control dose of zero by a specified amount. That is the BMD is the dose that solves the equation $\mid f(dose) - f(0) \mid = BMR$.
`bmr`	A numeric value specifying the benchmark response. The BMR is defined in relation to the calculation requested in bmr_type. Default is 0.5.
`fit`	A string specifying the method used to fit the model. Options are ("laplace", "mle", or "mcmc"). Default is "laplace".
`a`	The specified nominal coverage rate for computation of the lower bound on the BMDL and BMDU, i.e., one computes a $100 \times (1 - \alpha)\%$ confidence interval. For the interval (BMDL,BMDU) this is a $100 \times (1 - 2\alpha)\%$. By default, it is set to 0.025 for a CI of 95%.
`...`	Additional arguments to be passed to the model fitting function. See ma_continuous_fit for details.

Details

Model averaging is done over the default models described in The European Food Safety Authority's (2022) Guidance on the use of the benchmark dose approach in risk assessment. These models are:

- "exp-aerts": $f(x) = a(1 + (c-1)(1 - \exp(-bx^d)))$
- "invexp-aerts": $f(x) = a(1 + (c-1)(\exp(-bx^{-d})))$
- "hill-aerts": $f(x) = a(1 + (c-1)(1 - \frac{b^d}{b^d + x^d}))$
- "lognormal-aerts": $f(x) = a\left\{1 + (c-1)\left(\Phi(\ln(b) + d \times \ln(x))\right)\right\}$
- "gamma-efsa": $f(x) = a(1 + (c-1)(\Gamma(bx; d)))$
- "LMS": $f(x) = a(1 + (c-1)(1 - \exp(-bx - dx^2)))$
- "probit-aerts": $f(x) = c\left(\Phi(a + b \times x^d)\right)$
- "logistic-aerts": $f(x) = \frac{c}{1+\exp(-a-b\times x^d)}$

Here: $\Phi(\cdot)$ is the standard normal distribution and $\Phi_{SN}(\cdot;\cdot)$ is the skew-normal distribution

Value

A list with the following components:

- BMD: A data frame containing the BMD values and the $100 \times (1-2\alpha)\%$ confidence intervals for each response column.
- summary: A list containing the summary statistics for each repsonse column.
- model_plots: A list containing the model plots for each response column.
- cleveland_plots: A list containing the Cleveland plots for each response column.
- models: A list containing the model fit values for each response column. Values include:
  - Individual_Model_X: Individual model fits for each model, X, used for model averaging. See single_continuous_fit for details on the model object class structure.
  - ma_bmd: The CDF of the model averaged BMD distribution.
  - posterior_probs: the posterior probabilities used in the model averaging.

---

calculate_mut_freq *Calculate mutation frequency*

Description

Calculates the mutation frequency for arbitrary groupings and creates a new dataframe with the results. Mutation frequency is # mutations / total bases, but this can be subset in different ways: e.g., by mutation context. In this case, it is necessary to change the denominator of total bases to reflect the sequencing depth at the proper reference bases under consideration.

Usage

```
calculate_mut_freq(
  mutation_data,
  cols_to_group = "sample",
  subtype_resolution = "none",
 variant_types = c("snv", "deletion", "insertion", "complex", "mnv", "symbolic"),
  filter_germ = TRUE,
  summary = TRUE,
  retain_metadata_cols = NULL
)
```

Arguments

`mutation_data` The data frame to be processed containing mutation data. Required columns are listed below. Synonymous names for these columns are accepted.

- `contig`: The reference sequence name.
- `start`: 0-based start position of the feature in contig.
- `sample`: The sample name.
- `alt_depth`: The read depth supporting the alternate allele.
- total_depth: The total read depth at this position (excluding N-calls).
- `is_germline`: A logical variable indicating whether the mutation is a germline mutation.
- `variation_type`: The category to which this variant is assigned.
- subtype_col: The column containing the mutation subtype. This column depends on the `subtype_resolution` parameter.
- reference_col: The column containing the referene base(s) for the mutation. This column depends on the `subtype_resolution` parameter.
- metadata_cols for grouping: all columns across which you want to calculate the mutation frequency. Ex. `c("tissue", "dose")`. These columns should be listed in cols_to_group.

`cols_to_group` A vector of grouping variables: this should be the groups of interest that you want to calculate a frequency for. For instance, getting the frequency by `sample`. Other options might include `dose`, `locus`, or, `c("sample","locus")`. All listed variables must be a column in the mutation_data.

`subtype_resolution`

The resolution at which the frequencies are calculated. Options are

- "none" calculates mutation frequencies across all selected grouping columns. The reference_col is not needed.
- "type" calculates mutation frequencies across all selected grouping columns for each `variation_type` seperately; snv, mnv, deletion, insertion, complex, symbolic. The reference_col is not needed.
- "base_6" calculates mutation frequencies across all selected grouping columns for each variation_type with snv mutations separated by `normalized_subtype`; C>A, C>G, C>T, T>A, T>C, T>G. The reference_col is `normalized_ref`.
- "base_12" calculates mutation frequencies across all selected grouping columns for each variation_type with snv mutations separated by `subtype`; A>C, A>G, A>T, C>A, C>G, C>T, G>A, G>C, G>T, T>A, T>C, T>G. The reference_col is `short_ref`.

- "base_96" calculates mutation frequencies across all selected grouping columns for each variation_type with snv mutations separated by `normalized_context_with_mutation`, i.e. the 96-base trinucleotide context. Ex. A\[C>T\]A. The reference_col is `normalized_context`.
- "base_192" calculates mutation frequencies across all selected grouping columns for each variation_type with snv mutations separated by `context_with_mutation`, i.e. the 192-base trinucleotide context. Ex A\[G>A\]A. The reference_col is `context`.

`variant_types` Include these variant types in mutation counts. A vector of one or more variation_types. Options are: "snv", "complex", "deletion", "insertion", "mnv", "symbolic", "no_variant". Default includes all variants.

`filter_germ` A logical variable. If TRUE, exclude rows from the mutation count that were flagged as germline mutations in `is_germline`. Default is TRUE.

`summary` A logical variable, whether to return a summary table (i.e., where only relevant columns for frequencies and groupings are returned). Setting this to false returns all columns in the original mutation_data, which might make plotting more difficult, but may provide additional flexibility to power users.

`retain_metadata_cols`
a character vector that contains the names of the metadata columns that you would like to retain in the summary table. This may be useful for plotting your summary data. Ex. retain the "dose" column when summarising by "sample".

clonality_cutoff NOT CURRENTLY IMPLEMENTED! Up for consideration. This value determines the fraction of reads that is considered a constitutional variant. If a mutation is present at a fraction higher than this value, the reference base will be swapped, and the alt_depth recalculated. 0.3 (30%) would be a sane default?

Details

Additionally, by default, the operation is run by default using both the minimum and maximum independent methods for counting mutations.

Value

A data frame with the mutation frequency calculated. If summary is set to TRUE, the data frame will be a summary table with the mutation frequency calculated for each group. If summary is set to FALSE, the mutation frequency will be appended to each row of the original mutation_data.

- `_MF_min`: The mutation frequency calculated using the "min" method for mutation counting. All identical mutations within a samples are assumed to be the result of clonal expansion and are thus only counted once.
- `_MF_max`: The mutation frequency calculated using the "max" method for mutaiton counting. All identical mutations within a sample are assumed to be idenpendant mutational evens and are included in the mutation frequency calculation. Note that this does not apply for germline variants.
- `proportion_min`: The proportion of each mutation subtype within the group, normalized to its read depth. Calculated using the "min" method. This is only calculated if `subtype_resolution` is not "none".
- `proportion_max`: The proportion of each mutation subtype within the group, normalized to its read depth. Calculated using the "max" method. This is only calculated if `subtype_resolution` is not "none".

check_required_columns *Check that all required columns are present before proceeding with the function*

Description

A utility function that will check that all required columns are present.

Usage

```
check_required_columns(data, required_columns)
```

Arguments

data	mutation data
required_columns	a list of required column names.

Value

an error

cluster_spectra *Hierarchical Clustering*

Description

perform hierarchical clustering of samples based on the mutation spectra.

Usage

```
cluster_spectra(
  mf_data = mf_data,
  group_col = "sample",
  response_col = "proportion_min",
  subtype_col = "normalized_subtype",
  dist = "cosine",
  cluster_method = "ward.D"
)
```

Arguments

mf_data	A data frame containing the mutation data. This data must include a column containing the mutation subtypes, a column containing the sample/cohort names, and a column containing the response variable.
group_col	The name of the column in data that contains the sample/cohort names.
response_col	The name of the column in data that contains the response variable. Typical response variables can be the subtype frequency, proportion, or count.

subtype_col	The name of the column in data that contains the mutation subtypes.
dist	the distance measure to be used. This must be one of "cosine", "euclidean", "maximum", "manhattan","canberra", "binary" or "minkowski". See dist for details.
cluster_method	The agglomeration method to be used. See hclust for details.

Details

The cosine distance measure represents the inverted cosine similarity between samples:

Cosine Dissimilarity $= 1 - \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|}$

This equation calculates the cosine dissimilarity between two vectors A and B.

Value

A dendrogram object representing the hierarchical clustering of the samples.

---

denominator_dict *Values used for denominators in frequency calculations*

Description

These values are used to cross reference base substitution types to their appropriate denominators for calculations. That is", "for example, the 6 base substitution frequency should be subsetted based on the normalized_ref column which would contain only T or C (i.e., the pyrimidine context for base substitutions).

Usage

```
denominator_dict
```

Format

A vector with corresponding values

---

generate_bubble_plots *Generate Bubble Plots*

Description

Produces a ggplot object of bubble plots from given mutation data. Optionally, bubble plots can be facetted by a specified column.

Usage

```
generate_bubble_plots(
  mutation_data,
  facet_col = "dose",
  circle_spacing = 1,
  color_by = "normalized_subtype",
  circle_outline = "none",
  circle_resolution = 50
)
```

Arguments

mutation_data	Data frame containing the mutation data.
facet_col	A string with the column name to facet by. If NULL or not provided, no faceting is performed.
circle_spacing	Numerical value to adjust the spacing between circles.
color_by	Character vector specifying how to color the mutations. Accepted values are "normalized_subtype", "subtype", and "trinucleotide_subtype". NOT FULLY IMPLEMENTED.
circle_outline	Color for the circle outline. Default is "none", resulting in no outline color. Other accepted values are colors in the R language.
circle_resolution	Number of points to use for the circle resolution. Default is 50.

Value

A ggplot object with the bubble plot, facetted if specified.

---

get_CpG_mutations *Get mutations at CpG sites*

Description

Subset the mutation data provided and return only mutations that are found at CpG sites.

Usage

```
get_CpG_mutations(
  regions,
  mut_data,
  variant_types = c("snv", "insertion", "deletion", "mnv", "symbolic"),
  include_no_variants = T,
  motif = "CG"
)
```

Arguments

regions	A GRanges object containing the genomic regions of interest in which to look for CpG sites. Must have the metadata column "sequence" populated with the raw nucleotide sequence to search for CpGs. This object can be obtained using the get_seq.R function.
mut_data	A dataframe or GRanges object containing the mutation data to be interrogated.
variant_types	Include these variant types. A vector of one or more "snv", "complex", "deletion", "insertion", "mnv", "symbolic", "no_variant". Default includes all variants.
include_no_variants	TRUE or FALSE to indicate whether the table should include CpG sites with no variants. Useful if you want to know how many of the potential sites were mutated.
motif	Default "CG", which returns CpG sites. You could in theory use an arbitrary string to look at different motifs. Use with caution.

Value

A GRanges object where each range is a mutation at a CpG site (a subset of mutations from the larger object provided to the function).

`get_CpG_regions` *Get the coordinates of the CpG sites within your genomic regions*

Description

Filters the ranges of your genomic regions to find a positions with a specific motif. The default is CpG sites, but can be customizable.

Usage

```
get_CpG_regions(regions, motif = "CG")
```

Arguments

`regions`	A GRanges object containing the genomic regions of interest in which to look for CpG sites. Must have the metadata column "sequence" populated with the raw nucleotide sequence to search for CpGs. This object can be obtained using the get_seq.R function.
`motif`	Default "CG", which returns CpG sites. You could in theory use an arbitrary string to look at different motifs. Use with caution.

Value

A GRanges object where each range is a CpG site (a subset of ranges from the larger object provided to the function).

`get_ref_of_mut` *A utility function that will return the reference context of a mutation*

Description

A utility function that will return the reference context of a mutation

Usage

```
get_ref_of_mut(mut_string)
```

Arguments

`mut_string`	the mutation. Ex. T>C, `A[G>T]C`

get_region_seqs *Get sequence of Duplex Sequencing target regions*

Description

This is mostly a helper function. It imports package data to find target regions and some associated information, and further extends the table by getting raw nucleotide sequences for each region of the genome. Note that the way this is written, currently, the default genomes are hg38 and mm10 for human and mouse, respectively.

Usage

```
get_region_seqs(species)
```

Arguments

species	One of "mouse" or "human", to determine which regions to return.

Value

A GRanges object where each range is a target region.

get_seq *Get sequence of genomic target regions*

Description

Create a GRanges object from the target metadata and import raw nucleotide sequences from the UCSC database. https://genome.ucsc.edu

Usage

```
get_seq(
  regions = c("TSpanel_human", "TSpanel_mouse", "TSpanel_rat", "custom_interval"),
  custom_regions_file = NULL,
  rg_sep = "\t",
  genome = NULL,
  is_0_based = TRUE,
  padding = 0
)
```

Arguments

regions	"TSpanel_human", "TSpanel_mouse", "TSpanel_rat, or "custom_interval". The argument refers to the TS Mutagenesis panel of the specified species, or to a custom panel. If custom, provide file path in custom_regions_file.
custom_regions_file	"filepath". If regions is set to custom_interval, provide the file path for the tab-delimited file containing regions metadata. Required columns are "contig", "start", and "end".

rg_sep	The delimiter for importing the custom_regions_file. The default is tab-delimited.
genome	If a custom regions file is provided, indicate the genome assembly for the reference genome. Please refer to the UCSC genomes. Ex.Human GRCh38 = hg38 | Human GRCh37 = hg19 | Mouse GRCm38 = mm10 | Mouse GRCm39 = mm39 | Rat RGSC 6.0 = rn6 | Rat mRatBN7.2 = rn7
is_0_based	TRUE or FALSE. Indicates whether the target region coordinates are 0 based (TRUE) or 1 based (FALSE). If TRUE, ranges will be converted to 1-based.
padding	An integer value by which the function will extend the range of the target sequence on both sides. Modified region ranges will be reported in seq_start and seq_end. Default is 0.

Value

a GRanges object with sequences and metadata of targeted regions. Region ranges coordinates will become 1-based.

`import_mut_data` *Import a .mut file*

Description

Imports a .mut file into the local R environment.

Usage

```
import_mut_data(
  mut_file,
  mut_sep = "\t",
  rsids = FALSE,
  sample_data_file = NULL,
  sd_sep = "\t",
  vaf_cutoff,
  range_buffer = 0,
 regions = c("TSpanel_human", "TSpanel_mouse", "TSpanel_rat", "custom_interval", "none"),
  custom_regions_file = NULL,
  rg_sep = "\t",
  is_0_based = TRUE,
  depth_calc = "take_del",
  custom_column_names = NULL,
  output_granges = FALSE
)
```

Arguments

mut_file	"filepath". The .mut file containing mutation data to be imported. This can be either a data frame object or a filepath to a file or directory. If you specify a folder, the function will attempt to read all files in the folder and combine them into a single data frame. Required columns are listed below. Synonymous names for these columns are accepted. • `contig`: The reference sequence name.

- `start`: 0-based start position of the feature in contig.
- `end`: half-open end position of the feature in contig.
- `sample`: The sample name.
- `ref`: The reference allele at this position
- `alt`: The left-aligned, normalized, alternate allele at this position.
- `alt_depth`: The read depth supporting the alternate allele.
- depth col: The total read depth at this position. This column can be `total_depth` (excluding N-calls) or `depth`(including N-calls; if `total_depth` is not available).
- `variation_type`: The category to which this variant is assigned.
- `context`: The local reference trinucleotide context at this position (e.g. ATC - not necessarily the transcript codon)

`mut_sep` The delimiter for importing the .mut file. Default is tab-delimited.

`rsids` A logical variable; whether or not the .mut file contains rsID information (existing SNPs).

`sample_data_file`
An optional file containing additional sample metadata (dose, tissue, timepoint, etc.). This can be either a data frame object or a file path to a file.

`sd_sep` The delimiter for importing sample metadata table. Default is tab-delimited.

`vaf_cutoff` The function will add `is_germline` column that identifies ostensibly germline variants using a cutoff for variant allele fraction (VAF). There is no default value provided, but generally a value of 0.1 (i.e., 10%) is a good starting point. Setting this will flag variants that are present at a frequency greater than this value at a given site.

`range_buffer` An integer >= 0. Variants that occur outside of the defined regions' ranges will be filtered out. Use the range-buffer to extend the range within which a variant can occur. The default is 1 nucleotide outside of region ranges. Ex. Structural variants and indels may start outside of the regions. Adjust the range_buffer to include these variants. Rows that were filtered out of the mutation data will be returned in a separate data frame.

`regions` Values are `c("TSpanel_human", "TSpanel_mouse", "TSpanel_rat" "custom_interval", "non`
Indicates the target panel used for Duplex Sequencing. The argument refers to the TS Mutagenesis panel of the specified species, or to a custom panel. If "custom", provide the file path of your regions file in `custom_regions_file`.

`custom_regions_file`
"filepath". If `regions` is set to "custom_interval", provide the file containing regions metadata. Required columns are `contig`, `start`, and `end`. This can be either a data frame object or a file path to a file.

`rg_sep` The delimiter for importing the `custom_regions_file`. Default is tab-delimited.

`is_0_based` A logical variable. Indicates whether the `custom_regions_file` target region coordinates are 0 based (TRUE) or 1 based (FALSE). If TRUE, ranges will be converted to 1-based.

`depth_calc` Values are `c("take_del", "take_mean")`. In the instance when there are two or more calls at the same location within a sample, and the depths differ, this parameter chooses the method for resolving the difference. This occurs when a deletion is called in the data. It will be called alongside a no_variant. "take_mean" calculates the depth column by taking the mean of all depths in the group. "take_del" calculates the depth column by choosing only the depth of the deletion in the group, or if no deletion is present, the complex variant. If there is

no deletion or complex variant, then it takes the mean of the depths within the group. Default is "take_del". depth_col = `total_depth` or `depth`.

`custom_column_names`
: A list of names to specify the meaning of column headers. Since column names can vary with data, this might be necessary to digest the mutation data table properly. Typical defaults are set, but can be substituted in the form of `list(total_depth = "my_custom_depth_name", sample = "my_custom_sample_column_name")` For a comprehensive list, see examples. You can change one or more of these.

`output_granges`
: A logical variable; whether you want the mutation data to output as a GRanges object. Default output (FALSE) is as a dataframe.

Value

A table where each row is a mutation, and columns indicate the location, type, and other data. If `output_granges` is set to TRUE, the mutation data will be returned as a GRanges object, otherwise mutation data is returned as a dataframe. If the mutation data contains rows that are filtered out of the specified regions, results will be returned as a list. The first element will be the mutation data called `mut_dat` and the second element will be the rows that were filtered from the data called `rows_outside_regions`. If no rows are filtered out, the results will be returned as a single dataframe or GRanges object.

Output Column Definitions:

- `nchar_ref`: The length (in bp) of the reference allele.
- `nchar_alt`: The length (in bp) of the alternate allele.
- `varlen`: The length (in bp) of the variant.
- `total_depth`: The total read depth at this position, excluding N-calls.
- `vaf`: The variant allele fraction. Calculated as `alt_depth/depth_col` where `depth_col` can be `total_depth` or `depth`.
- `is_germline`: TRUE or FALSE. Flags ostensible germline mutations (`vaf` > `vaf_cutoff`).
- `ref_depth`: The total read depth at the position calling for the reference allele. Calculated as `depth_col` - `alt_depth` where `depth_col` can be `total_depthor` `depth`.
- `subtype`: The substitution type for the snv variant (12-base spectrum; e.g. A>C)
- `short_ref`: The reference base at this position.
- `normalized_subtype`: The C/T-based substitution type for the snv variant (6-base spectrum; e.g. A>C -> T>G).
- `normalized_ref`: The reference base in C/T-base notation for this position (e.g. A -> T).
- `context_with_mutation`: The substitution type fo the snv variant including the two flanking nucleotides (192-trinucleotide spectrum; e.g. `T[A>C]G`)
- `normalized_context_with_mutation`: The C/T-based substitution type for the snv variant including the two flanking nucleotide (96-base spectrum e.g. `T[A>C]G` -> `C[T>G]A`)
- `normalized_context`: The trinucleotide context in C/T base notation for this position (e.g. TAG -> CTA).
- `gc_content`: % GC of the trinucleotide context at this position.

`import_vcf_data` *Import a vcf file*

Description

The function reads the genomic vcf file(s) and extracts the data into a dataframe. The function also reads in sample metadata if provided and joins it with the mutation data. An interval list of genomic regions can be provided to filter out variants that occur outside of the defined regions' ranges. The function will use the reference genome to extract the trinucleotide context of every position in the mutation data. The function can output the mutation data as a dataframe or a granges object.

Usage

```
import_vcf_data(
  vcf_file,
  vaf_cutoff,
  range_buffer = 1,
  sample_data_file = NULL,
  sd_sep = "\t",
 regions = c("TSpanel_human", "TSpanel_mouse", "TSpanel_rat", "custom_interval", "none"),
  custom_regions_file = NULL,
  rg_sep = "\t",
  genome = NULL,
  species = NULL,
  masked_BS_genome = FALSE,
  depth_calc = "take_del",
  output_granges = FALSE
)
```

Arguments

`vcf_file` The path to the genomic .vcf or .vcf.gz file(s) to be imported. If you specify a folder, the function will attempt to read all files in the folder and combine them into on dataset. Multisample vcf files are not supported; vcf files must contain one sample each. Required fields are listed below

- FIXED FIELDS:
- `CHROM`: The reference sequence name.Equivalent to `contig`
- `POS`: 0-based start position of the feature in contig.
- `REF`: The reference allele at this position
- `ALT`: The left-aligned, normalized, alternate allele at this position.
- FORMAT FIELDS:
- `AD`: The allelic depths for the reference and alternate alleles in the order listed.
- `DP`: The total read depth at this position (including N-calls). Equivalent to `depth`.
- `VD`: Variant Depth. Equivalent to `alt_depth`.
- INFO FIELDS
- `TYPE`: The category to which this variant is assigned. Equivalent to `variation_type`.
- `END`: The half-open end position of the feature in contig.

- sample: An identifying field for your samples; either in the INFO field or as the header to the FORMAT field.
- SUGGESTED INFO FIELDS:
- SVTYPE: Structural variant types; INV DUP DEL INS FUS.
- SVLEN: Length of the structural variant in base pairs.

vaf_cutoff
: Add is_germline column that identifies ostensibly germline variants using a cutoff for variant allele fraction (VAF). There is no default value provided, but generally a value of 0.01 - 0.1 (i.e., 1% - 10%) is a good starting point. Setting this flag variants that are present at a frequency greater than this value at a given site.

range_buffer
: An integer >= 0 .Required if using a targetted approach. Variants that occur outside of the defined regions' ranges will be filtered out. Use the range-buffer to extend the range outside of a region within which a variant can occur. The default is 1 nucleotide outside of region ranges. Ex. Structural variants and indels may start outside of the regions. Adjust the range_buffer to include these variants in the final mutation data. Variants that are filtered out of the data are returned in a separate dataframe.

sample_data_file
: An optional file containing additional sample metadata (dose, timepoint, etc.). This can be a data frame or a file path.

sd_sep
: The delimiter for importing sample metadata tables. Default is tab-delimited

regions
: "TSpanel_human", "TSpanel_mouse", "TSpanel_rat" , "custom_interval" or "none". The 'TSpanel_' argument refers to the TS Mutagenesis panel of the specified species, or to a custom regions interval file. If set to 'custom_interval', please provide the file path in custom_regions_file and the genome assembly version of the reference genome using the 'genome' parameter. If you are not using a targeted approach, set regions to none, and supply the species and genome assembly of the reference genome using the 'species' and 'genome' parameters respectively.

custom_regions_file
: "filepath". If regions is set to custom_interval, provide the file path for the file containing regions metadata. Required columns are "contig", "start", and "end"

rg_sep
: The delimiter for importing the custom_regions_file. Default is tab-delimited

genome
: The genome assembly of the reference genome. For a ######## complete list, refer to https://genome.ucsc.edu. Ex.Human GRCh38 = hg38 | Human GRCh37 = hg19 | Mouse GRCm38 = mm10 | Mouse GRCm39 = mm39 | Rat RGSC 6.0 = rn6 | Rat mRatBN7.2 = rn7

species
: The species of the reference genome. Required if regions is set to none. The value can be the common name of the species or the scientific name. Ex. "human" or "Homo sapiens".

depth_calc
: In the instance when there are two or more calls at the same location within a sample, and the depths differ, this parameter chooses the method of calculation for the total_depth. take_mean calculates the total_depth by taking the mean reference depth and then adding all the alt depths. take_del calculates the total_depth by choosing only the reference depth of the deletion in the group, or if no deletion is present, the complex variant, then adding all alt depths. If there is no deletion or complex variant, it takes the mean of the reference depths. Default is "take_del".

output_granges
: TRUE or FALSE; whether you want the mutation data to output as a GRanges object. Default output is as a dataframe.

Value

A data frame or a GRanges object where each row is a mutation, and columns indicate the location, type, and other data.

---

install_ref_genome *Install the reference genome for the specified organism.*

Description

This function will use BSgenome to install the reference genome for a specified organism and assembly version.

Usage

```
install_ref_genome(organism, genome, masked = FALSE)
```

Arguments

organism	the name of the organism for which to install the reference genome. This can be the scientific name or a common name. For example Homo Sapiens, H. sapiens, or human
genome	The reference genome assembly version. Ex. hg18, mm10, rn6.
masked	Logical value. Whether to search for the 'masked' BSgenome. Default is FALSE.

Value

a BSgenome object

---

load_regions_file *Imports the regions file*

Description

A helper function to import the regions metadata file. It is used in import_mut_data and get_seq.

Usage

```
load_regions_file(regions, custom_regions_file = NULL, rg_sep = "\t")
```

Arguments

regions	"TSpanel_human", "TSpanel_mouse", 'TSpanel_rat', or "custom_interval". The argument refers to the TS Mutagenesis panel of the specified species, or to a custom panel. If custom, provide file path in custom_regions_file.
custom_regions_file	"filepath". If regions is set to 'custom_interval', provide the file path for the tab-delimited file containing regions metadata. Required columns are "contig", "start", and "end".
rg_sep	The delimiter for importing the custom_regions_file. The default is tab-delimited

lollipop_mutations *Plot mutations in lollipop plot*

Description

TO DO: Create plt without trackViewer package. Uses the trackViewer package to plot mutations in a lollipop plot in specific regions as defined by the user input.

Usage

```
lollipop_mutations(species = "human", mutations, ...)
```

Arguments

species	One of "human" or "mouse"
mutations	A GRanges object with mutation data
...	Additional arguments to trackViewer::lolliplot (e.g., ranges = GRanges("chr1", IRanges(104, 109))) Suggests trackViewer lolliplot

make_CpG_summary_table *Summarize CpG sites*

Description

Creates a summary table of CpG sites based on groupings of interest. This is basically a convenience function that wraps `calculate_mut_freqs()` over CpG data (or any data). See the documentation for that function for parameters. It is up to the user to supply proper data to the function.

Usage

```
make_CpG_summary_table(cpg_muts, ...)
```

Arguments

cpg_muts	A data frame containing CpG mutations TO DO: cpg_muts = df "cpg_mutations" is created in the .Rmd file, but is not created by any other function.
...	Additional arguments to calculate_mut_freqs()

mf_bmd *Fit a BMD model for mutation frequency data*

Description

This function fits a continuous BMD model to dose-response data for mutation frequency.

Usage

```
mf_bmd(
  mf_data,
  dose_col = "dose",
  response_col = c("sample_MF_min", "sample_MF_max"),
  model_types = c("hill", "exp-3", "exp-5", "power", "polynomial"),
  BMR = 0.5
)
```

Arguments

mf_data	A data frame with columns "dose" and "MFmin"
dose_col	A character string specifying the column in data to be used to identify dose.
response_col	A character string specifying the column in data to be used to identify response.
model_types	A string specifying the model type. Options are "hill", "exp-3","exp-5", "power", "polynomial"
BMR	A numeric value specifying the benchmark response. Default is 0.5.

Value

A list with the following components:

migrate_mut *Validate a mut file*

Description

Not tested yet. TODO! Could avoid data.table dependency here I think.

Usage

```
migrate_mut(mut_table, op = MutSeqR::op)
```

Arguments

mut_table	mutation data
op	Default is MutSeqR::op, a list of options (see also, op, TODO)

`model_mf` *Perform linear modelling on mutation frequency for given fixed and random effects*

Description

`model_mf` will fit a linear model to analyse the effect(s) of given factor(s) on mutation frequency and perform specified pairwise comparisons. This function will fit either a generalized linear model (glm) or, if supplied random effects, a generalized linear mixed-effects model (glmer) . Pairwise comparisons are conducted using the doBy library (esticon) and estimates are then back-transformed. The delta method is employed to approximate the back-transformed standard-errors. A Sidak correction is applied to adjust p-values for multiple comparisons.

Usage

```
model_mf(
  mf_data,
  fixed_effects,
  test_interaction = TRUE,
  random_effects = NULL,
  reference_level,
  muts = "sample_sum_min",
  total_count = "sample_group_depth",
  contrasts = NULL,
  cont_sep = "\t",
  ...
)
```

Arguments

`mf_data`	The data frame containing the mutation frequency data. Mutation counts and total sequencing depth should be summarized per sample alongside columns for your fixed effects. This data can be obtained using `calculate_mut_freq(summary=TRUE)`.
`fixed_effects`	The name(s) of the column(s) that will act as the fixed_effects (factor/independent variable) for modelling mutation frequency.
`test_interaction`	a logical value. Whether or not your model should include the interaction between the `fixed_effects`.
`random_effects`	The name of the column(s) to be analysed as a random effect in the model. Providing this effect will cause the function to fit a generalized linear mixed-effects model.
`reference_level`	Refers to one of the levels within each of your fixed_effects. The coefficient for the reference level will represent the baseline effect. The coefficients of the other levels will be interpreted in relation to the reference_level as deviations from the baseline effect.
`muts`	The column containing the mutation count per sample.
`total_count`	The column containing the sequencing depth per sample.

contrasts	a data frame or a filepath to a file that will provide the information necessary to make pairwise comparisons between groups. The table must consist of two columns. The first column will be a group within your fixed_effects and the second column must be the group that it will be compared to. The values must correspond to entries in your mf_data column for each fixed effect. Put the group that you expect to have the higher mutation frequency in the 1st column and the group that you expect to have a lower mutation frequency in the second column. For multiple fixed effects, separate the levels of each fixed_effect of a group with a colon. Ensure that all fixed_effects are represented in each entry for the table. See details for examples.
cont_sep	The delimiter for importing the contrast table file. Default is tab-delimited.
...	Extra arguments for glm or glmer. The glmer function is used when a random_effect is supplied, otherwise, the model uses the glm function.

Details

fixed_effects are variables that have a direct and constant effect on the dependent variable (ie mutation frequency).They are typically the experimental factors or covariates of interest for their impact on the dependent variable. One or more fixed_effect may be provided. If you are providing more than one fixed effect, avoid using correlated variables; each fixed effect must independently predict the dependent variable. Ex. `fixed_effects = c("dose", "genomic_target", "tissue", "age", etc)`.

Interaction terms enable you to examine whether the relationship between the dependent and independent variable changes based on the value of another independent variable. In other words, if an interaction is significant, then the relationship between the fixed effects is not constant across all levels of each variable. Ex. Consider investigating the effect of dose group and tissue on mutation frequency. An interaction between dose and tissue would capture whether the dose response differs between tissues.

random_effects account for the unmeasured sources of statistical variance that affect certain groups in the data. They help account for unobserved heterogeneity or correlation within groups. Ex. If your model uses repeated measures within a sample, `random_effects = "sample"`.

Setting a reference_level for your fixed effects enhances the interpretability of the model. Ex. Consider a fixed_effect "dose" with levels 0, 25, 50, and 100 mg/kg. Intuitively, the reference_level would refer to the negative control dose, "0" since we are interested in testing how the treatment might change mutation frequency relative to the control.

Examples of contrasts:

If you have a fixed_effect "dose" with dose groups 0, 25, 50, 100, then the first column would contain the treated groups (25, 50, 100), while the second column would be 0, thus comparing each treated group to the control group.

25 0

50 0

100 0

Alternatively, if you would like to compare mutation frequency between treated dose groups, then the contrast table would look as follows, with the lower dose always in the second column, as we expect it to have a lower mutation frequency. Keeping this format aids in interpretability of the estimates for the pairwise comparisons. Should the columns be reversed, with the higher group in the second column, then the model will compute the fold-decrease instead of the fold-increase.

100 25

100 50

50 25

Ex. Consider the scenario where the `fixed_effects` are "dose" (0, 25, 50, 100) and "genomic_target" ("chr1", "chr2"). To compare the three treated dose groups to the control for each genomic target, the contrast table would look like:

25:chr1 0:chr1

50:chr1 0:chr1

100:chr1 0:chr1

25:chr2 0:chr2

50:chr2 0:chr2

100:chr2 0:chr2

Troubleshooting: If you are having issues with convergence for your generalized linear mixed- effects model, it may be advisable to increase the tolerance level for convergence checking during model fitting. This is done through the `control` argument for the `lme4::glmer` function. The default tolerance is tol = 0.002. Add this argument as an extra argument in the `model_mf` function. Ex. `control = lme4::glmerControl(check.conv.grad = lme4:::.makeCC("warning", tol = 3e-3, relTol = NULL))`

Value

Model results are output as a list. Included are:

- model_data: the supplied mf_data with added column for model residuals.
- summary: the summary of the model.
- anova: the analysis of variance for models with two or more effects. Anova(`model`)
- residuals_histogram: the model residuals plotted as a histogram. This is used to check whether the variance is normally distributed. A symmetric bell-shaped histogram, evenly distributed around zero indicates that the normality assumption is likely to be true.
- residuals_qq_plot: the model residuals plotted in a quantile-quantile plot. For a normal distribution, we expect points to roughly follow the y=x line.
- point_estimates_matrix: the contrast matrix used to generate point-estimates for the fixed effects.
- point_estimates: the point estimates for the fixed effects.
- pairwise_comparisons_matrix: the contrast matrix used to conduct the pairwise comparisons specified in the `contrasts`.
- pairwise_comparisons: the results of pairwise comparisons specified in the `contrasts`.

---

op *Column names for mut tables*

Description

A list of column specifications

Usage

```
op
```

Format

A list with potential variable column names

`plot_mean_mf` *Plot the Mean Mutatation Frequency*

Description

This function calculates the mean mutation frequency across samples for given groups and plots the results.

Usage

```
plot_mean_mf(
  mf_data,
  group_col = "dose",
  mf_type = "both",
  plot_type = "line",
  plot_error_bars = TRUE,
  plot_indiv_vals = TRUE,
  group_order = "none",
  group_order_input = NULL,
  add_labels = "mean_count",
  scale_y_axis = "linear",
  x_lab = NULL,
  y_lab = NULL,
  plot_title = NULL,
  custom_palette = NULL
)
```

Arguments

`mf_data`	A data frame containing the mutation frequency data. This is obtained from the calculate_mut_freq function with SUMMARY = TRUE.
`group_col`	The column in mf_data by which to calculate the mean. Ex. "dose" or c("dose", "tissue").
`mf_type`	The type of mutation frequency to plot. Options are "min", "max", "both", or "stacked". If "both", the min and max mutation frequencies are plotted side by side. "stacked" can be chosen for bar plot_type only. If "stacked", the difference between the min and max MF is stacked on top of the min MF such that the total height of both bars represent the max MF. Default is "min".
`plot_type`	The type of plot to create. Options are "bar" or "line". Default is "bar".
`plot_error_bars`	Whether to plot the error bars. Default is TRUE. Error bars are standard error of the mean.
`plot_indiv_vals`	Whether to plot the individual values as data points. Default is FALSE.
`group_order`	The order of the groups along the x-axis. ' Options include: • `none`: No ordering is performed. Default.

- `smart`: Groups are ordered based on the sample names.
- `arranged`: Groups are ordered based on one or more factor column(s) in mf_data. Factor column names are passed to the function using the `group_order_input`.
- `custom`: Groups are ordered based on a custom vector of group names. The custom vector is passed to the function using the `group_order_input`.

`group_order_input`
: The order of the groups if group_order is "custom". The column name by which to arrange groups if group_order is "arranged".

`add_labels`
: The data labels to display on the plot. Either "indiv_count", "indiv_MF", "mean_count", "mean_MF", or "none". Count labels display the number of mutations, MF labels display the mutation frequency. Mean plots the mean value. Indiv plots the labels for individual data points (only if plot_indiv_vals = TRUE). Default is "none".

`scale_y_axis`
: The scale of the y axis. Either "linear" or "log". Default is "linear".

`plot_title`
: The title of the plot. Default is "Mean Mutation Frequency".

`custom_palette`
: A custom color palette to use for the plot. Values that can be customized include "Mean", "Individual", "Mean min", "Mean max", "Individual min", and "Individual max". Default is NULL. @return a ggplot object

`xlab`
: The x-axis label. Default is the value of group_col.

`ylab`
: The y-axis label. Default is "Mutation Frequency (mutations/bp)".

`plot_mf` *Plot the Mutation Frequency*

Description

This function creates a plot of the mutation frequency.

Usage

```
plot_mf(
  mf_data,
  group_col,
  plot_type = "bar",
  mf_type = "min",
  fill_col = NULL,
  custom_palette = NULL,
  group_order = "none",
  group_order_input = NULL,
  labels = "count",
  scale_y_axis = "linear",
  x_lab = NULL,
  y_lab = NULL,
  title = NULL
)
```

Arguments

mf_data	A data frame containing the mutation frequency data. This is obtained from the calculate_mut_freq function with SUMMARY = TRUE.
group_col	The name of the column containing the sample/group names for the x-axis.
plot_type	The type of plot to create. Options are "bar" or "point".
mf_type	The type of mutation frequency to plot. Options are "min", "max", "both", or "stacked". If "both", the min and max mutation frequencies are plotted side by side. "stacked" can be chosen for bar plot_type only. If "stacked", the difference between the min and max MF is stacked on top of the min MF such that the total height of both bars represent the max MF.
fill_col	The name of the column containing the fill variable.
custom_palette	A character vector of colour codes to use for the plot. If NULL, a default palette is used
group_order	The order of the samples/groups along the x-axis. ' Options include: • none: No ordering is performed. Default. • smart: Samples are ordered based on the sample names. • arranged: Samples are ordered based on one or more factor column(s) in mf_data. Factor column names are passed to the function using the group_order_input. • custom: Samples are ordered based on a custom vector of sample names. The custom vector is passed to the function using the group_order_input.
group_order_input	The order of the samples/groups if group_order is "custom". The column name by which to arrange samples/groups if group_order is "arranged"
labels	The data labels to display on the plot. Either "count", "MF", or "none". Count labels display the number of mutations, MF labels display the mutation frequency.
scale_y_axis	The scale of the y axis. Either "linear" or "log".
x_lab	The label for the x axis.
y_lab	The label for the y axis.
title	The title of the plot.

Value

A ggplot object

---

plot_spectra *Transition-transversion plot*

Description

Given a data frame construct a plot displaying the mutation subtypes observed in a cohort.

Usage

```
plot_spectra(
  mutation_data,
  group_col = "sample",
  subtype_resolution = "base_6",
  response = c("frequency", "proportion", "sum"),
  mf_type = c("min", "max"),
 variant_types = c("snv", "deletion", "insertion", "complex", "mnv", "symbolic"),
  group_order = "none",
  group_order_input = NULL,
  dist = "cosine",
  cluster_method = "ward.D",
  palette = NULL,
  x_lab = NULL,
  y_lab = NULL
)
```

Arguments

`mutation_data`	A data frame containing the mutation data. This data must include a column containing the mutation subtypes, a column containing the sample/cohort names, and a column containing the response variable. Typical response variables can be the subtype frequency, proportion, or count.
`group_col`	The name of the column in data that contains the sample/cohort names.
`subtype_resolution`	The resolution of the mutation spectra. Default is `base_6`.
`response`	The name of the column in data that contains the response variable. Typical response variables can be the subtype frequency, proportion, or count.
`mf_type`	The type of mutation frequency to use. Default is `min`.
`variant_types`	A character vector of the mutation types to include.
`group_order`	The method for ordering the samples within the plot. Options include: • `none`: No ordering is performed. Default. • `smart`: Groups are ordered based on the group names. • `arranged`: Groups are ordered based on one or more factor column(s) in mf_data. Column names are passed to the function using the `group_order_input`. • `custom`: Groups are ordered based on a custom vector of group names. The custom vector is passed to the function using the `group_order_input`. • `clustered`: Groups are ordered based on hierarchical clustering. The dissimilarity matrix can be specified using the `dist` argument. The agglomeration method can be specified using the `cluster_method` argument.
`group_order_input`	A character vector specifying details for the group order method. If `group_order` is `arranged`, `group_order_input` should contain the column name(s) to be used for ordering the samples. If `group_order` is `custom`, `group_order_input` should contain the custom vector of group names.
`dist`	The dissimilarity matrix for hierarchical clustering. Options are `cosine`, `euclidean`, `maximum`, `manhattan`, `canberra`, `binary` or `minkowski`. The default is `cosine`. See dist for details.

cluster_method	The agglomeration method for hierarchical clustering. Options are ward.D, ward.D2, single, complete, average (= UPGMA), mcquitty (= WPGMA), median (= WPGMC) or centroid (= UPGMC). The default is Ward.D. See hclust for details.
palette	A named vector of colors to be used for the mutation subtypes. The names of the vector should correspond to the mutation subtypes in the data. The default is a set of colors from the RColorBrewer package.
x_lab	The label for the x-axis. Default is the value of group_col.
y_lab	The label for the y-axis. Default is the value of response_col.

`plot_trinucleotide` *Plot the trinucleotide spectrum*

Description

Creates barplots of the trinucleotide spectrum for all levels of a given group based on the mutation data. All plots are exported.

Usage

```
plot_trinucleotide(
  mutation_data,
  response = c("frequency", "proportion", "sum"),
  mf_type = "min",
  group_col = "dose",
  max_y = c("individual", "group"),
  sum_totals = TRUE,
  output_path = NULL,
  output_type = "svg"
)
```

Arguments

mutation_data	A data frame containing mutation data. This can be obtained using the 'import_mut_data' or 'read_vcf' functions.
response	A character string specifying the type of response to plot. Must be one of 'frequency', 'proportion', or 'sum'.
mf_type	A character string specifying the mutation count method to plot. Must be one of 'min' or 'max'. Default is 'min'.
group_col	A character string specifying the column(s) in 'mutation_data' to group the data by. Default is 'sample'. The sum, proportion, or frequency will be calculated and a plot will be generated for all unique levels of this group. You can specify more than one column to group by.
max_y	A character string specifying the max response value for the y-axis. Must be one of 'individual' or 'group'.'individual' will adjust the maximum y-axis value for each level of the group independently of the others. 'group' will set the maximum y-axis value based on the entire dataset such that all plots will have the same scale. Default is 'group'.
sum_totals	A logical value specifying whether to sum the total mutations.

output_path	A character string specifying the path to save the output plot. Default is NULL. This will create an output directory in the current working directory.
output_type	A character string specifying the type of output file. Options are 'jpeg', 'pdf', 'png', 'svg', or 'tiff'. Default is 'svg'.

Details

The function calculates the mutation frequency and plots the trinucleotide spectrum for all levels of a given group based on the mutation data. The function calculates the mutation frequency using the 'calculate_mut_freq' function with "cols_to_group" set to 'group_col' and "subtype_resolution" set to 'base_96'. For a given group, mutation counts and total informative duplex bases are summed across all samples. Mutation frequency is calculated by dividing the total mutation counts by the total number of duplex bases. For a given mutation subtype, proportion is calculated as the proportion of total mutation counts normalized to the total number of duplex bases for a given group and subtype. ^ should just explain this in calculate mutation freq and refer to that function.

---

plot_trinucleotide_heatmap

Create a heatmap plot

Description

This function creates a heatmap plot using the provided data file.

Usage

```
plot_trinucleotide_heatmap(
  mf_data,
  group_var = "dose",
  mf_type = "min",
  mut_proportion_scale = "turbo",
  max = 0.2,
  rescale_data = FALSE,
  condensed = FALSE
)
```

Arguments

mf_data	The data file
group_var	The variable to group by.
mf_type	The type of mutation frequency to plot. Options are "min" or "max". (Default: "min")
mut_proportion_scale	The scale option for the mutation proportion. Options are passed to viridis::scale_fill_viridis_c. One of # inferno, magma, plasma, viridis, cividis, turbo, mako, or rocket. We highly reccomend the default for its ability to disciminate hard to see patterns. (Default: "turbo")
max	Maximum value used for plotting the relative contributions. Contributions that are higher will have the maximum colour. (Default: 0.2)
rescale_data	Logical value indicating whether to rescale the mutation proportions to increase the dynamic range of colors shown on the plot. (Default: TRUE)
condensed	More condensed plotting format. Default = F.

Value

A ggplot object representing the heatmap plot.

Examples

```
create_heatmap(mutation_data, dose, "inferno")
```

---

print_ascii_art *This function prints ASCII art when the package is loaded*

Description

This function prints ASCII art when the package is loaded

Usage

```
print_ascii_art()
```

---

radar_plot *Create a radar plot*

Description

Create a radar plot

Usage

```
radar_plot(mf_data, response_col, label_col, facet_col)
```

Arguments

mf_data	A data frame with the data to plot
response_col	The column with the response values
label_col	The column with the labels for the radar plot.
facet_col	The column with the group to facet the radar plots.

Value

A radar plot

`rename_columns` *Map column names of mutation data to default column names. A utility function that renames columns of mutation data to default columns names.*

Description

Map column names of mutation data to default column names. A utility function that renames columns of mutation data to default columns names.

Usage

```
rename_columns(data, column_map = op$column)
```

Arguments

`data`	mutation data
`column_map`	a list that maps synonymous column names to their default.

Value

the mutation data with column names changed to match default.

`render_report` *Read configuration file and render R Markdown document*

Description

This function reads a configuration file in YAML format, extracts the parameters, and renders an R Markdown document using the specified parameters.

Usage

```
render_report(config_filepath, output_file)
```

Arguments

`config_filepath`	The path to the configuration file.
`output_file`	The path to the output file.

Value

None

Examples

```
read_config_and_render("config.yaml", "output.html")
```

`reverseComplement` *Get the reverse complement of a DNA or RNA sequence.*

Description

Get the reverse complement of a DNA or RNA sequence.

Usage

```
reverseComplement(
  x,
  content = c("dna", "rna"),
  case = c("lower", "upper", "as is")
)
```

Arguments

`x`	A character vector of DNA or RNA sequences.
`content`	c("dna", "rna") The type of sequence to be reversed.
`case`	c("lower", "upper", "as is") The case of the output sequence.

Details

This file is part of the source code for SPGS: an R package for identifying statistical patterns in genomic sequences. Copyright (C) 2015 Universidad de Chile and INRIA-Chile A copy of Version 2 of the GNU Public License is available in the share/licenses/gpl-2 file in the R installation directory or from http://www.R-project.org/Licenses/GPL-2. reverseComplement.R

`sidak` *Correct p-values for multiple comparisons*

Description

Correct p-values for multiple comparisons

Usage

```
sidak(vecP)
```

Arguments

`vecP`	vector of p-values

Details

This function corrects a vector of probabilities for multiple testing using the Bonferroni (1935) and Sidak (1967) corrections. References: Bonferroni (1935), Sidak (1967), Wright (1992). Bonferroni, C. E. 1935. Il calcolo delle assicurazioni su gruppi di teste. Pp. 13-60 in: Studi in onore del Professore Salvatore Ortu Carboni. Roma. Sidak, Z. 1967. Rectangular confidence regions for the means of multivariate normal distributions. Journal of the American Statistical Association 62:626-633. Wright, S. P. 1992. Adjusted P-values for simultaneous inference. Biometrics 48: 1005-1013. Pierre Legendre, May 2007

Value

adjusted p-values

`signature_analysis_sigminer` *Run COSMIC signatures comparison*

Description

After cleaning the mutation data input, runs several Alexandrov Lab tools for COSMIC signature analysis (assigns signatures to best explain the input data).

Usage

```
signature_analysis_sigminer(
  mutations,
  project_name = "Default",
  project_genome = "BSgenome.Mmusculus.UCSC.hg38",
  group = "sample",
  run_bootstrapping = F,
  vaf_cutoff,
  ...
)
```

Arguments

`mutations`	A data frame, imported from a .mut or .vcf file
`project_name`	The name of the project; used to get mutation data into the required .txt format for SigProfiler
`project_genome`	A string describing the reference genome to use; e.g., GRCh38
`group`	The column in the mutation data used to aggregate groups (e.g., sample ID, tissue, dose)
`run_bootstrapping`	TRUE or FALSE. Default FALSE. Determines if the sig_fit_bootstrap_batch() function should be run. This *should* be done, but the process is slow, so it's best to confirm that the rest of the analysis is working as expected first.
`vaf_cutoff`	The threshold above which variants are identified as ostensibly germline variants using a cutoff for variant allele fraction (VAF). There is no default value provided, but generally a value of 0.1 (i.e., 10%) is a good starting point. Setting this will remove variants that are present at a frequency greater than this value at a given site.
`...`	additional arguments may be supplied To Do: we need to document the elipsis ... in the parameters of the function in a way that doesn't cause a warning during Check()

Value

Creates a subfolder in the output directory with SigProfiler tools results. Suggests: sigminer

signature_fitting *Run COSMIC signatures comparison*

Description

After cleaning the mutation data input, runs several Alexandrov Lab tools for COSMIC signature analysis (assigns signatures to best explain the input data).

Usage

```
signature_fitting(
  mutation_data,
  project_name = "Default",
  project_genome = "GRCh38",
  env_name = "MutSeqR",
  group = "sample",
  output_path = NULL,
  python_version = "3.11",
  python_path = "~/../../AppData/Local/Programs/Python/Python310/python.exe",
  python_home =
   "C:/Users/adodge/OneDrive - HC-SC PHAC-ASPC/Documents/.virtualenvs/r-reticulate"
)
```

Arguments

mutation_data	A data frame, imported from a .mut file
project_name	The name of the project; used to get mutation data into the required .txt format for SigProfiler
project_genome	A string describing the reference genome to use; e.g. GRCh37, GRCH38, mm10, mm9, rn6 output_path The directory where output results should be written. *not a parameter of the function
env_name	The name of the virtual environment. This will be created on first use.
group	The column in the mutation data used to aggregate groups (e.g., sample ID, tissue, dose)
output_path	The filepath to thedirectory in which the output folder will be created to store results.
python_version	The version of python to be used.
python_path	The path to the version of python to be used with reticulate. It is important that this version of python meets the dependencies, including the SigProfiler python tools.
python_home	The path to the conda virtual environment that contains the required python dependencies

Value

Creates a subfolder in the output directory with SigProfiler tools results. SigProfilerMatrixGeneratorR SigProfilerMatrixGeneratorR install

`spectra_comparison` *Compare the overall mutation spectra between groups*

Description

spectra_comparison compares the mutation spectra of groups using a modified contingency table approach.

Usage

```
spectra_comparison(
  mutation_data,
  cols_to_group,
  subtype_resolution = "base_6",
 variant_types = c("snv", "deletion", "insertion", "complex", "mnv", "symbolic"),
  mf_type = "min",
  contrasts,
  cont_sep = "\t"
)
```

Arguments

`mutation_data`	A data frame containing the mutation data. This is the output from import_mut_data or import_vcf_data.
`cols_to_group`	A character vector of the column names in the mutation data that you want to group by. Ex. c("dose", "tissue"). This function will sum the mutations across groups before running the comparison.
`subtype_resolution`	The resolution of the mutation spectra to be compared. Options include "base_6", "base_12", "base_96", and "base_192" and "type". See calculate_mut_freq for more details.
`variant_types`	A character vector of the mutation types to include in the comparison. Default is all types of mutations.
`mf_type`	The type of mutation frequency to use. Default is "min".
`contrasts`	a filepath to a tab-delimited `.txt` file OR a dataframe that will specify the comparisons to be made between groups. The table must consist of two columns. The first column will be a level within your group column and the second column must be the group level that it will be compared to. All values must correspond to entries in your `cols_to_group` column. For more than one `group` variable, separate the levels of each group with a colon. Ensure that all `groups` listed in cols_to_group are represented in each entry for the table. See details for examples.
`cont_sep`	The delimiter used to import the contrasts table. Default is tab.

Details

Examples of `contrasts`: If you have `group = "dose"` with dose groups 0, 25, 50, 100. The first column would contain the treated groups (25, 50, 100), while the second column would be 0, thus comparing each treated group to the control group.

25 0

50 0

100 0

Ex. Consider two grouping variables `group = c("dose", "tissue")`; with levels dose (0, 25, 50, 100) and tissue("bone_marrow", "liver"). To compare the mutation spectra between tissues for each dose group, the contrast table would look like:

0:bone_marrow 0:liver

25:bone_marrow 25:liver

50:bone_marrow 50:liver

100:bone_marrow 100:liver

Value

the log-likelihood statistic G2 for the specified comparisons with the p-value adjusted for multiple-comparisons.

`subtype_dict` *Values accepted for mutation subtypes*

Description

These values are used to enable user input to translate to columns in a mut file

Usage

```
subtype_dict
```

Format

A vector with corresponding values

`subtype_list` *A list of mutation subtypes at different resolutions*

Description

A list of mutation subtypes at different resolutions

Usage

```
subtype_list
```

Format

A list with corresponding values

write_excel_from_list *Write Excel tables*

Description

Takes a list of tables (data frames) and writes each one to a separate Excel sheet in a workbook. Names of tabs will be based on names in the list.

Usage

```
write_excel_from_list(list_of_tables, output_path, workbook_name)
```

Arguments

list_of_tables	A named list of data frames to be written.
output_path	The directory where the Excel file should be written.
workbook_name	The file name for the Excel file.

Value

A saved Excel workbook.

write_excel_single_table *Write Excel table*

Description

Takes a single data frame and writes it to an Excel workbook.

Usage

```
write_excel_single_table(
  mut_data,
  output_path = "./",
  workbook_name = "Default"
)
```

Arguments

mut_data	The data frame to be written.
output_path	The directory where the Excel file should be written.
workbook_name	The file name for the Excel file.

Value

A saved Excel workbook.

`write_mutational_matrix` *Write a Mutational Matrix to input into the sigprofiler web application*

Description

Creates a .txt file from mutation data that can be used for mutational signatures analysis using the SigProfiler web application. Can handle group analyses (ex dose, tissue, etc). Currently only supports snv.

Usage

```
write_mutational_matrix(
  mutation_data,
  group = "dose",
  matrices = "base_96",
  mf_type = "min",
  filter = "somatic",
  output_path = NULL
)
```

Arguments

`mutation_data`	The GRanges object containing the mutation data. The output of import_mut_data() or read_vcf().
`group`	The column in the mutation data used to aggregate groups (e.g., sample, tissue, dose)
`matrices`	At what resolution should the snv mutation counts be calculated? Options are "base_96", 1536?, 384?, 6144?, DINUC? 6, 24, INDEL TO DO: add more matrices options.
`mf_type`	Mutation counting method. Options are "min" or "max". Minimum Independent Frequency (min): All identical mutations within a sample are assumed to be the result of clonal expansion and are thus only counted once. Maximum Independent Frequency (max): All identical mutations within a sample are assumed to be idenpendant mutational evens and are included in the mutation frequency calculation. Note that this does not apply for germline variants. Default is "min". TO DO: clonality cut_off?
`filter`	Parameter allows you to choose to filter for only somatic or germline mutations.
`output_path`	The path to save the output file. Default is NULL, which will save the file in the current working directory. Values = c("somatic", "germline", "none). "none" will leave the data unfiltered.

Value

a .txt file that can be uploaded to the sigprofiler web application as "Mutational Matrix"

write_mutation_calling_file

Write the mutation calling file to input into the sigprofiler web application

Description

Creates a .txt file from mutation data that can be used for mutational signatures analysis using the SigProfiler web application.Cannot group higher than sample.

Usage

```
write_mutation_calling_file(
  mutation_data,
  project_name = "test",
  project_genome = "GRCm38",
  output_path = NULL
)
```

Arguments

mutation_data	The GRanges object containing the mutation data. The output of import_mut_data() or read_vcf().
project_name	The name of the project; used to get mutation data into the required .txt format for SigProfiler
project_genome	A string describing the reference genome to use (e.g., GRCh38, GRCm38)
output_path	The path to save the output file. Default is NULL. See import_mut_data() or read_vcf() for more details.

Value

a .txt file that can be uploaded to the sigprofiler web application as "Mutational calling file" Filters out ostensibly germline mutations identified in mutation data.

write_reference_fasta *Write FASTA file of reference sequences*

Description

In some cases, you might want to generate an arbitrary multi-sequence FASTA file from GRanges including the reference sequences. This function allows you to do this.

Usage

```
write_reference_fasta(ref_ranges, fasta_out = "./reference_output.fasta")
```

Arguments

ref_ranges	A GRanges object including the sequences of the reference regions included for the data.
fasta_out	The path for the FASTA file output.

Value

Writes a FASTA reference file to be used in downstream applications.

---

write_VCF_from_mut *Write FASTA file of reference sequences*

Description

In some cases, you might want to generate an arbitrary multi-sequence FASTA file from GRanges including the reference sequences. This function allows you to do this.

Usage

```
write_VCF_from_mut(mutation_data, vcf_out = "./mutation_output.vcf")
```

Arguments

mutation_data	A GRanges object imported from a .mut file
vcf_out	The path for the VCF file output.

Value

Writes a VCF file of mutations to be used in downstream applications.

Index