Package ‘musicatk’

April 4, 2024

Type Package

Title Mutational Signature Comprehensive Analysis Toolkit

Version 1.12.0

Description Mutational signatures are carcinogenic exposures or aberrant cellular processes that can cause alterations to the genome. We created musicatk (MUtational SIgnature Comprehensive Analysis ToolKit) to address shortcomings in versatility and ease of use in other pre-existing computational tools. Although many different types of mutational data have been generated, current software packages do not have a flexible framework to allow users to mix and match different types of mutations in the mutational signature inference process. Musicatk enables users to count and combine multiple mutation types, including SBS, DBS, and indels. Musicatk calculates replication strand, transcription strand and combinations of these features along with discovery from unique and proprietary genomic feature associated with any mutation type. Musicatk also implements several methods for discovery of new signatures as well as methods to infer exposure given an existing set of signatures. Musicatk provides functions for visualization and downstream exploratory analysis including the ability to compare signatures between cohorts and find matching signatures in COSMIC V2 or COSMIC V3.

License LGPL-3

BugReports https://github.com/campbio/musicatk/issues

Encoding UTF-8

LazyData TRUE

biocViews Software, BiologicalQuestion, SomaticMutation, VariantAnnotation

Depends R (>= 4.0.0), NMF

Imports SummarizedExperiment, VariantAnnotation, Biostrings, base, methods, magrittr, tibble, tidyr, gtools, gridExtra, MCMCprecision, MASS, matrixTests, data.table, dplyr, rlang, BSgenome, GenomeInfoDb, GenomicFeatures, GenomicRanges, IRanges, S4Vectors, uwot, ggplot2, stringr, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Musculus.UCSC.mm9, BSgenome.Musculus.UCSC.mm10, deconstructSigs, decompTumor2Sig,
R topics documented:

topicmodels, ggrepel, plotly, utils, factoextra, cluster,
ComplexHeatmap, philentropy, mftools, shiny, stringi,
tidyverse, ggpubr

Suggests  TCGAbiolinks, shinyBS, shinyalert, shinybusy, shinydashboard,
shinyjs, shinygui, sortable, testthat, BiocStyle, knitr,
rmarkdown, survival, XVector, qpdf, covr, shinyWidgets,
cowplot, withr

RoxygenNote  7.2.3
VignetteBuilder  knitr
git_url  https://git.bioconductor.org/packages/musicatk
git_branch  RELEASE_3_18
git_last_commit  69a345e
git_last_commit_date  2023-10-24
Repository  Bioconductor 3.18
Date/Publication  2024-04-03

Author  Aaron Chevalier  [cre]  (0000-0002-3968-9250),
Joshua D. Campbell  [aut]  (<https://orcid.org/0000-0003-0780-8662>)
Maintainer  Aaron Chevalier  <atgc@bu.edu>
topics documented:

- cosmic_v3_sbs_sigs_exome
- count_table-class
- create_dbs78_table
- create_musica
- create_sbs192_table
- create_sbs96_table
- create_umap
- dbs_musica
- discover_signatures
- drop_annotation
- exposures
- exposure_differential_analysis
- extract_count_tables
- extract_variants
- extract_variants_from_maf
- extract_variants_from_maf_file
- extract_variants_from_matrix
- extract_variants_from_vcf
- extract_variants_from_vcf_file
- generate_result_grid
- get_musica
- indel_musica
- k_select
- musica
- musica-class
- musicatk
- musica_annot
- musica_result-class
- musica_result_grid-class
- musica_sbs96
- musica_sbs96_tiny
- name_signatures
- plot_cluster
- plot_differential_analysis
- plot_exposures
- plot_heatmap
- plot_sample_counts
- plot_sample_reconstruction_error
- plot_signatures
- plot_umap
- predict_exposure
- rc
- rep_range
- res
- res_annot
- sample_names
- samp_annot
- select_genome
.jsd

Calculates 1 - Jensen-Shannon Divergences between all pairs of columns between two matrices

Description

Calculates 1 - Jensen-Shannon Divergences between all pairs of columns between two matrices

Usage

.jsd(p, q, epsilon = 1e-07)

Arguments

p  First matrix
q  Second matrix
epsilon  Number to add to all probabilities. Default 0.0000001.

Value

Returns matrix of 1 - Jensen-Shannon Divergences
**add_flank_to_variants**  
*Uses a genome object to find context and add it to the variant table*

**Description**

Uses a genome object to find context and add it to the variant table

**Usage**

```r
add_flank_to_variants(
  musica,
  g,
  flank_start,
  flank_end,
  build_table = TRUE,
  overwrite = FALSE
)
```

**Arguments**

- **musica**: Input samples
- **g**: A BSgenome object indicating which genome reference the variants and their coordinates were derived from.
- **flank_start**: Start of flank area to add, can be positive or negative
- **flank_end**: End of flank area to add, can be positive or negative
- **build_table**: Automatically build a table using the annotation and add
- **overwrite**: Overwrite existing count table

**Value**

None it to the musica

**Examples**

```r
data(musica_sbs96_tiny)
g <- select_genome("19")
add_flank_to_variants(musica_sbs96_tiny, g, 1, 2)
add_flank_to_variants(musica_sbs96_tiny, g, -2, -1)
```
add_variant_type

Generates a variant type table

**Description**
Generates a variant type table

**Usage**
```
add_variant_type(tab)
```

**Arguments**
- **tab**: Input variant table

**Value**
Returns the inputted variant table with variant type ("SBS", "DBS", "INS", "DEL") added as an appended "Variant_Type" column

**Examples**
```
data(musica)
variants <- variants(musica)
musicatk:::add_variant_type(variants)
```

annotate_replication_strand

Add replication strand annotation to SBS variants based on bedgraph file

**Description**
Add replication strand annotation to SBS variants based on bedgraph file

**Usage**
```
annotate_replication_strand(musica, rep_range, build_table = TRUE)
```

**Arguments**
- **musica**: A musica object.
- **rep_range**: A GRanges object with replication timing as metadata
- **build_table**: Automatically build a table from this annotation
annotate_transcript_strand

Value
None

Examples

data(musica)
data(rep_range)
annotate_replication_strand(musica, rep_range)

annotate_transcript_strand
Add transcript strand annotation to SBS variants (defined in genes only)

Description
Add transcript strand annotation to SBS variants (defined in genes only)

Usage
annotate_transcript_strand(musica, genome_build, build_table = TRUE)

Arguments

  musica     A musica object.
  genome_build Which genome build to use: hg19, hg38, or a custom TxDb object
  build_table Automatically build a table from this annotation

Value
None

Examples

data(musica)
annotate_transcript_strand(musica, 19)
annotate_variant_length

Adds an annotation to the input musica's variant table with length of each variant

Description

Adds an annotation to the input musica's variant table with length of each variant

Usage

annotate_variant_length(musica)

Arguments

musica Input samples

Value

None

Examples

data(musica)
annotate_variant_length(musica)
musica

annotate_variant_type

Annotate variants with variant type ("SBS", "INS", "DEI", "DBS")

Description

Annotate variants with variant type ("SBS", "INS", "DEI", "DBS")

Usage

annotate_variant_type(musica)

Arguments

musica A musica object.

Value

None
Examples

    data(musica)
    annotate_variant_type(musica)

auto_predict_grid

Automatic filtering of signatures for exposure prediction gridded across specific annotation

Description

Automatic filtering of signatures for exposure prediction gridded across specific annotation

Usage

auto_predict_grid(
  musica,
  table_name,
  signature_res,
  algorithm,
  sample_annotation = NULL,
  min_exists = 0.05,
  proportion_samples = 0.25,
  rare_exposure = 0.4,
  verbose = TRUE,
  combine_res = TRUE
)

Arguments

  musica      Input samples to predict signature weights
  table_name  Name of table used for posterior prediction (e.g. SBS96)
  signature_res  Signatures to automatically subset from for prediction
  algorithm    Algorithm to use for prediction. Choose from "lda_posterior", decompTumor2Sig, and deconstructSigs
  sample_annotation  Annotation to grid across, if none given, prediction subsetting on all samples together
  min_exists  Threshold to consider a signature active in a sample
  proportion_samples  Threshold of samples to consider a signature active in the cohort
  rare_exposure  A sample will be considered active in the cohort if at least one sample has more than this threshold proportion
  verbose  Print current annotation value being predicted on
  combine_res  Automatically combines a list of annotation results into a single result object with zero exposure values for signatures not found in a given annotation’s set of samples
auto_subset_sigs

Value

A list of results, one per unique annotation value, if no annotation value is given, returns a single result for all samples, or combines into a single result if combines_res = TRUE

Examples

data(musica_annot)
data(cosmic_v2_sigs)
auto_predict_grid(musica = musica_annot, table_name = "SBS96",
signature_res = cosmic_v2_sigs, algorithm = "lda",
sample_annotation = "Tumor_Subtypes")
auto_predict_grid(musica_annot, "SBS96", cosmic_v2_sigs, "lda")

auto_subset_sigs

Automatic filtering of inactive signatures

Description

Automatic filtering of inactive signatures

Usage

auto_subset_sigs(
  musica,
  table_name,
  signature_res,
  algorithm,
  min_exists = 0.05,
  proportion_samples = 0.25,
  rare_exposure = 0.4
)

Arguments

musica  A musica object.
table_name  Name of table used for posterior prediction (e.g. SBS96)
signature_res  Signatures to automatically subset from for prediction
algorithm  Algorithm to use for prediction. Choose from "lda_posterior", decompTumor2Sig, and deconstructSigs
min_exists  Threshold to consider a signature active in a sample
proportion_samples  Threshold of samples to consider a signature active in the cohort
rare_exposure  A sample will be considered active in the cohort if at least one sample has more than this threshold proportion

Value

A result object containing automatically subset signatures and corresponding sample weights
**build_custom_table**  
Builds a custom table from specified user variants

**Description**

Builds a custom table from specified user variants

**Usage**

```r
build_custom_table(
  musica, 
  variant_annotation, 
  name, 
  description = character(), 
  data_factor = NA, 
  annotation_df = NULL, 
  features = NULL, 
  type = NULL, 
  color_variable = NULL, 
  color_mapping = NULL, 
  return_instead = FALSE, 
  overwrite = FALSE 
)
```

**Arguments**

- **musica**  
  A **musica** object.
- **variant_annotation**  
  User column to use for building table
- **name**  
  Table name to refer to (must be unique)
- **description**  
  Optional description of the table content
- **data_factor**  
  Full set of table values, in case some are missing from the data. If NA, a superset of all available unique data values will be used
- **annotation_df**  
  A data.frame of annotations to use for plotting
- **features**  
  A data.frame of the input data from which the count table will be built
- **type**  
  The type of data/mutation in each feature as an Rle object
- **color_variable**  
  The name of the column of annotation_df used for the coloring in plots
- **color_mapping**  
  The mapping from the values in the selected color_variable column to color values for plotting
- **return_instead**  
  Instead of adding to musica object, return the created table
- **overwrite**  
  Overwrite existing count table
Value

If return_instead = TRUE then the created table object is returned, otherwise the table object is automatically added to the musica’s count_tables list and nothing is returned.

Examples

data(musica)
anotate_transcript_strand(musica, "19", build_table = FALSE)
build_custom_table(musica, "Transcript_Strand", "Transcript_Strand",
data_factor = factor(c("T", "U")))

---

build_standard_table  Builds count tables using various mutation type schemas

Description

Generates count tables for different mutation type schemas which can be used as input to the mutational signature discovery or prediction functions. "SBS96" generates a table for single base substitutions following the standard 96 mutation types derived from the trinucleotide context. "SBS192" is the 96 mutation type schema with the addition of transcriptional strand or replication strand information added to each base. "DBS" generates a table for the double base substitution schema used in COSMIC V3. "Indel" generates a table for insertions and deletions following the schema used in COSMIC V3.

Usage

build_standard_table(
  musica,
g,
table_name,
strand_type = NULL,
overwrite = FALSE,
verbose = FALSE
)

Arguments

musica  A musica object.
g  A BSgenome object indicating which genome reference the variants and their coordinates were derived from.
table_name  Name of standard table to build. One of "SBS96", "SBS192", "DBS", or "Indel".
strand_type  Strand type to use in SBS192 schema. One of "Transcript_Strand" or "Replication_Strand". Only used if table_name = SBS192.
overwrite  If TRUE, any existing count table with the same name will be overwritten. If FALSE, then an error will be thrown if a table with the same name exists within the musica object.
verbose  Show progress bar for processed samples
built_tables

Value

No object will be returned. The count tables will be automatically added to the musica object.

Examples

```r
# Select and set genome
g <- select_genome("19")
data(musica)
build_standard_table(musica, g, "SBS96", overwrite = TRUE)
data(musica)
anotate_transcript_strand(musica, "19")
build_standard_table(musica, g, "SBS192", "Transcript_Strand")
data(musica)
data(rep_range)
anotate_replication_strand(musica, rep_range)
build_standard_table(musica, g, "SBS192", "Replication_Strand")
data(dbs_musica)
build_standard_table(dbs_musica, g, "DBS", overwrite = TRUE)
data(indel_musica)
build_standard_table(indel_musica, g, table_name = "INDEL")
```

---

built_tables

Retrieve the names of count_tables from a musica or musica_result object

Description

The count_tables contains standard and/or custom count tables created from variants

Usage

```r
built_tables(object)
```

## S4 method for signature 'musica'
built_tables(object)

## S4 method for signature 'musica_result'
built_tables(object)

Arguments

- **object**
  - A musica object generated by the create_musica function or a musica_result object generated by a mutational discovery or prediction tool.
cluster_exposure

Value

The names of created count_tables

Examples

data(res)
built_tables(res)

---

cluster_exposure  Perform clustering analysis from a musica result object

Description

Proportional sample exposures will be used as input to perform clustering.

Usage

ccluster_exposure(
  result,
  nclust,
  proportional = TRUE,
  method = "kmeans",
  dis.method = "euclidean",
  hc.method = "ward.D",
  clara.samples = 5,
  iter.max = 10,
  tol = 1e-15
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>A <code>musica_result</code> object generated by a mutational discovery or prediction tool.</td>
</tr>
<tr>
<td>nclust</td>
<td>Pre-defined number of clusters.</td>
</tr>
<tr>
<td>proportional</td>
<td>Logical, indicating if proportional exposure (default) will be used for clustering.</td>
</tr>
<tr>
<td>method</td>
<td>Clustering algorithms. Options are &quot;kmeans&quot; (K-means), &quot;hkmeans&quot; (hybrid of hierarchical K-means), &quot;hclust&quot; (hierarchical clustering), &quot;pam&quot; (PAM), and &quot;clara&quot; (Clara).</td>
</tr>
<tr>
<td>dis.method</td>
<td>Methods to calculate dissimilarity matrix. Options are &quot;euclidean&quot; (default), &quot;manhattan&quot;, &quot;jaccard&quot;, &quot;cosine&quot;, and &quot;canberra&quot;.</td>
</tr>
<tr>
<td>hc.method</td>
<td>Methods to perform hierarchical clustering. Options are &quot;ward.D&quot; (default), &quot;ward.D2&quot;, &quot;single&quot;, &quot;complete&quot;, &quot;average&quot;, &quot;mcquitty&quot;, &quot;median&quot;, and &quot;centroid&quot;.</td>
</tr>
<tr>
<td>clara.samples</td>
<td>Number of samples to be drawn from dataset. Only used when &quot;clara&quot; is selected. Default is 5.</td>
</tr>
<tr>
<td>iter.max</td>
<td>Maximum number of iterations for k-means clustering.</td>
</tr>
<tr>
<td>tol</td>
<td>Tolerance level for kmeans clustering level iterations</td>
</tr>
</tbody>
</table>
**Value**

A one-column data frame with sample IDs as row names and cluster number for each sample.

**See Also**

kmeans

**Examples**

```r
set.seed(123)
data(res_annot)
clust_out <- cluster_exposure(res_annot, nclust = 2)
```

---

**Description**

Combines tables into a single table that can be used for discovery/prediction

**Usage**

```r
combine_count_tables(
musica, 
to_comb, 
nname, 
description = character(), 
color_variable = character(), 
color_mapping = character(), 
overwrite = FALSE
)
```

**Arguments**

- **musica** A musica object.
- **to_comb** A vector of table names to combine. Each table must already exist within the input musica object.
- **name** Name of table build, must be a new name.
- **description** Description of the new table.
- **color_variable** Annotation column to use for coloring plotted motifs, provided by counts table from input result’s musica object.
- **color_mapping** Mapping from color_variable to color names, provided by counts table from input result’s musica object.
- **overwrite** Overwrite existing count table.
combine_predict_grid

Value

None

Examples

g <- select_genome("19")

data(musica)
build_standard_table(musica, g, "SBS96", overwrite = TRUE)

annotate_transcript_strand(musica, "19")
build_standard_table(musica, g, "SBS192", "Transcript_Strand")
combine_count_tables(musica, c("SBS96", "SBS192_Trans"), "combo")

combine_predict_grid(grid_list, musica, table_name, signature_res)

Description

Combine prediction grid list into a result object. Exposure values are zero for samples in an annotation where that signature was not predicted

Usage

combine_predict_grid(grid_list, musica, table_name, signature_res)

Arguments

grid_list A list of result objects from the prediction grid to combine into a single result
musica A musica object.
table_name Table name used for prediction
signature_res Signatures to automatically subset from for prediction

Value

A result object combining all samples and signatures from a prediction grid. Samples have zero exposure value for signatures not found in that annotation type.
**compare_cosmic_v2**

**Examples**

```r
data(musica_annot)
data(cosmic_v2_sigs)
grid <- auto_predict_grid(musica_annot, "SBS96", cosmic_v2_sigs, "lda",
"Tumor_Subtypes", combine_res = FALSE)
combined <- combine_predict_grid(grid, musica_annot, "SBS96", cosmic_v2_sigs)
plot_exposures(combined, group_by = "annotation",
annotation="Tumor_Subtypes")
```

**Description**

Compare a result object to COSMIC V2 SBS Signatures (combination whole-exome and whole-genome)

**Usage**

```r
compare_cosmic_v2(
  result,
  threshold = 0.9,
  metric = "cosine",
  result_name = deparse(substitute(result)),
  decimals = 2,
  same_scale = FALSE
)
```

**Arguments**

- `result` A `musica_result` object.
- `threshold` threshold for similarity
- `metric` One of "cosine" for cosine similarity or "jsd" for 1 minus the Jensen-Shannon Divergence. Default "cosine".
- `result_name` title for plot user result signatures
- `decimals` Specifies rounding for similarity metric displayed. Default 2.
- `same_scale` If TRUE, the scale of the probability for each signature will be the same. If FALSE, then the scale of the y-axis will be adjusted for each signature. Default TRUE.

**Value**

Returns the comparisons

**Examples**

```r
data(res)
compare_cosmic_v2(res, threshold = 0.7)
```
Compare a result object to COSMIC V3 Signatures; Select exome or genome for SBS and only genome for DBS or Indel classes

Usage

```r
compare_cosmic_v3(
  result,
  variant_class,
  sample_type,
  metric = "cosine",
  threshold = 0.9,
  result_name = deparse(substitute(result)),
  decimals = 2,
  same_scale = FALSE
)
```

Arguments

- **result**: A `musica_result` object.
- **variant_class**: Compare to SBS, DBS, or Indel
- **sample_type**: exome (SBS only) or genome
- **metric**: One of "cosine" for cosine similarity or "jsd" for 1 minus the Jensen-Shannon Divergence. Default "cosine".
- **threshold**: threshold for similarity
- **result_name**: title for plot user result signatures
- **decimals**: Specifies rounding for similarity metric displayed. Default 2.
- **same_scale**: If TRUE, the scale of the probability for each signature will be the same. If FALSE, then the scale of the y-axis will be adjusted for each signature. Default TRUE.

Value

Returns the comparisons

Examples

```r
data(res)
compare_cosmic_v3(res, "SBS", "genome", threshold = 0.8)
```
**compare_results**  
*Compare two result files to find similar signatures*

**Description**

Compare two result files to find similar signatures

**Usage**

```r
compare_results(
  result,
  other_result,
  threshold = 0.9,
  metric = "cosine",
  result_name = deparse(substitute(result)),
  other_result_name = deparse(substitute(other_result))
)
```

**Arguments**

- **result**: A `musica_result` object.
- **other_result**: A second `musica_result` object.
- **threshold**: threshold for similarity
- **metric**: One of "cosine" for cosine similarity or "jsd" for 1 minus the Jensen-Shannon Divergence. Default "cosine".
- **result_name**: title for plot of first result signatures
- **other_result_name**: title for plot of second result signatures

**Value**

Returns the comparisons

**Examples**

```r
data(res)
cmp <- compare_results(res, res, threshold = 0.8)
```
cosmic_v2_sigs  

**COSMIC v2 SBS96 Signatures Result Object**

**Description**

Data from COSMIC formatted to be used for prediction with individual tumors and cohorts.

**Usage**

```r
data(cosmic_v2_sigs)
```

**Format**

An object of class `musica_result` See [predict_exposure()].

**Source**

COSMIC v2, [https://cancer.sanger.ac.uk/cosmic/signatures_v2](https://cancer.sanger.ac.uk/cosmic/signatures_v2)

**References**


---

**cosmic_v2_subtype_map**  

**Input a cancer subtype to return a list of related COSMIC signatures**

**Description**

Input a cancer subtype to return a list of related COSMIC signatures.

**Usage**

```r
cosmic_v2_subtype_map(tumor_type)
```

**Arguments**

- `tumor_type`  
  Cancer subtype to view related signatures

**Value**

Returns signatures related to a partial string match.

**Examples**

```r
cosmic_v2_subtype_map("lung")
```
**cosmic_v3_dbs_sigs**  
*COSMIC v3 DBS Genome Signatures Result Object*

**Description**
Data from COSMIC formatted to be used for prediction with individual tumors and cohorts.

**Usage**
```r
data(cosmic_v3_dbs_sigs)
```

**Format**
An object of class `musica_result`. See `[predict_exposure()]`.

**Source**
COSMIC v3, <https://cancer.sanger.ac.uk/cosmic/signatures>

**References**

---

**cosmic_v3_indel_sigs**  
*COSMIC v3 Indel Genome Signatures Result Object*

**Description**
Data from COSMIC formatted to be used for prediction with individual tumors and cohorts.

**Usage**
```r
data(cosmic_v3_indel_sigs)
```

**Format**
An object of class `musica_result`. See `[predict_exposure()]`.

**Source**
COSMIC v3, <https://cancer.sanger.ac.uk/cosmic/signatures>

**References**
**cosmic_v3_sbs_sigs**  
COSMIC v3 SBS96 Genome Signatures Result Object

**Description**
Data from COSMIC formatted to be used for prediction with individual tumors and cohorts.

**Usage**
```
data(cosmic_v3_sbs_sigs)
```

**Format**
An object of class `musica_result`. See `predict_exposure()`.

**Source**
COSMIC v3, <https://cancer.sanger.ac.uk/cosmic/signatures>

**References**

**cosmic_v3_sbs_sigs_exome**  
COSMIC v3 SBS96 Exome Signatures Result Object

**Description**
Data from COSMIC formatted to be used for prediction with individual tumors and cohorts.

**Usage**
```
data(cosmic_v3_sbs_sigs_exome)
```

**Format**
An object of class `musica_result`. See `predict_exposure()`.

**Source**
COSMIC v3, <https://cancer.sanger.ac.uk/cosmic/signatures>

**References**
count_table-class

Object containing the count table matrices, their names and descriptions that we generated by provided and by user functions. These are used to discover and infer signatures and exposures.

Description

Object containing the count table matrices, their names and descriptions that we generated by provided and by user functions. These are used to discover and infer signatures and exposures.

Slots

- name: A name that describes the type of table (e.g. "SBS96")
- count_table: An array of counts with samples as the columns and motifs as the rows
- annotation: A data.frame of annotations with three columns used for plotting: motif, mutation, and context
- features: Original features used to generate the count_table
- type: The mutation type of each feature, in case we need to plot or model them differently
- color_variable: The variable used for plotting colors, selected from the annotation slot
- color_mapping: The mapping of the annotations chosen by color_variable to color values for plotting
- description: A summary table of the result objects in result_list a list of lists. The nested lists created combined (rbind) tables, and the tables at the first list level are modelled independently. Combined tables must be named. list("tableA", comboTable = list("tableC", "tableD"))

create_dbs78_table

Creates and adds a table for standard doublet base substitution (DBS)

Description

Creates and adds a table for standard doublet base substitution (DBS)

Usage

create_dbs78_table(musica, overwrite = overwrite, verbose)

Arguments

- musica: A musica object.
- overwrite: Overwrite existing count table

Value

Returns the created DBS table object
**create_musica**

*Creates a musica object from a variant table*

**Description**

This function creates a musica object from a variant table or matrix. The musica class stores variants information, variant-level annotations, sample-level annotations, and count tables and is used as input to the mutational signature discovery and prediction algorithms. The input variant table or matrix must have columns for chromosome, start position, end position, reference allele, alternate allele, and sample names. The column names in the variant table can be mapped using the chromosome_col, start_col, end_col, ref_col, alt_col, and sample_col parameters.

**Usage**

```r
create_musica(
  x, genome,
  check_ref_chromosomes = TRUE,
  check_ref_bases = TRUE,
  chromosome_col = "chr",
  start_col = "start",
  end_col = "end",
  ref_col = "ref",
  alt_col = "alt",
  sample_col = "sample",
  extra_fields = NULL,
  standardize_indels = TRUE,
  convert_dbs = TRUE,
  verbose = TRUE
)
```

**Arguments**

- **x**: A data.table, matrix, or data.frame that contains columns with the variant information.
- **genome**: A BSgenome object indicating which genome reference the variants and their coordinates were derived from.
- **check_ref_chromosomes**: Whether to perform a check to ensure that the chromosomes in the variant object match the reference chromosomes in the genome object. If there are mismatches, this may cause errors in downstream generation of count tables. If mismatches occur, an attempt to automatically fix these with the seqlevelsStyle function will be made. Default TRUE.
- **check_ref_bases**: Whether to check if the reference bases in the variant object match the reference bases in the genome object. Default TRUE.
chromosome_col  The name of the column that contains the chromosome reference for each variant. Default "chr".
start_col     The name of the column that contains the start position for each variant. Default "start".
end_col       The name of the column that contains the end position for each variant. Default "end".
ref_col       The name of the column that contains the reference base(s) for each variant. Default "ref".
alt_col       The name of the column that contains the alternative base(s) for each variant. Default "alt".
sample_col    The name of the column that contains the sample id for each variant. Default "sample".
extra_fields  Which additional fields to extract and include in the musica object. Default NULL.
standardize_indels Flag to convert indel style (e.g. 'C > CAT' becomes '- > AT' and 'GCACA > G' becomes 'CACA > -')
convert_dbs   Flag to convert adjacent SBS into DBS (original SBS are removed)
verbose       Whether to print status messages during error checking. Default TRUE.

Value

Returns a musica object

Examples

maf_file <- system.file("extdata", "public_TCGA.LUSC.maf", package = "musicatk")
variants <- extract_variants_from_maf_file(maf_file)
g <- select_genome("38")
musica <- create_musica(x = variants, genome = g)

create_sbs192_table  Uses a genome object to find context and generate standard SBS192 table using transcript strand

Description

Uses a genome object to find context and generate standard SBS192 table using transcript strand

Usage

create_sbs192_table(musica, g, strand_type, overwrite = FALSE)
create_sbs96_table

**Arguments**

- **musica**
  - Input samples
- **g**
  - A BSgenome object indicating which genome reference the variants and their coordinates were derived from.
- **strand_type**
  - Transcript_Strand or Replication_Strand
- **overwrite**
  - Overwrite existing count table

**Value**

Returns the created SBS192 count table object built using either transcript strand or replication strand.

---

**Description**

Uses a genome object to find context and generate standard SBS96 tables.

**Usage**

```r
create_sbs96_table(musica, g, overwrite = FALSE)
```

**Arguments**

- **musica**
  - A musica object.
- **g**
  - A BSgenome object indicating which genome reference the variants and their coordinates were derived from.
- **overwrite**
  - Overwrite existing count table

**Value**

Returns the created SBS96 count table object.
create_umap

Create a UMAP from a musica result

Description

Proportional sample exposures will be used as input into the `umap` function to generate a two-dimensional UMAP.

Usage

```r
create_umap(result, n_neighbors = 30, min_dist = 0.75, spread = 1)
```

Arguments

- **result**: A `musica_result` object generated by a mutational discovery or prediction tool.
- **n_neighbors**: The size of local neighborhood used for views of manifold approximation. Larger values result in more global the manifold, while smaller values result in more local data being preserved. If `n_neighbors` is larger than the number of samples, then `n_neighbors` will automatically be set to the number of samples in the `musica_result`. Default 30.
- **min_dist**: The effective minimum distance between embedded points. Smaller values will result in a more clustered/clumped embedding where nearby points on the manifold are drawn closer together, while larger values will result on a more even dispersal of points. Default 0.2.
- **spread**: The effective scale of embedded points. In combination with `min_dist`, this determines how clustered/clumped the embedded points are. Default 1.

Value

A `musica_result` object with a new UMAP stored in the UMAP slot.

See Also

See `plot_umap` to display the UMAP and `umap` for more information on the individual parameters for generating UMAPs.

Examples

```r
data(res_annot)
create_umap(result = res_annot)
```
**dbs_musica**

**Description**

A musica created for testing that includes DBS variants

**Usage**

```r
data(dbs_musica)
```

**Format**

An object of class musica See [create_musica()].

---

**discover_signatures**

**Description**

Mutational signatures and exposures will be discovered using methods such as Latent Dirichlet Allocation (lda) or Non-Negative Matrix Factorization (nnmf). These algorithms will deconvolute a matrix of counts for mutation types in each sample to two matrices: 1) a "signature" matrix containing the probability of each mutation type in each sample and 2) an "exposure" matrix containing the estimated counts for each signature in each sample. Before mutational discovery can be performed, variants from samples first need to be stored in a musica object using the create_musica function and mutation count tables need to be created using functions such as build_standard_table.

**Usage**

```r
discover_signatures(
  musica,
  table_name,
  num_signatures,
  algorithm = "lda",
  seed = 1,
  nstart = 10,
  par_cores = 1
)
```
**drop_annotation**

**Description**

Drops a column from the variant table that the user no longer needs.

**Usage**

```r
drop_annotation(musica, column_name)
```

**Arguments**

- `musica`: A `musica` object.
- `column_name`: Name of column to drop.

**Value**

- None

---

**Argument Descriptions**

- **musica**: A `musica` object.
- **table_name**: Name of the table to use for signature discovery. Needs to be the same name supplied to the table building functions such as `build_standard_table`.
- **num_signatures**: Number of signatures to discover.
- **algorithm**: Method to use for mutational signature discovery. One of "lda" or "nmf". Default "lda".
- **seed**: Seed to be used for the random number generators in the signature discovery algorithms. Default 1.
- **nstart**: Number of independent random starts used in the mutational signature algorithms. Default 10.
- **par_cores**: Number of parallel cores to use. Only used if method = "nmf". Default 1.

**Value**

Returns a `musica_result` object containing signatures and exposures.

**Examples**

```r
data(musica)
g <- select_genome("19")
build_standard_table(musica, g, "SBS96", overwrite = TRUE)
discover_signatures(musica = musica, table_name = "SBS96", num_signatures = 3, algorithm = "lda", seed = 12345, nstart = 1)
```
Examples

data(musica)
drop_annotation(musica, "Variant_Type")

Description

The exposure matrix contains estimated amount of each signature for each sample. Rows correspond to each signature and columns correspond to each sample.

Usage

exposures(result)

## S4 method for signature 'musica_result'
exposures(result)
exposures(result) <- value

## S4 replacement method for signature 'musica_result,matrix'
exposures(result) <- value

Arguments

result       A musica_result object generated by a mutational discovery or prediction tool.
value        A matrix of samples by signature exposures

Value

A matrix of exposures

Examples

data(res)
exposures(res)
data(res)
exposures(res) <- matrix()
exposure_differential_analysis

*Compare exposures of annotated samples*

**Description**

`exposure_differential_analysis` is used to run differential analysis on the signature exposures of annotated samples within the `musica_result` object.

**Usage**

```r
exposure_differential_analysis(
  musica_result,
  annotation,
  method = c("wilcox", "kruskal", "glm.nb"),
  group1 = NULL,
  group2 = NULL,
  ...
)
```

**Arguments**

- `musica_result`: A `musica_result` object
- `annotation`: Column in the sample_annotations table of the `musica_result` object
- `method`: Any method in `c("wilcox", "kruskal", "glm.nb")` used to perform differential analysis on signature exposures
- `group1`: character vector used in the Wilcoxon test. Elements in `group1` are compared to elements in `group2`. This is required for `annotation` with more than 2 levels.
- `group2`: character vector used in the Wilcoxon test. Elements in `group2` are compared to elements in `group1`. This is required for `annotation` with more than 2 levels.
- `...`: Additional arguments to be passed to the chosen method

**Value**

A matrix containing statistics summarizing the analysis dependent on the chosen method

**Examples**

```r
data("res_annot")
exposure_differential_analysis(res_annot, "Tumor_Subtypes", method="wilcox")
```
**extract_count_tables**  
*Extract count tables list from a musica object*

**Description**
Extract count tables list from a musica object

**Usage**
```
extract_count_tables(musica)
```

**Arguments**
- `musica` A `musica` object.

**Value**
List of count tables objects

**Examples**
```
data(musica)
extract_count_tables(musica)
```

---

**extract_variants**  
*Extract variants from multiple objects*

**Description**
Chooses the correct function to extract variants from input based on the class of the object or the file extension. Different types of objects can be mixed within the list. For example, the list can include VCF files and maf objects. Certain parameters such as `id` and `rename` only apply to VCF objects or files and need to be individually specified for each VCF. Therefore, these parameters should be supplied as a vector that is the same length as the number of inputs. If other types of objects are in the input list, then the value of `id` and `rename` will be ignored for these items.

**Usage**
```
extract_variants(
  inputs,
  id = NULL,
  rename = NULL,
  sample_field = NULL,
  filename_as_id = FALSE,
  strip_extension = c(".vcf", "vcf.gz", ".gz"),
  filter = TRUE,
)```
extract_variants = c("expand", "exclude"),
fix_vcf_errors = TRUE,
extra_fields = NULL,
chromosome_col = "chr",
start_col = "start",
end_col = "end",
ref_col = "ref",
alt_col = "alt",
sample_col = "sample",
verbose = TRUE
)

Arguments

inputs A vector or list of objects or file names. Objects can be CollapsedVCF, ExpandedVCF, MAF, an object that inherits from matrix or data.frame, or character strings that denote the path to a vcf or maf file.

id A character vector the same length as inputs denoting the sample to extract from a vcf. See extract_variants_from_vcf for more details. Only used if the input is a vcf object or file. Default NULL.

rename A character vector the same length as inputs denoting what the same will be renamed to. See extract_variants_from_vcf for more details. Only used if the input is a vcf object or file. Default NULL.

sample_field Some algorithms will save the name of the sample in the ##SAMPLE portion of header in the VCF. See extract_variants_from_vcf for more details. Default NULL.

filename_as_id If set to TRUE, the file name will be used as the sample name. See extract_variants_from_vcf_file for more details. Only used if the input is a vcf file. Default TRUE.

strip_extension Only used if filename_as_id is set to TRUE. If set to TRUE, the file extension will be stripped from the filename before setting the sample name. See extract_variants_from_vcf_file for more details. Only used if the input is a vcf file. Default "vcf", "vcf.gz", "gz"

filter Exclude variants that do not have a PASS in the FILTER column of VCF inputs.

multiallele Multialleles are when multiple alternative variants are listed in the same row in the vcf. See extract_variants_from_vcf for more details. Only used if the input is a vcf object or file. Default "expand".

fix_vcf_errors Attempt to automatically fix VCF file formatting errors. See extract_variants_from_vcf_file for more details. Only used if the input is a vcf file. Default TRUE.

extra_fields Optionally extract additional fields from all input objects. Default NULL.

chromosome_col The name of the column that contains the chromosome reference for each variant. Only used if the input is a matrix or data.frame. Default "Chromosome".

start_col The name of the column that contains the start position for each variant. Only used if the input is a matrix or data.frame. Default "Start_Position".

end_col The name of the column that contains the end position for each variant. Only used if the input is a matrix or data.frame. Default "End_Position".
ref_col: The name of the column that contains the reference base(s) for each variant. Only used if the input is a matrix or data.frame. Default "Tumor_Seq_Allele1".

alt_col: The name of the column that contains the alternative base(s) for each variant. Only used if the input is a matrix or data.frame. Default "Tumor_Seq_Allele2".

col: The name of the column that contains the sample id for each variant. Only used if the input is a matrix or data.frame. Default "sample".

Value

Returns a data.table of variants from a vcf.

Examples

# Get locations of two vcf files and a maf file
luad_vcf_file <- system.file("extdata", "public_LUAD_TCGA-97-7938.vcf", package = "musicatk")
lusc_maf_file <- system.file("extdata", "public_TCGA.LUSC.maf", package = "musicatk")
melanoma_vcfs <- list.files(system.file("extdata", package = "musicatk"), pattern = glob2rx("*SKCM*vcf"), full.names = TRUE)

# Read all files in at once
inputs <- c(luad_vcf_file, melanoma_vcfs, lusc_maf_file)
variants <- extract_variants(inputs = inputs)
table(variants$sample)

# Run again but renaming samples in first four vcfs
new_name <- c(paste0("Sample", 1:4), NA)
variants <- extract_variants(inputs = inputs, rename = new_name)
table(variants$sample)
extract_variants_from_maf_file

Value

Returns a data.table of variants from a maf which can be used to create a musica object.

Examples

```r
maf_file <- system.file("extdata", "public_TCGA.LUSC.maf", 
package = "musicatk")
library(maftools)
maf <- read.maf(maf_file)
variants <- extract_variants_from_maf(maf = maf)
```

extract_variants_from_maf_file

*Extracts variants from a maf file*

Description

Add Description - Aaron

Usage

```r
extract_variants_from_maf_file(maf_file, extra_fields = NULL)
```

Arguments

- **maf_file**: Location of maf file
- **extra_fields**: Optionally extract additional columns from the object. Default NULL.

Value

Returns a data.table of variants from a maf

Examples

```r
maf_file <- system.file("extdata", "public_TCGA.LUSC.maf", 
package = "musicatk")
maf <- extract_variants_from_maf_file(maf_file = maf_file)
```
extract_variants_from_matrix

Extract variants from matrix or data.frame like objects

Description

Add Description

Usage

```r
extract_variants_from_matrix(
  mat,
  chromosome_col = "Chromosome",
  start_col = "Start_Position",
  end_col = "End_Position",
  ref_col = "Tumor_Seq_Allele1",
  alt_col = "Tumor_Seq_Allele2",
  sample_col = "Tumor_Sample_Barcode",
  extra_fields = NULL
)
```

Arguments

- **mat**: An object that inherits from classes "matrix" or "data.frame" Examples include a matrix, data.frame, or data.table.
- **chromosome_col**: The name of the column that contains the chromosome reference for each variant. Default "Chromosome".
- **start_col**: The name of the column that contains the start position for each variant. Default "Start_Position".
- **end_col**: The name of the column that contains the end position for each variant. Default "End_Position".
- **ref_col**: The name of the column that contains the reference base(s) for each variant. Default "Tumor_Seq_Allele1".
- **alt_col**: The name of the column that contains the alternative base(s) for each variant. Default "Tumor_Seq_Allele2".
- **sample_col**: The name of the column that contains the sample id for each variant. Default "Tumor_Sample_Barcode".
- **extra_fields**: Optionally extract additional columns from the object. Default NULL.

Value

Returns a data.table of variants from a maf which can be used to create a musica object.
## extract_variants_from_vcf

`extract_variants_from_vcf()` extracts variants from a VariantAnnotation VCF object.

### Examples

```r
maf_file <- system.file("extdata", "public_TCGA.LUSC.maf", package = "musicatk")
library(maftools)
maf <- read.maf(maf_file)
variants <- extract_variants_from_maf(maf = maf)
variants <- extract_variants_from_matrix(mat = variants, chromosome_col = "chr", start_col = "start", end_col = "end", ref_col = "ref", alt_col = "alt", sample_col = "sample")
```

### Description

Aaron - Need to describe difference between ID, and name in the header, and rename in terms of naming the sample. Need to describe differences in multiallelic choices. Also need to describe the automatic error fixing.

### Usage

```r
extract_variants_from_vcf(
  vcf,
  id = NULL,
  rename = NULL,
  sample_field = NULL,
  filter = TRUE,
  multiallele = c("expand", "exclude"),
  extra_fields = NULL
)
```

### Arguments

- **vcf**: Location of vcf file
- **id**: ID of the sample to select from VCF. If NULL, then the first sample will be selected. Default NULL.
- **rename**: Rename the sample to this value when extracting variants. If NULL, then the sample will be named according to ID.
- **sample_field**: Some algorithms will save the name of the sample in the ##SAMPLE portion of header in the VCF (e.g. ##SAMPLE=<ID=TUMOR,SampleName=TCGA-01-0001>). If the ID is specified via the id parameter ("TUMOR" in this example), then sample_field can be used to specify the name of the tag ("SampleName" in this example). Default NULL.
- **filter**: Exclude variants that do not have a PASS in the FILTER column of the VCF. Default TRUE.
multiallele | Multialleles are when multiple alternative variants are listed in the same row in the vcf. One of "expand" or "exclude". If "expand" is selected, then each alternate allele will be given their own rows. If "exclude" is selected, then these rows will be removed. Default "expand".

extra_fields | Optionally extract additional fields from the INFO section of the VCF. Default NULL.

Value

Returns a data.table of variants from a vcf

Examples

```r
cvf_file <- system.file("extdata", "public_LUAD_TCGA-97-7938.vcf", package = "musicatk")
library(VariantAnnotation)
vcf <- readVcf(vcf_file)
variants <- extract_variants_from_vcf(vcf = vcf)
```

extract_variants_from_vcf_file

*Extracts variants from a vcf file*

Description

Add Description

Usage

```r
extract_variants_from_vcf_file(
  vcf_file,
  id = NULL,
  rename = NULL,
  sample_field = NULL,
  filename_as_id = FALSE,
  strip_extension = c(".vcf", ".vcf.gz", ".gz"),
  filter = TRUE,
  multiallele = c("expand", "exclude"),
  extra_fields = NULL,
  fix_vcf_errors = TRUE
)
```
Arguments

vcf_file Path to the vcf file
id ID of the sample to select from VCF. If NULL, then the first sample will be selected. Default NULL.
rename Rename the sample to this value when extracting variants. If NULL, then the sample will be named according to ID.
sample_field Some algorithms will save the name of the sample in the ##SAMPLE portion of header in the VCF (e.g. ##SAMPLE=<ID=TUMOR,SampleName=TCGA-01-0001>). If the ID is specified via the id parameter ("TUMOR" in this example), then sample_field can be used to specify the name of the tag ("SampleName" in this example). Default NULL.
filename_as_id If set to TRUE, the file name will be used as the sample name.
strip_extension Only used if filename_as_id is set to TRUE. If set to TRUE, the file extension will be stripped from the filename before setting the sample name. If a character vector is given, then all the strings in the vector will removed from the end of the filename before setting the sample name. Default c(".vcf",".vcf.gz",".gz")
filter Exclude variants that do not have a PASS in the FILTER column of the VCF. Default TRUE.
multiallele Multialleles are when multiple alternative variants are listed in the same row in the vcf. One of "expand" or "exclude". If "expand" is selected, then each alternate allele will be given their own rows. If "exclude" is selected, then these rows will be removed. Default "expand".
extra_fields Optionally extract additional fields from the INFO section of the VCF. Default NULL.
fix_vcf_errors Attempt to automatically fix VCF file formatting errors.

Value

Returns a data.table of variants extracted from a vcf

Examples

vcf <- system.file("extdata", "public_LUAD_TCGA-97-7938.vcf",
    package = "musicatk")
variants <- extract_variants_from_vcf_file(vcf_file = vcf)

Description

Generate result_grid from musica based on annotation and range of k
generate_result_grid

Usage

```r
generate_result_grid(
  musica,
  table_name,
  algorithm = "lda",
  annotation = NA,
  k_start,
  k_end,
  n_start = 1,
  seed = NULL,
  par_cores = FALSE,
  verbose = FALSE
)
```

Arguments

- **musica**: A `musica` object.
- **table_name**: Name of table used for signature discovery.
- **algorithm**: Algorithm for signature discovery.
- **annotation**: Sample annotation to split results into.
- **k_start**: Lower range of number of signatures for discovery.
- **k_end**: Upper range of number of signatures for discovery.
- **n_start**: Number of times to discover signatures and compare based on posterior loglikelihood.
- **seed**: Seed to use for reproducible results, set to null to disable.
- **par_cores**: Number of parallel cores to use (NMF only).
- **verbose**: Whether to output loop iterations.

Value

A result object containing signatures and sample weights.

Examples

```r
data(musica_sbs96)
grid <- generate_result_grid(musica_sbs96, "SBS96", "lda", k_start = 2, k_end = 5)
```
### get_musica

**Retrieve musica from a musica_result object**

#### Description

The musica musica contains variants, count tables, and sample annotations.

#### Usage

```r
get_musica(result)
```

```r
## S4 method for signature 'musica_result'
get_musica(result)
```

#### Arguments

- `result`: A musica_result object generated by a mutational discovery or prediction tool.

#### Value

A musica musica object.

#### Examples

```r
data(res)
get_musica(res)
```

---

### indel_musica

**indel_musica**

#### Description

A musica created for testing that includes INDEL variants.

#### Usage

```r
data(indel_musica)
```

#### Format

An object of class musica See [create_musica()].
**k_select**

*Plots for helping decide number of clusters*

**Description**

To help decide the number of cluster, three different methods are provided: total within cluster sum of squares, average silhouette coefficient, and gap statistics.

**Usage**

```r
k_select(
  result,
  method = "wss",
  clust.method = "kmeans",
  n = 10,
  proportional = TRUE
)
```

**Arguments**

- `result`: A `musica_result` object generated by a mutational discovery or prediction tool.
- `method`: A single character string indicating which statistic to use for plot. Options are "wss" (total within cluster sum of squares), "silhouette" (average silhouette coefficient), and "gap_stat" (gap statistic). Default is "wss".
- `clust.method`: A character string indicating clustering method. Options are "kmeans" (default), "hclust" (hierarchical clustering), "hkmeans", "pam", and "clara".
- `n`: An integer indicating maximum number of clusters to test. Default is 10.
- `proportional`: Logical, indicating if proportional exposure (default) will be used for clustering.

**Value**

A ggplot object.

**See Also**

- `fviz_nbclust`

**Examples**

```r
data(res_annot)
set.seed(123)
# Make an elbow plot
k_select(res_annot, method = "wss", n = 6)
# Plot average silhouette coefficient against number of clusters
k_select(res_annot, method = "silhouette", n = 6)
# Plot gap statistics against number of clusters
k_select(res_annot, method = "gap_stat", n = 6)
```
**musica**

**Description**

A musica created for testing that includes SBS variants

**Usage**

data(musica)

**Format**

An object of class `musica` See [create_musica()].

**musica-class**

The primary object that contains variants, count_tables, and samples annotations

**Description**

The primary object that contains variants, count_tables, and samples annotations

**Slots**

- variants: data.table of variants
- count_tables: Summary table with per-sample unnormalized motif counts
- sample_annotations: Sample-level annotations (e.g. age, sex, primary)

**musicatk**

Starts the musicatk interactive Shiny app

**Description**

The musicatk Shiny app allows users to perform mutational signature analysis using an interactive graphical user interface (GUI)

**Usage**

musicatk(include_version = TRUE, theme = "yeti")
Arguments

- include_version
  - Include the version number in the header. Default TRUE.
- theme
  - The theme to use for the GUI. Default "yeti".

Value

The shiny app will open. No data will be returned.

Examples

```r
## Not run:
# Start the app
musicatk()

## End(Not run)
```

Description

A musica created for testing that includes SBS variants and sample annotations

Usage

data(musica_annot)

Format

An object of class musica See [create_musica()].

Description

Object containing deconvolved/predicted signatures, sample weights, and the musica object the result was generated from
**musica_result_grid-class**

**Slots**

- **signatures** A matrix of signatures by mutational motifs
- **exposures** A matrix of samples by signature weights
- **table_name** A character vector of table names used to make the result
- **algorithm** Describes how the signatures/weights were generated
- **musica** The musica object the results were generated from
- **umap** List of umap data.frames for plotting and analysis

---

**Description**

Object containing the result objects generated from the combination of annotations and a range of k values

**Slots**

- **grid_params** The parameters the result grid was created using
- **result_list** A list of result objects with different parameters
- **grid_table** A summary table of the result objects in result_list

---

**musica_sbs96**

**Description**

A musica created for testing that includes SBS variants and a build counts table for them

**Usage**

data(musica_sbs96)

**Format**

An object of class `musica` See `[build_standard_table()]`. 
**Description**

A very small musica created for testing that includes SBS variants and a build counts table for them.

**Usage**

```r
data(musica_sbs96_tiny)
```

**Format**

An object of class musica. See `build_standard_table()`.

---

**name_signatures**

Return sample from musica object

**Description**

Return sample from musica object

**Usage**

```r
name_signatures(result, name_vector)
```

**Arguments**

- `result`: Result object containing signatures and weights
- `name_vector`: Vector of user-defined signature names

**Value**

Result object with user-defined signatures names

**Examples**

```r
data(res)
name_signatures(res, c("smoking", "apobec", "unknown"))
```
plot_cluster

Visualize clustering results

Description
The clustering results can be visualized on a UMAP panel. Three different types of plots can be generated using this function: cluster-by-signature plot, cluster-by-annotation plot, and a single UMAP plot.

Usage

```r
plot_cluster(
  result,
  clusters,
  group = "signature",
  annotation = NULL,
  plotly = TRUE
)
```

Arguments

- `result`: A `musica_result` object generated by a mutational discovery or prediction tool. A two-dimensional UMAP has to be stored in this object.
- `clusters`: The result generated from `cluster_exposure` function.
- `group`: A single character string indicating the grouping factor. Possible options are: "signature" (columns are signatures in a grid), "annotation" (columns are sample annotation), and "none" (a single UMAP plot). Default is "signature".
- `annotation`: Column name of annotation.
- `plotly`: If TRUE, the plot will be made interactive using plotly.

Value
Generate a ggplot or plotly object.

See Also

`create_umap`

Examples

```r
set.seed(123)
data(res_annot)
# Get clustering result
clust_out <- cluster_exposure(result = res_annot, nclust = 2)
# UMAP
create_umap(result = res_annot)
# generate cluster X signature plot
```
plot_differential_analysis

Description

plot_differential_analysis is used to plot differential analysis created by exposure_differential_analysis.

Usage

plot_differential_analysis(analysis, analysis_type, samp_num)

Arguments

analysis Analysis created by exposure_differential_analysis
analysis_type Currently only "glm" supported
samp_num Number of samples that went into the analysis

Value

Generates a ggplot object

Examples

data("res_annot")
analysis <- exposure_differential_analysis(res_annot, "Tumor_Subtypes", method="wilcox")
plot_differential_analysis(analysis, "glm", 2)
plot_exposures

Display sample exposures with bar, box, or violin plots

Description

The distributions of mutational signatures can be viewed with barplots or box/violin plots. Barplots are most useful for viewing the proportion of signatures within and across samples. The box/violin plots are most useful for viewing the distributions of signatures with respect to sample annotations. Samples can be grouped using the group_by parameter. For barplots, various methods of sorting samples from left to right can be chosen using the sort_samples parameter.

Usage

plot_exposures(
  result,
  plot_type = c("bar", "box", "violin"),
  proportional = FALSE,
  group_by = "none",
  color_by = c("signature", "annotation"),
  annotation = NULL,
  num_samples = NULL,
  sort_samples = "total",
  threshold = NULL,
  same_scale = FALSE,
  add_points = FALSE,
  point_size = 2,
  label_x_axis = FALSE,
  legend = TRUE,
  plotly = FALSE
)

Arguments

result A musica_result object generated by a mutational discovery or prediction tool.
plot_type One of "bar", "box", or "violin". Default "bar".
proportional If TRUE, then the exposures will be normalized to between 0 and 1 by dividing by the total number of counts for each sample. Default FALSE.
group_by Determines how to group samples into the subplots (i.e. facets). One of "none", "signature" or "annotation". If set to "annotation", then a sample annotation must be supplied via the annotation parameter. Default "none".
color_by Determines how to color the bars or box/violins. One of "signature" or "annotation". If set to "annotation", then a sample annotation must be supplied via the annotation parameter. Default "signature".
annotation Sample annotation used to group the subplots and/or color the bars, boxes, or violins. Default NULL.
num_samples: The top number of sorted samples to display. If NULL, then all samples will be displayed. If group_by is set, then the top samples will be shown within each group. Default NULL.

sort_samples: This is used to change how samples are sorted in the barplot from left to right. If set to "total", then samples will be sorted from those with the highest number of mutation counts to the lowest (regardless of how the parameter "proportional" is set). If set to "name", then samples are sorted by their name with the mixedsort function. If set to one or more signature names (e.g. "Signature1"), then samples will be sorted from those with the highest level of that signature to the lowest. If multiple signatures are supplied then, samples will be sorted by each signature sequentially. Default "total".

threshold: Exposures less than this threshold will be set to 0. This is most useful when more than one signature is supplied to sort_samples as samples that are set to zero for the first exposure will then be sorted by the levels of the second exposure. Default NULL.

same_scale: If TRUE, then all subplots will have the same scale. Only used when group_by is set. Default FALSE.

add_points: If TRUE, then points for individual sample exposures will be plotted on top of the violin/box plots. Only used when plot_type is set to "violin" or "box". Default TRUE.

point_size: Size of the points to be plotted on top of the violin/box plots. Only used when plot_type is set to "violin" or "box" and add_points is set to TRUE. Default 2.

label_x_axis: If TRUE, x-axis labels will be displayed at the bottom of the plot. Default FALSE.

legend: If TRUE, the legend will be displayed. Default TRUE.

plotly: If TRUE, the the plot will be made interactive using plotly. Default FALSE.

Value
Generates a ggplot or plotly object

Examples
```r
data(res_annot)
plot_exposures(res_annot, plot_type = "bar", annotation = "Tumor_Subtypes")
```

Description
The exposures for different signatures can be visualized using a heatmap with this function. Heatmaps make it easier to visualize the data by representing the magnitude of exposure values as color in 2-dimensions. The variation in color intensity can help see if the exposures are clustered or how they vary over space. Exposures can be normalized by providing the proportional argument. Column annotations can also be seen by passing the col_annot argument.
**plot_heatmap**

Usage

\[
\text{plot_heatmap}( \\
\quad \text{res}\_\text{annot}, \\
\quad \text{proportional} = \text{FALSE}, \\
\quad \text{show}\_\text{column}\_\text{names} = \text{FALSE}, \\
\quad \text{show}\_\text{row}\_\text{names} = \text{TRUE}, \\
\quad \text{scale} = \text{TRUE}, \\
\quad \text{subset}\_\text{tumor} = \text{NULL}, \\
\quad \text{subset}\_\text{signatures} = \text{NULL}, \\
\quad \text{annotation} = \text{NULL}, \\
\quad \ldots
\)
\]

Arguments

- **res_annot** A *musica_result* object generated by a mutational discovery or prediction tool.
- **proportional** If TRUE, then the exposures will be normalized to between 0 and 1 by dividing by the total number of counts for each sample. Default FALSE.
- **show_column_names** Boolean check. If True, column names are shown. Otherwise, they aren’t. Default FALSE.
- **show_row_names** Boolean check. If True, row names are shown. Otherwise, they aren’t. Default FALSE.
- **scale** Boolean check. If True, values are scaled by z-score. Otherwise, they aren’t. Default TRUE.
- **subset_tumor** Users can specify certain tumor types on which they want to subset the exposure matrix for plotting the heatmap.
- **subset_signatures** Users can specify certain signatures on which they want to subset the exposure matrix plotting the heatmap.
- **annotation** Users have the option of plotting the exposure matrix based on their given annotation like Tumor_Subtypes or age. Error given if the user given annotation doesn’t exist in the res_annot annotation object.
- **...** Ellipsis used for passing any arguments directly to the ComplexHeatmap’s heatmap function.

Value

Generates a heatmap for using the exposure matrix.

Examples

```r
data(res_annot)
plot_heatmap(res_annot = res_annot, proportional = TRUE, scale = TRUE, annotation = "Tumor_Subtypes")
```
plot_sample_counts  

Plot distribution of sample counts

Description

Displays the proportion of counts for each mutation type across one or more samples.

Usage

plot_sample_counts(
  musica,
  sample_names,
  table_name = NULL,
  text_size = 10,
  show_x_labels = TRUE,
  show_y_labels = TRUE,
  same_scale = TRUE,
  annotation = NULL
)

Arguments

musica  
A musica object.

sample_names  
Names of the samples to plot.

table_name  
Name of table used for plotting counts. If NULL, then the first table in the musica object will be used. Default NULL.

text_size  
Size of axis text. Default 10.

show_x_labels  
If TRUE, the labels for the mutation types on the x-axis will be shown. Default TRUE.

show_y_labels  
If TRUE, the y-axis ticks and labels will be shown. Default TRUE.

same_scale  
If TRUE, the scale of the y-axis for each sample will be the same. If FALSE, then the scale of the y-axis will be adjusted for each sample. Default TRUE.

annotation  
Vector of annotations to be displayed in the top right corner of each sample. Vector length must be equivalent to the number of samples. Default NULL.

Value

Generates a ggplot object

Examples

data(musica_sbs96)
plot_sample_counts(musica_sbs96, sample_names = sample_names(musica_sbs96)[1])
plot_sample_reconstruction_error

Plot reconstruction error for a sample

Description
Displays the observed distribution of counts for each mutation type, the distribution of reconstructed counts for each mutation type using the inferred mutational signatures, and the difference between the two distributions.

Usage
plot_sample_reconstruction_error(result, sample, plotly = FALSE)

Arguments
- result: A musica_result object generated by a mutational discovery or prediction tool.
- sample: Name of the sample within the musica_result object.
- plotly: If TRUE, the the plot will be made interactive using plotly. Default FALSE.

Value
Generates a ggplot or plotly object

Examples

data(res)
plot_sample_reconstruction_error(res, "TCGA-ER-A197-06A-32D-A197-08")

plot_signatures
Plots the mutational signatures

Description
After mutational signature discovery has been performed, this function can be used to display the distribution of each mutational signature. The color_variable and color_mapping parameters can be used to change the default color scheme of the bars.
Usage

plot_signatures(
    result,
    plotly = FALSE,
    color_variable = NULL,
    color_mapping = NULL,
    text_size = 10,
    show_x_labels = TRUE,
    show_y_labels = TRUE,
    same_scale = TRUE,
    y_max = NULL,
    annotation = NULL,
    percent = TRUE
)

Arguments

result A \texttt{musica_result} object generated by a mutational discovery or prediction tool.

plotly If TRUE, the plot will be made interactive using \texttt{plotly}. Default FALSE.

color_variable Name of the column in the variant annotation data.frame to use for coloring the mutation type bars. The variant annotation data.frame can be found within the count table of the \texttt{musica} object. If NULL, then the default column specified in the count table will be used. Default NULL.

color_mapping A character vector used to map items in the \texttt{color_variable} to a color. The items in \texttt{color_mapping} correspond to the colors. The names of the items in \texttt{color_mapping} should correspond to the unique items in \texttt{color_variable}. If NULL, then the default \texttt{color_mapping} specified in the count table will be used. Default NULL.

text_size Size of axis text. Default 10.

show_x_labels If TRUE, the labels for the mutation types on the x-axis will be shown. Default TRUE.

show_y_labels If TRUE, the y-axis ticks and labels will be shown. Default TRUE.

same_scale If TRUE, the scale of the probability for each signature will be the same. If FALSE, then the scale of the y-axis will be adjusted for each signature. Default TRUE.

y_max Vector of maximum y-axis limits for each signature. One value may also be provided to specify a constant y-axis limit for all signatures. Vector length must be 1 or equivalent to the number of signatures. Default NULL.

annotation Vector of annotations to be displayed in the top right corner of each signature. Vector length must be equivalent to the number of signatures. Default NULL.

percent If TRUE, the y-axis will be represented in percent format instead of mutation counts. Default TRUE.

Value

Generates a ggplot or plotly object
Examples

```r
data(res)
plot_signatures(res)
```

Description

Plots samples on a UMAP scatterplot. Samples can be colored by the levels of mutational signatures or by an annotation variable.

Usage

```r
plot_umap(
  result,
  color_by = c("signatures", "annotation", "cluster", "none"),
  proportional = TRUE,
  annotation = NULL,
  point_size = 0.7,
  same_scale = TRUE,
  add_annotation_labels = FALSE,
  annotation_label_size = 3,
  annotation_text_box = TRUE,
  plotly = FALSE,
  clust = NULL,
  legend = TRUE,
  strip_axes = FALSE
)
```

Arguments

- **result**: A `musica_result` object generated by a mutational discovery or prediction tool.
- **color_by**: One of "signatures", "annotation", or "none". If "signatures", then one UMAP scatterplot will be generated for each signature and points will be colored by the level of that signature in each sample. If annotation, a single UMAP will be generated colored by the annotation selected using the parameter annotation. If "none", a single UMAP scatterplot will be generated with no coloring. Default "signature".
- **proportional**: If TRUE, then the exposures will be normalized to between 0 and 1 by dividing by the total number of counts for each sample. Default TRUE.
- **annotation**: Sample annotation used to color the points. One used when color_by = "annotation". Default NULL.
- **point_size**: Scatter plot point size. Default 0.7.
predict_exposure

same_scale If TRUE, then all points will share the same color scale in each signature subplot. If FALSE, then each signature subplot will be colored by a different scale with different maximum values. Only used when color_by = "signature". Setting to FALSE is most useful when the maximum value of various signatures are vastly different from one another. Default TRUE.

addAnnotation_labels If TRUE, labels for each group in the annotation variable will be displayed. Only used if codecolor_by = "annotation". This not recommended if the annotation is a continuous variable. The label is plotting using the centroid of each group within the annotation variable. Default FALSE.

annotation_label_size Size of annotation labels. Only used if codecolor_by = "annotation" and add_annotation_labels = TRUE. Default 3.

annotation_text_box Place a white box around the annotation labels to improve readability. Only used if codecolor_by = "annotation" and add_annotation_labels = TRUE. Default TRUE.

plotly If TRUE, the plot will be made interactive using plotly. Not used if color_by = "signature" and same_scale = FALSE. Default FALSE.

clust Add cluster labels as annotation

legend Plot legend

strip_axes Remove axes labels for cleaner looking plots

Value Generates a ggplot or plotly object

See Also See create_umap to generate a UMAP in a musica result.

Examples

data(res_annot)
create_umap(res_annot, "Tumor_Subtypes")
plot_umap(res_annot, "none")

predict_exposure Prediction of exposures in new samples using pre-existing signatures

Description Exposures for samples will be predicted using an existing set of signatures stored in a musica_result object. Algorithms available for prediction include a modify version of "lda", "decompTumor2Sig", and "deconstructSigs".
predict_exposure

Usage

predict_exposure(
  musica,  
g,  
table_name,  
signature_res,  
algorithm = c("lda", "decompTumor2Sig", "deconstructSigs"),  
signatures_to_use = seq_len(ncol(signatures(signature_res))),  
verbose = FALSE
)

Arguments

musica A musica object.

g A BSgenome object indicating which genome reference the variants and their
coordinates were derived from. Only used if algorithm = "deconstructSigs"
table_name Name of table used for posterior prediction. Must match the table type used to
generate the prediction signatures
signature_res Signatures used to predict exposures for the samples musica object. Existing
signatures need to stored in a musica_result object.
algorithm Algorithm to use for prediction of exposures. One of "lda", "decompTumor2Sig", or "deconstructSigs".
signatures_to_use Which signatures in the signature_res result object to use. Default is to use
all signatures.
verbose If TRUE, progress will be printing. Only used if algorithm = "lda". Default FALSE.

Value

Returns a A musica_result object containing signatures given by the signature_res parameter
and exposures predicted from these signatures.

Examples

data(musica)
data(cosmic_v2_sigs)
g <- select_genome("19")
build_standard_table(musica, g, "SBS96", overwrite = TRUE)
result <- predict_exposure(musica = musica, table_name = "SBS96",
signature_res = cosmic_v2_sigs, algorithm = "lda")

# Predict using LDA-like algorithm with seed set to 1
set.seed(1)
predict_exposure(musica = musica, table_name = "SBS96",
signature_res = cosmic_v2_sigs, algorithm = "lda")
**rc**

*Reverse complement of a string using biostrings*

Description

Reverse complement of a string using biostrings

Usage

rc(dna)

Arguments

dna

Input DNA string

Value

Returns the reverse compliment of the input DNA string

Examples

rc("ATGC")

**rep_range**

*Replication Timing Data as GRanges Object*

Description

Supplementary data converted from bigWig to bedgraph to GRanges, with low RFD indicating the leading strand and high RFD indicating lagging strand and removing uninformative zero RFD intervals. Timing data is 10kb bins from a colon cancer sample.

Usage

data(rep_range)

Format

An object of class "GRanges"; see [annotate_replication_strand()].

Source


References

Description

A musica result created for testing that includes SBS variants with discovered exposures and signatures

Usage

data(res)

Format

An object of class `musica_result` See [discover_signatures()].

Description

A musica result created for testing that includes SBS variants with annotations and discovered exposures and signatures

Usage

data(res_annot)

Format

An object of class `musica_result` See [discover_signatures()].

sample_names

Retrieve sample names from a musica or musica_result object

Description

Sample names were included in the sample column in the variant object passed to `create_musica`. This returns a unique list of samples names in the order they are inside the musica object.
Usage

```r
sample_names(object)
```

## S4 method for signature 'musica'

```r
sample_names(object)
```

## S4 method for signature 'musica_result'

```r
sample_names(object)
```

Arguments

- `object`: A `musica` object generated by the `create_musica` function or a `musica_result` object generated by a mutational discovery or prediction tool.

Value

A character vector of sample names

Examples

```r
data(res)
sample_names(res)
```

### samp_annot

*Get or set sample annotations from a musica or musica_result object*

#### Description

Sample annotations can be used to store information about each sample such as tumor type or treatment status. These are used in downstream plotting functions such as `plot_exposures` or `plot_umap` to group or color samples by a particular annotation.

Usage

```r
samp_annot(object)
```

## S4 method for signature 'musica'

```r/nsamp_annot(object)
```

## S4 method for signature 'musica_result'

```r/nsamp_annot(object)
```

```r
samp_annot(object, name) <- value
```

## S4 replacement method for signature 'musica,character,vector'

```r
samp_annot(object, name) <- value
```

## S4 replacement method for signature 'musica_result,character,vector'

```r
samp_annot(object, name) <- value
```
Arguments

object A `musica` object generated by the `create_musica` function or a `musica_result` object generated by a mutational discovery or prediction tool.

name The name of the new annotation to add.

value A vector containing the new sample annotations. Needs to be the same length as the number of samples in the object.

Value

A new object with the sample annotations added to the table in the `sample_annotations` slot.

See Also

See `sample_names` to get a vector of sample names in the `musica` or `musica_result` object.

Examples

data(res_annot)
samp_annot(res_annot)

# Add new annotation
samp_annot(res_annot, "New_Annotation") <- rep(c("A", "B"), c(3, 4))
samp_annot(res_annot)
data(musica)
samp_annot(musica, "example") <- rep("ex", 7)

---

select_genome

Helper function to load common human or mouse genomes

Description

Helper function to load common human or mouse genomes

Usage

select_genome(x)

Arguments

x Select the hg19 or hg38 human genome or the mm9 or mm10 mouse genome in UCSC format

Value

Returns BSgenome of given version

Examples

g <- select_genome(x = "hg38")
**signatures**

*Retrieve signatures from a musica_result object*

**Description**

The signature matrix contains the probability of mutation motif in each sample. Rows correspond to each motif and columns correspond to each signature.

**Usage**

```r
signatures(result)
```

```r
## S4 method for signature 'musica_result'
signatures(result)
```

```r
signatures(result) <- value
```

```r
## S4 replacement method for signature 'musica_result,matrix'
signatures(result) <- value
```

**Arguments**

- **result**
  
  A *musica_result* object generated by a mutational discovery or prediction tool.

- **value**
  
  A matrix of motifs counts by samples

**Value**

A matrix of mutational signatures

**Examples**

```r
data(res)
signatures(res)
data(res)
signatures(res) <- matrix()
```

---

**subset_musica_by_annotation**

*Creates a new musica object subsetted to only one value of a sample annotation*

**Description**

Creates a new musica object subsetted to only one value of a sample annotation
**subset_musica_by_counts**

Usage

```r
subset_musica_by_counts(musica, annot_col, annot_names)
```

Arguments

- `musica`: A `musica` object.
- `annot_col`: Annotation class to use for subsetting.
- `annot_names`: Annotational value to subset to.

Value

Returns a new `musica` object with sample annotations, count tables, and variants subsetted to only contains samples of the specified annotation type.

Examples

```r
data(musica_sbs96)
annot <- read.table(system.file("extdata", "sample_annotations.txt", package = "musicatk"), sep = "\t", header=TRUE)
samp_annot(musica_sbs96, "Tumor_Subtypes") <- annot$Tumor_Subtypes
musica_sbs96 <- subset_musica_by_annotation(musica_sbs96, "Tumor_Subtypes", "Lung")
```

**subset_musica_by_counts**

*Creates a new musica subsetted to only samples with enough variants*

Description

Creates a new musica subsetted to only samples with enough variants.

Usage

```r
subset_musica_by_counts(musica, table_name, num_counts)
```

Arguments

- `musica`: A `musica` object.
- `table_name`: Name of table used for subsetting.
- `num_counts`: Minimum sum count value to drop samples.

Value

Returns a new `musica` object with sample annotations, count tables, and variants subsetted to only contains samples with the specified minimum number of counts (column sums) in the specified table.
subset_variant_by_type

Description
Subsets a variant table based on Variant Type

Usage
subset_variant_by_type(tab, type)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tab</td>
<td>Input variant table</td>
</tr>
<tr>
<td>type</td>
<td>Variant type to return e.g. &quot;SBS&quot;, &quot;INS&quot;, &quot;DEL&quot;, &quot;DBS&quot;</td>
</tr>
</tbody>
</table>

Examples
data(musica)  
subset_variant_by_type(musica, "SBS")
**Value**

Returns the input variant table subsetted to only contain variants of the specified variant type.

**Examples**

```r
data(musica)
analyze_variant_type(musica)
subset_variant_by_type(variants(musica), "SBS")
```

---

**Description**

The `count_tables` contains standard and/or custom count tables created from variants.

**Usage**

```r
tables(object)
```

## S4 method for signature 'musica'
```
tables(object)
```

## S4 method for signature 'musica_result'
```
tables(object)
```

tables(musica) <- value

## S4 replacement method for signature 'musica,list'
```
tables(musica) <- value
```

**Arguments**

- **object**
  - A `musica` object generated by the `create_musica` function or a `musica_result` object generated by a mutational discovery or prediction tool.

- **value**
  - A list of `count_table` objects representing counts of motifs in samples.

**Value**

A list of `count_tables`
Examples

```r
data(res)
tables(res)
data(musica)
tables(musica)
```

table_96

*Generates a 96 motif table based on input counts for plotting*

Description

Generates a 96 motif table based on input counts for plotting

Usage

```r
table_96(sample_df)
```

Arguments

- `sample_df` Input counts table

Value

Returns a 96 motif summary table

table_selected

*Retrieve table name used for plotting from a musica_result object*

Description

The table name

Usage

```r
table_selected(result)
```

Arguments

- `result` A `musica_result` object generated by a mutational discovery or prediction tool.

Value

Table name used for plotting
umap

Examples

   data(res)
   table_selected(res)

---

umap  

Retrieve umap list from a musica_result object

Description

The signature matrix contains the probability of mutation motif in each sample. Rows correspond to each motif and columns correspond to each signature.

Usage

umap(result)

## S4 method for signature 'musica_result'

umap(result)

umap(result) <- value

## S4 replacement method for signature 'musica_result,matrix'

umap(result) <- value

Arguments

  result      A musica_result object generated by a mutational discovery or prediction tool.
  value       A list of umap dataframes

Value

A list of umap dataframes

Examples

   data(res)
   umap(res)
   data(res)
   umap(res) <- matrix()
variants  

Retrieve variants from a musica or musica_result object

Description

The variants data.table contains the variants and variant-level annotations

Usage

variants(object)

## S4 method for signature 'musica'
variants(object)

## S4 method for signature 'musica_result'
variants(object)

variants(musica) <- value

## S4 replacement method for signature 'musica,data.table'
variants(musica) <- value

Arguments

object  A musica object generated by the create_musica function or a musica_result object generated by a mutational discovery or prediction tool.
musica  A musica object generated by the create_musica function
value  A data.table of mutational variants and variant-level annotations

Value

A data.table of variants

Examples

data(res)
variants(res)
data(musica)
variants(musica)
Description
Pipe operator

Usage
lhs %>% rhs

Value
NA

Examples
  c(1,2) %>% barplot()
Index

* datasets
  cosmic_v2_sigs, 20
  cosmic_v3_dbs_sigs, 21
  cosmic_v3_indel_sigs, 21
  cosmic_v3_sbs_sigs, 22
  cosmic_v3_sbs_sigs_exome, 22
  dbs_musica, 28
  indel_musica, 41
  musica, 43
  musica_annot, 44
  musica_sbs96, 45
  musica_sbs96_tiny, 46
  rep_range, 58
  res, 59
  res_annot, 59
* internal
  .jsd, 4
  %>%%, 69
  add_variant_type, 6
  auto_subset_sigs, 10
  create_dbs78_table, 23
  create_sbs192_table, 25
  create_sbs96_table, 26
  table_96, 66
  .jsd, 4
  %>%%, 69
  add_flank_to_variants, 5
  add_variant_type, 6
  annotate_replication_strand, 6
  annotate_transcript_strand, 7
  annotate_variant_length, 8
  annotate_variant_type, 8
  auto_predict_grid, 9
  auto_subset_sigs, 10

BSgenome, 5, 12, 24, 26, 57
build_custom_table, 11
build_standard_table, 12, 28, 29
built_tables, 13

built_tables, musica-method
(built_tables), 13
built_tables, musica_result-method
(built_tables), 13

cluster_exposure, 14
CollapsedVCF, 33
combine_count_tables, 15
combine_predict_grid, 16
compare_cosmic_v2, 17
compare_cosmic_v3, 18
compare_results, 19

.cosmic_v2_sigs, 20
.cosmic_v2_subtype_map, 20
.cosmic_v3_dbs_sigs, 21
.cosmic_v3_indel_sigs, 21
.cosmic_v3_sbs_sigs, 22
.cosmic_v3_sbs_sigs_exome, 22
count_table, 65
count_table-class, 23
create_dbs78_table, 23
create_dbs92_table, 25
create_sbs96_table, 26
create_umap, 27, 47, 56
data.table, 68
dbs_musica, 28
discover_signatures, 28
drop_annotation, 29

ExpandedVCF, 33
exposure_differential_analysis, 31
exposures, 30
exposures, musica_result-method
(exposures), 30

exposures<- (exposures), 30
exposures<-, musica_result, matrix-method
(exposures), 30
extract_count_tables, 32
variants<-(variants), 68
variants<-, musica, data.table-method
   (variants), 68