

# Package ‘musicatk’

December 21, 2024

**Type** Package

**Title** Mutational Signature Comprehensive Analysis Toolkit

**Version** 2.1.0

**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 (>= 4.4.0), 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.Mmusculus.UCSC.mm9, BSgenome.Mmusculus.UCSC.mm10, decompTumor2Sig, topicmodels,

ggrepel, plotly, utils, factoextra, cluster, ComplexHeatmap,  
 philentropy, maftools, shiny, stringi, tidyverse, ggpubr,  
 Matrix ( $\geq 1.6.1$ ), scales, conclust

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

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/musicatk>

**git\_branch** devel

**git\_last\_commit** 45539aa

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-20

**Author** Aaron Chevalier [aut] (ORCID: 0000-0002-3968-9250),  
 Natasha Gurevich [aut] (ORCID: 0000-0002-0747-6840),  
 Tao Guo [aut] (ORCID: 0009-0005-8960-9203),  
 Joshua D. Campbell [aut, cre] (ORCID:  
<https://orcid.org/0000-0003-0780-8662>)

**Maintainer** Joshua D. Campbell <camp@bu.edu>

## Contents

.jsd . . . . .	4
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
build_custom_table . . . . .	11
build_standard_table . . . . .	12
built_tables . . . . .	14
cluster_exposure . . . . .	14
combine_count_tables . . . . .	16
combine_predict_grid . . . . .	17
compare_cosmic_v2 . . . . .	18
compare_cosmic_v3 . . . . .	19
compare_k_vals . . . . .	20
compare_results . . . . .	21
cosmic_v2_sigs . . . . .	22

cosmic_v2_subtype_map . . . . .	23
cosmic_v3_dbs_sigs . . . . .	23
cosmic_v3_indel_sigs . . . . .	24
cosmic_v3_sbs_sigs . . . . .	24
cosmic_v3_sbs_sigs_exome . . . . .	25
count_table-class . . . . .	25
create_dbs78_table . . . . .	26
create_musica_from_counts . . . . .	27
create_musica_from_variants . . . . .	27
create_result_model . . . . .	29
create_sbs192_table . . . . .	30
create_sbs96_table . . . . .	30
create_umap . . . . .	31
credible_intervals . . . . .	32
dbs_musica . . . . .	33
discover_signatures . . . . .	33
drop_annotation . . . . .	35
exposures . . . . .	35
exposure_differential_analysis . . . . .	37
extract_count_tables . . . . .	38
extract_variants . . . . .	38
extract_variants_from_maf . . . . .	41
extract_variants_from_maf_file . . . . .	41
extract_variants_from_matrix . . . . .	42
extract_variants_from_vcf . . . . .	43
extract_variants_from_vcf_file . . . . .	44
generate_result_grid . . . . .	46
get_count_table . . . . .	47
get_modality . . . . .	48
get_model . . . . .	48
get_result_list_entry . . . . .	49
hyperparameter . . . . .	50
indel_musica . . . . .	51
k_select . . . . .	51
metrics . . . . .	52
modality . . . . .	53
model_id . . . . .	55
musica . . . . .	56
musica-class . . . . .	56
musicatk . . . . .	57
musica_annot . . . . .	57
musica_sbs96 . . . . .	58
musica_sbs96_tiny . . . . .	58
name_signatures . . . . .	58
num_signatures . . . . .	59
other_parameters . . . . .	60
parameter . . . . .	61
plot_cluster . . . . .	62

plot_differential_analysis . . . . .	64
plot_exposures . . . . .	65
plot_heatmap . . . . .	67
plot_k_comparison . . . . .	68
plot_sample_counts . . . . .	69
plot_sample_reconstruction_error . . . . .	70
plot_signatures . . . . .	71
plot_umap . . . . .	72
predict_exposure . . . . .	74
rc . . . . .	76
rep_range . . . . .	76
res . . . . .	77
result_collection-class . . . . .	77
result_list . . . . .	77
result_model-class . . . . .	78
res_annot . . . . .	79
sample_names . . . . .	79
samp_annot . . . . .	80
select_genome . . . . .	81
signatures . . . . .	81
subset_musica_by_annotation . . . . .	82
subset_musica_by_counts . . . . .	83
subset_variants_by_samples . . . . .	84
subset_variant_by_type . . . . .	84
tables . . . . .	85
table_96 . . . . .	86
table_selected . . . . .	86
umap . . . . .	87
variants . . . . .	88
%>% . . . . .	89
<b>Index</b>	<b>90</b>

---

.jsd	<i>Calculates 1 - Jensen-Shannon Divergences between all pairs of columns between two matrices</i>
------	--

---

## 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**

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

**Arguments**

musica	Input samples
g	A <a href="#">BSgenome</a> 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**

```
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

**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	<i>Automatic filtering of signatures for exposure prediction gridded across specific annotation</i>
-------------------	---

---

**Description**

Automatic filtering of signatures for exposure prediction gridded across specific annotation

**Usage**

```
auto_predict_grid(
  musica,
  modality,
  signature_res,
  algorithm,
  model_id = NULL,
  result_name = "result",
  sample_annotation = NULL,
  min_exists = 0.05,
  proportion_samples = 0.25,
  rare_exposure = 0.4,
  verbose = TRUE,
  combine_res = TRUE,
  make_copy = FALSE,
  table_name = NULL
)
```

**Arguments**

musica	Input samples to predict signature weights
modality	Modality 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", and decompTumor2Sig
model_id	Name of model
result_name	Name for result_list entry to save the results to. Default "result".
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
make_copy	If FALSE, the inputted <code>musica</code> object is updated and nothing is returned. If TRUE, a new <code>musica</code> object is created and returned. Default FALSE.
table_name	Use modality instead

**Value**

Returns nothing or a new `musica` object, depending on the `make_copy` parameter.

**Examples**

```
data(musica_annot)
data(cosmic_v2_sigs)
auto_predict_grid(
  musica = musica_annot, modality = "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	<i>Automatic filtering of inactive signatures</i>
------------------	---

---

**Description**

Automatic filtering of inactive signatures

**Usage**

```
auto_subset_sigs(
  musica,
  modality,
  signature_res,
  algorithm,
  min_exists = 0.05,
  proportion_samples = 0.25,
  rare_exposure = 0.4,
  result_name = "result",
  model_id = NULL
)
```

**Arguments**

musica	A <code>musica</code> object.
modality	Modality 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" and decompTumor2Sig
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
result_name	Name for result_list entry to save the results to. Default "result".
model_id	Identifier for the result. If NULL, will be automatically set to the algorithm and number of signatures. Default NULL.

**Value**

Returns new `musica` object with the results.

---

build_custom_table	<i>Builds a custom table from specified user variants</i>
--------------------	---

---

**Description**

Builds a custom table from specified user variants

**Usage**

```
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 <code>musica</code> 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 <code>data.frame</code> of annotations to use for plotting
features	A <code>data.frame</code> 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 <code>annotation_df</code> used for the coloring in plots
color_mapping	The mapping from the values in the selected <code>color_variable</code> column to color values for plotting
return_instead	Instead of adding to <code>musica</code> 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)
annotate_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,
  modality,
  strand_type = NULL,
  overwrite = FALSE,
  verbose = FALSE,
  table_name = NULL
)

```

**Arguments**

musica	A <a href="#">musica</a> object.
g	A <a href="#">BSgenome</a> object indicating which genome reference the variants and their coordinates were derived from.
modality	Modality of 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 modality = 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
table_name	Use modality instead

**Value**

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

**Examples**

```

g <- select_genome("19")

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

data(musica)
annotate_transcript_strand(musica, "19")
build_standard_table(musica, g, "SBS192", "Transcript_Strand")

data(musica)
data(rep_range)
annotate_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, modality = "INDEL")
```

---

built_tables	<i>Retrieve the names of count_tables from a musica object</i>
--------------	--

---

### Description

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

### Usage

```
built_tables(object)

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

### Arguments

object            A *musica* object generated by the [create\\_musica\\_from\\_variants](#) or [create\\_musica\\_from\\_counts](#) function.

### Value

The names of created count\_tables

### Examples

```
data(res)
built_tables(res)
```

---

cluster_exposure	<i>Perform clustering analysis from a musica result object</i>
------------------	--

---

### Description

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

**Usage**

```
cluster_exposure(
  musica,
  model_name,
  modality = "SBS96",
  result_name = "result",
  nclust,
  proportional = TRUE,
  method = "kmeans",
  dis.method = "euclidean",
  hc.method = "ward.D",
  clara.samples = 5,
  iter.max = 10,
  tol = 1e-15
)
```

**Arguments**

<code>musica</code>	A <a href="#">musica</a> object containing a mutational discovery or prediction.
<code>model_name</code>	The name of the desired model.
<code>modality</code>	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
<code>result_name</code>	Name of the result list entry containing desired model. Default "result".
<code>nclust</code>	Pre-defined number of clusters.
<code>proportional</code>	Logical, indicating if proportional exposure (default) will be used for clustering.
<code>method</code>	Clustering algorithms. Options are "kmeans" (K-means), "hkmeans" (hybrid of hierarchical K-means), "hclust" (hierarchical clustering), "pam" (PAM), and "clara" (Clara).
<code>dis.method</code>	Methods to calculate dissimilarity matrix. Options are "euclidean" (default), "manhattan", "jaccard", "cosine", and "canberra".
<code>hc.method</code>	Methods to perform hierarchical clustering. Options are "ward.D" (default), "ward.D2", "single", "complete", "average", "mcquitty", "median", and "centroid".
<code>clara.samples</code>	Number of samples to be drawn from dataset. Only used when "clara" is selected. Default is 5.
<code>iter.max</code>	Maximum number of iterations for k-means clustering.
<code>tol</code>	Tolerance level for kmeans clustering level iterations

**Value**

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

**See Also**

[kmeans](#)

**Examples**

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

---

combine\_count\_tables *Combines tables into a single table that can be used for discovery/prediction*

---

**Description**

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

**Usage**

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

**Arguments**

musica	A <a href="#">musica</a> 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

**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` *Combine signatures and exposures of different models. Exposure values are zero for samples in an annotation where that signature was not predicted*

---

**Description**

Combine signatures and exposures of different models. Exposure values are zero for samples in an annotation where that signature was not predicted

**Usage**

```
combine_predict_grid(
  musica,
  modality,
  signature_res,
  model_ids = NULL,
  result_name = "result",
  model_rename = NULL,
  make_copy = FALSE,
  table_name = NULL
)
```

**Arguments**

<code>musica</code>	A <code>musica</code> object.
<code>modality</code>	Modality used for prediction.
<code>signature_res</code>	Signatures to automatically subset from for prediction
<code>model_ids</code>	Vector of ids for the models to combine. If null, all models in the modality and <code>result_list</code> entry will be combined. Default NULL.
<code>result_name</code>	Name of the result list entry containing the signatures to plot. Default "result".
<code>model_rename</code>	New model identifier. If null, will be combination of the ids for the models being combined. Default NULL.
<code>make_copy</code>	If FALSE, the inputted <code>musica</code> object is updated and nothing is returned. If TRUE, a new <code>musica</code> object is created and returned. Default FALSE.
<code>table_name</code>	Use modality instead

**Value**

Returns nothing or a new `musica` object, depending on the `make_copy` parameter.

**Examples**

```
data(musica_annot)
data(cosmic_v2_sigs)
grid <- auto_predict_grid(musica_annot, "SBS96", cosmic_v2_sigs, "lda",
  "Tumor_Subtypes",
  combine_res = FALSE, make_copy = TRUE
)
combined <- combine_predict_grid(grid, "SBS96", cosmic_v2_sigs,
  make_copy = TRUE)
```

---

compare_cosmic_v2	<i>Compare a result object to COSMIC V2 SBS Signatures (combination whole-exome and whole-genome)</i>
-------------------	---

---

**Description**

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

**Usage**

```
compare_cosmic_v2(
  musica,
  model_id,
  modality = "SBS96",
  result_name = "result",
  metric = "cosine",
  threshold = 0.9,
  result_rename = deparse(substitute(result)),
  decimals = 2,
  same_scale = FALSE
)
```

**Arguments**

musica	A <code>musica</code> object.
model_id	The name of the model containing the signatures to compare.
modality	Compare to SBS, DBS, or Indel. Default "SBS96"
result_name	Name of the result list entry. Default "result".
metric	One of "cosine" for cosine similarity or "jsd" for 1 minus the Jensen-Shannon Divergence. Default "cosine".
threshold	threshold for similarity

result_rename	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**

```
data(res)
compare_cosmic_v2(res, model_id = "res", threshold = 0.7)
```

---

compare_cosmic_v3	<i>Compare a result object to COSMIC V3 Signatures; Select exome or genome for SBS and only genome for DBS or Indel classes</i>
-------------------	---

---

**Description**

Compare a result object to COSMIC V3 Signatures; Select exome or genome for SBS and only genome for DBS or Indel classes

**Usage**

```
compare_cosmic_v3(
  musica,
  model_id,
  sample_type,
  modality = "SBS96",
  result_name = "result",
  metric = "cosine",
  threshold = 0.9,
  result_rename = deparse(substitute(model_id)),
  decimals = 2,
  same_scale = FALSE
)
```

**Arguments**

musica	A <a href="#">musica</a> object.
model_id	The name of the model containing the signatures to compare.
sample_type	exome (SBS only) or genome
modality	Compare to SBS, DBS, or Indel. Default "SBS96"
result_name	Name of the result list entry. Default "result".

metric	One of "cosine" for cosine similarity or "jsd" for 1 minus the Jensen-Shannon Divergence. Default "cosine".
threshold	threshold for similarity
result_rename	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

```
data(res)
compare_cosmic_v3(res,
  model_id = "res", modality = "SBS96",
  sample_type = "genome", threshold = 0.8
)
```

---

compare_k_vals	<i>Compare k values</i>
----------------	-------------------------

---

### Description

Compare the stability and error of various k values to help determine the correct number of signatures (k).

### Usage

```
compare_k_vals(
  musica,
  modality,
  reps = 100,
  min_k = 1,
  max_k = 10,
  error_type = "prop",
  algorithm = "nmf"
)
```

### Arguments

musica	A <a href="#">musica</a> object.
modality	The modality to use, either "SBS96", "DBS78", or "IND83".
reps	Number of times prediction is performed. For each replicate, the count table data is perturbed. Multiple replicates allows for stability analysis by calculating silhouette width on the multiple results. Default 100.

min_k	Lower range of number of signatures for discovery. Default 1.
max_k	Upper range of number of signatures for discovery. Default 10.
error_type	Whether to calculate reconstruction error by proportions ("prop") or raw counts ("raw"). Default "prop".
algorithm	Algorithm for signature discovery. Default "nmf".

**Value**

a data.frame with stats for each k value tested

**Examples**

```
data(musica)
compare_k_vals(musica, "SBS96", reps = 3, min_k = 1, max_k = 5)
```

---

compare_results	<i>Compare two result files to find similar signatures</i>
-----------------	--

---

**Description**

Compare two result files to find similar signatures

**Usage**

```
compare_results(
  musica,
  model_id,
  other_model_id,
  modality = "SBS96",
  result_name = "result",
  other_musica = NULL,
  other_result_name = "result",
  threshold = 0.9,
  metric = "cosine",
  result_rename = deparse(substitute(model_id)),
  other_result_rename = deparse(substitute(other_model_id)),
  decimals = 2,
  same_scale = FALSE
)
```

**Arguments**

musica	A <a href="#">musica</a> object.
model_id	The name of the first model to compare.
other_model_id	The name of the second model to compare.
modality	Modality of results being compared. Default "SBS96".

result_name	Name of the result list entry for the first result to compare. Default "result".
other_musica	A second <code>musica</code> object. If null, the provided musica object is used twice. Default NULL.
other_result_name	Name of the result list entry for the second result to compare. Default "result".
threshold	threshold for similarity
metric	One of "cosine" for cosine similarity or "jsd" for 1 minus the Jensen-Shannon Divergence. Default "cosine".
result_rename	title for plot of first result signatures
other_result_rename	title for plot of second 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 FALSE.

**Value**

Returns the comparisons

**Examples**

```
data(res)
compare_results(res,
  model_id = "res", other_model_id = "res",
  modality = "SBS96", 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**

```
data(cosmic_v2_sigs)
```

**Format**

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

**Source**

COSMIC v2, <[https://cancer.sanger.ac.uk/cosmic/signatures\\_v2](https://cancer.sanger.ac.uk/cosmic/signatures_v2)>

**References**

Alexandrov, L., Nik-Zainal, S., Wedge, D. et al. (2013) Signatures of mutational processes in human cancer. Nature 500, 415–421 ([Nature](https://www.ncbi.nlm.nih.gov/pubmed/23945592))

---

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**

```
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**

```
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**

```
data(cosmic_v3_dbs_sigs)
```

**Format**

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

**Source**

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

**References**

Alexandrov, L.B., Kim, J., Haradhvala, N.J. et al. (2020) The repertoire of mutational signatures in human cancer. *Nature* 578, 94–101 ([Nature](https://doi.org/10.1038/s41586-020-1943-3))

---

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**

```
data(cosmic_v3_indel_sigs)
```

**Format**

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

**Source**

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

**References**

Alexandrov, L.B., Kim, J., Haradhvala, N.J. et al. (2020) The repertoire of mutational signatures in human cancer. *Nature* 578, 94–101 ([Nature](https://doi.org/10.1038/s41586-020-1943-3))

---

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 `result_model`. See `[predict_exposure()]`.

**Source**

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



**References**

Alexandrov, L.B., Kim, J., Haradhvala, N.J. et al. (2020) The repertoire of mutational signatures in human cancer. *Nature* 578, 94–101 ([Nature](https://doi.org/10.1038/s41586-020-1943-3))

---

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 `result_model`. See `[predict_exposure()]`.

**Source**

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

**References**

Alexandrov, L.B., Kim, J., Haradhvala, N.J. et al. (2020) The repertoire of mutational signatures in human cancer. *Nature* 578, 94–101 ([Nature](https://doi.org/10.1038/s41586-020-1943-3))

---

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 they 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 independantly. 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_from_counts`*Creates a musica object from a mutation count table*

---

### Description

This function creates a `musica` object from a mutation count 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.

### Usage

```
create_musica_from_counts(x, variant_class)
```

### Arguments

`x` A `data.table`, `matrix`, or `data.frame` that contains counts of mutation types for each sample, with samples as columns.

`variant_class` Mutations are SBS, DBS, or Indel.

### Value

Returns a `musica` object

### Examples

```
data(musica)
count_table <- get_count_table(extract_count_tables(musica)$SBS96)
musica <- create_musica_from_counts(count_table, "SBS96")
```

---

`create_musica_from_variants`*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**

```

create_musica_from_variants(
  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 <a href="#">BSgenome</a> 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 be automatically fix these with the <a href="#">seqlevelsStyle</a> 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_from_variants(x = variants, genome = g)
```

---

create\_result\_model    *Load an external model into a result\_model object*

---

**Description**

This function creates a [result\\_model](#) object from signatures, exposures, and a mutation count table.

**Usage**

```
create_result_model(signatures, exposures, model_id, modality)
```

**Arguments**

signatures	A matrix or data.frame of signatures by mutational motifs
exposures	A matrix or data.frame of samples by signature weights
model_id	Name of model
modality	Modality of the model

**Value**

A [result\\_model](#) object

**Examples**

```
signatures <- signatures(res, "result", "SBS96", "res")
exposures <- exposures(res, "result", "SBS96", "res")
model <- create_result_model(signatures, exposures, "example_model", "SBS96")
```

---

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)
```

### Arguments

musica	Input samples
g	A <a href="#">BSgenome</a> 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

---

create\_sbs96\_table    *Uses a genome object to find context and generate standard SBS96 tables*

---

### Description

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

### Usage

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

### Arguments

musica	A <a href="#">musica</a> object.
g	A <a href="#">BSgenome</a> 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 model result*

---

## Description

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

## Usage

```
create_umap(  
  musica,  
  model_name,  
  modality = "SBS96",  
  result_name = "result",  
  n_neighbors = 30,  
  min_dist = 0.75,  
  spread = 1  
)
```

## Arguments

<code>musica</code>	A <code>musica</code> object containing a mutational signature discovery or prediction.
<code>model_name</code>	The name of the desired model.
<code>modality</code>	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
<code>result_name</code>	Name of the result list entry containing the model. Default "result".
<code>n_neighbors</code>	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 <code>n_neighbors</code> is larger than the number of samples, then <code>n_neighbors</code> will automatically be set to the number of samples in the <code>musica</code> . Default 30.
<code>min_dist</code>	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.
<code>spread</code>	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` object with a new UMAP stored in the UMAP slot of the `result_model` object for the model.

**See Also**

See [plot\\_umap](#) to display the UMAP and [umap](#) for more information on the individual parameters for generating UMAPs.

**Examples**

```
data(res_annot)
create_umap(res_annot, model_name = "res_annot")
```

---

credible_intervals	<i>Retrieve credible_intervals from a result_model, result_collection, or musica object</i>
--------------------	---

---

**Description**

Credible intervals for the model

**Usage**

```
credible_intervals(x, ...)

## S4 method for signature 'musica'
credible_intervals(x, result, modality, model_id)

## S4 method for signature 'result_collection'
credible_intervals(x, modality, model_id)

## S4 method for signature 'result_model'
credible_intervals(x)

credible_intervals(x, ...) <- value

## S4 replacement method for signature 'musica,matrix'
credible_intervals(x, result, modality, model_id) <- value

## S4 replacement method for signature 'result_collection,matrix'
credible_intervals(x, modality, model_id) <- value

## S4 replacement method for signature 'result_model,matrix'
credible_intervals(x) <- value
```

**Arguments**

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs



result	Name of result from result_list to assign the credible_intervals. Used when result is a <a href="#">musica</a> object.
modality	Modality to assign the credible_intervals. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the credible_intervals. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
value	List of credible intervals

**Value**

The credible intervals for the model

**Examples**

```
data(res)
credible_intervals(res, "result", "SBS96", "res")
```

---

dbs_musica	<i>dbs_musica</i>
------------	-------------------

---

**Description**

A musica created for testing that includes DBS variants

**Usage**

```
data(dbs_musica)
```

**Format**

An object of class musica See `[create_musica_from_variants()]` and `[create_musica_from_counts()]`.

---

discover_signatures	<i>Discover mutational signatures</i>
---------------------	---------------------------------------

---

**Description**

Mutational signatures and exposures will be discovered using methods such as Latent Dirichlet Allocation (lda) or Non-Negative Matrix Factorization (nmf). 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, samples first need to be stored in a [musica](#) object using the [create\\_musica\\_from\\_variants](#) or [create\\_musica\\_from\\_counts](#) function and mutation count tables need to be created using functions such as [build\\_standard\\_table](#) if [create\\_musica\\_from\\_counts](#) was not used.

**Usage**

```
discover_signatures(
  musica,
  modality,
  num_signatures,
  algorithm = "lda",
  result_name = "result",
  model_id = NULL,
  seed = 1,
  nstart = 10,
  par_cores = 1,
  make_copy = FALSE,
  table_name = NULL
)
```

**Arguments**

musica	A <a href="#">musica</a> object.
modality	Modality to use for signature discovery. Needs to be the same name supplied to the table building functions such as <a href="#">build_standard_table</a> .
num_signatures	Number of signatures to discover.
algorithm	Method to use for mutational signature discovery. One of "lda" or "nmf". Default "lda".
result_name	Name for result_list entry to save the results to. Default "result".
model_id	Identifier for the result. If NULL, will be automatically set to the algorithm and number of signatures. Default NULL.
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.
make_copy	If FALSE, the inputted <a href="#">musica</a> object is updated and nothing is returned. If TRUE, a new <a href="#">musica</a> object is created and returned. Default FALSE.
table_name	Use modality instead

**Value**

Returns nothing or a new [musica](#) object, depending on the make\_copy parameter.

**Examples**

```
data(musica)
g <- select_genome("19")
build_standard_table(musica, g, "SBS96", overwrite = TRUE)
discover_signatures(
  musica = musica, modality = "SBS96",
```

```
    num_signatures = 3, algorithm = "lda", seed = 12345, nstart = 1
  )
```

---

drop_annotation	<i>Drops a column from the variant table that the user no longer needs</i>
-----------------	--

---

### Description

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

### Usage

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

### Arguments

musica	A <a href="#">musica</a> object.
column_name	Name of column to drop

### Value

None

### Examples

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

---

exposures	<i>Retrieve exposures from a result_model, result_collection, or musica object</i>
-----------	--

---

### 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(x, ...)  
  
## S4 method for signature 'musica'  
exposures(x, result, modality, model_id)  
  
## S4 method for signature 'result_collection'  
exposures(x, modality, model_id)  
  
## S4 method for signature 'result_model'  
exposures(x)  
  
exposures(x, ...) <- value  
  
## S4 replacement method for signature 'musica,matrix'  
exposures(x, result, modality, model_id) <- value  
  
## S4 replacement method for signature 'result_collection,matrix'  
exposures(x, modality, model_id) <- value  
  
## S4 replacement method for signature 'result_model,matrix'  
exposures(x) <- value
```

## Arguments

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs
result	Name of result from result_list to assign the exposures. Used when result is a <a href="#">musica</a> object.
modality	Modality to assign the exposures. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the exposures. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
value	A matrix of samples by signature exposures

## Value

A matrix of exposures

## Examples

```
data(res)  
exposures(res, "result", "SBS96", "res")  
data(res)  
exposures(res, "result", "SBS96", "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.

## Usage

```
exposure_differential_analysis(  
  musica,  
  model_name,  
  annotation,  
  modality = "SBS96",  
  result_name = "result",  
  method = c("wilcox", "kruskal", "glm.nb"),  
  group1 = NULL,  
  group2 = NULL,  
  ...  
)
```

## Arguments

<code>musica</code>	A <code>musica</code> object.
<code>model_name</code>	The name of the model.
<code>annotation</code>	Column in the <code>sample_annotated</code> table of the <code>musica</code> object
<code>modality</code>	The modality. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
<code>result_name</code>	Name of the result list entry. Default "result".
<code>method</code>	Any method in <code>c("wilcox", "kruskal", "glm.nb")</code> used to perform differential analysis on signature exposures
<code>group1</code>	character vector used in the Wilcox test. Elements in <code>group1</code> are compared to elements in <code>group2</code> . This is required for annotation with more than 2 levels.
<code>group2</code>	character vector used in the Wilcox test. Elements in <code>group2</code> are compared to elements in <code>group1</code> . This is required for annotation with more than 2 levels.
<code>...</code>	Additional arguments to be passed to the chosen method

## Value

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

### Examples

```
data("res_annot")
exposure_differential_analysis(res_annot,
  model_name = "res_annot",
  annotation = "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,
  multiallele = 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 <a href="#">CollapsedVCF</a> , <a href="#">ExpandedVCF</a> , <a href="#">MAF</a> , 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 <a href="#">extract_variants_from_vcf</a> 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 <a href="#">extract_variants_from_vcf</a> 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 <a href="#">extract_variants_from_vcf</a> for more details. Default NULL.
filename_as_id	If set to TRUE, the file name will be used as the sample name. See <a href="#">extract_variants_from_vcf_file</a> 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 <a href="#">extract_variants_from_vcf_file</a> for more details. Only used if the input is a vcf file. Default c(".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 <a href="#">extract_variants_from_vcf</a> 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 <a href="#">extract_variants_from_vcf_file</a> 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".
sample_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".
verbose	Show progress of variant extraction. Default TRUE.

### Value

Returns a data.table of variants from a vcf

### Examples

```
# Get loations 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`*Extract variants from a maf object*

---

**Description**

Add description

**Usage**

```
extract_variants_from_maf(maf, extra_fields = NULL)
```

**Arguments**

`maf` MAF object loaded by `read.maf()` from the 'maftools' package  
`extra_fields` Optionally extract additional columns from the maf object. Default NULL.

**Value**

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

**Examples**

```
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**

```
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**

```
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**

```
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.

**Examples**

```
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"
)
```

---

extract\_variants\_from\_vcf

*Extracts variants from a VariantAnnotation VCF object*

---

**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**

```
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**

```
vcf_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**

```

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**

<code>vcf_file</code>	Path to the vcf file
<code>id</code>	ID of the sample to select from VCF. If NULL, then the first sample will be selected. Default NULL.
<code>rename</code>	Rename the sample to this value when extracting variants. If NULL, then the sample will be named according to ID.
<code>sample_field</code>	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 <code>id</code> parameter ("TUMOR" in this example), then <code>sample_field</code> can be used to specify the name of the tag ("SampleName" in this example). Default NULL.
<code>filename_as_id</code>	If set to TRUE, the file name will be used as the sample name.
<code>strip_extension</code>	Only used if <code>filename_as_id</code> 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 be removed from the end of the filename before setting the sample name. Default <code>c(".vcf", ".vcf.gz", ".gz")</code>
<code>filter</code>	Exclude variants that do not have a PASS in the FILTER column of the VCF. Default TRUE.
<code>multiallele</code>	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".
<code>extra_fields</code>	Optionally extract additional fields from the INFO section of the VCF. Default NULL.
<code>fix_vcf_errors</code>	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)
```

---

generate\_result\_grid *Generate result\_grid from musica based on annotation and range of k*

---

**Description**

Generate result\_grid from musica based on annotation and range of k

**Usage**

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

**Arguments**

musica	A <a href="#">musica</a> object.
modality	Modality 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
result_name	Name for result_list entry to save the results to. Default "result_grid".
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
make_copy	If FALSE, the inputted <code>musica</code> object is updated and nothing is returned. If TRUE, a new <code>musica</code> object is created and returned. Default FALSE.
table_name	Use modality instead

**Value**

Returns nothing or a new `musica` object, depending on the `make_copy` parameter.

**Examples**

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

---

get_count_table	<i>Retrieve count_table matrix from count_table object</i>
-----------------	--

---

**Description**

The count table

**Usage**

```
get_count_table(count_table)
```

**Arguments**

`count_table` A `count_table` object.

**Value**

The count table

---

get_modality	<i>Retrieve a specific modality entry from a musica or result_collection object</i>
--------------	---

---

### Description

modality list contains model results for a modality

### Usage

```
get_modality(x, ...)  
  
## S4 method for signature 'musica'  
get_modality(x, result, modality)  
  
## S4 method for signature 'result_collection'  
get_modality(x, modality)
```

### Arguments

x	A <a href="#">result_model</a> or <a href="#">result_collection</a> object
...	Other inputs
result	The name of the result_list entry.
modality	The modality.

### Value

A list of modality which contains result\_model objects

### Examples

```
data(res)  
get_modality(res, "result", "SBS96")
```

---

get_model	<i>Retrieve model from a musica or result collection object</i>
-----------	---

---

### Description

Extract the [result\\_model](#) object from the [musica](#) or [result\\_collection](#) object that contains the model.



**Usage**

```

get_model(x, ...)

## S4 method for signature 'musica'
get_model(x, result, modality, model)

## S4 method for signature 'result_collection'
get_model(x, modality, model)

```

**Arguments**

x	A <a href="#">musica</a> or <a href="#">result_collection</a> object.
...	Other inputs
result	The name of the <code>result_list</code> entry.
modality	The modality.
model	The name of the model.

**Value**

A [result\\_model](#) object

**Examples**

```

data(res)
get_model(res, "result", "SBS96", "res")

```

---

`get_result_list_entry` Retrieve `result_list` entry from a `musica` object

---

**Description**

The `result_list` contains results from various runs

**Usage**

```

get_result_list_entry(object, result_name)

## S4 method for signature 'musica,character'
get_result_list_entry(object, result_name)

```

**Arguments**

object	A <a href="#">musica</a> object generated by the <a href="#">create_musica_from_variants</a> or <a href="#">create_musica_from_counts</a> function.
result_name	The name of the <code>result_list</code> entry.

**Value**

A list of results

**Examples**

```
data(res)
get_result_list_entry(res, "result")
```

---

hyperparameter	<i>Retrieve hyperparameter from a musica or result_collection object</i>
----------------	--

---

**Description**

The hyperparameter contain list of prior and tuning parameters

**Usage**

```
hyperparameter(x, ...)

## S4 method for signature 'musica'
hyperparameter(x, result)

## S4 method for signature 'result_collection'
hyperparameter(x)

hyperparameter(x, ...) <- value

## S4 replacement method for signature 'musica,list'
hyperparameter(x, result) <- value

## S4 replacement method for signature 'result_collection,list'
hyperparameter(x) <- value
```

**Arguments**

x	A <a href="#">result_model</a> or <a href="#">result_collection</a> object
...	Other inputs
result	The name of the result_list entry.
value	A <a href="#">list</a> of hyperparameters for model

**Value**

A list of hyperparameters

**Examples**

```
data(res)
hyperparameter(res, "result")
```

---

indel_musica	<i>indel_musica</i>
--------------	---------------------

---

**Description**

A musica created for testing that includes INDEL variants

**Usage**

```
data(indel_musica)
```

**Format**

An object of class musica See [create\_musica\_from\_variants()] and [create\_musica\_from\_counts()].

---

k_select	<i>Plots for helping decide number of clusters</i>
----------	--

---

**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**

```
k_select(
  musica,
  model_name,
  modality = "SBS96",
  result_name = "result",
  method = "wss",
  clust.method = "kmeans",
  n = 10,
  proportional = TRUE
)
```

**Arguments**

musica	A <a href="#">musica</a> object containing a mutational discovery or prediction. A two-dimensional UMAP has to be stored in this object.
model_name	The name of the desired model.
modality	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
result_name	Name of the result list entry containing desired model. Default "result".

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**

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

---

metrics	<i>Retrieve metrics from a result_model, result_collection, or musica object</i>
---------	--

---

**Description**

Metrics for the model

**Usage**

```
metrics(x, ...)

## S4 method for signature 'musica'
metrics(x, result, modality, model_id)

## S4 method for signature 'result_collection'
metrics(x, modality, model_id)

## S4 method for signature 'result_model'
metrics(x)
```

```

metrics(x, ...) <- value

## S4 replacement method for signature 'musica,SimpleList'
metrics(x, result, modality, model_id) <- value

## S4 replacement method for signature 'result_collection,SimpleList'
metrics(x, modality, model_id) <- value

## S4 replacement method for signature 'result_model,SimpleList'
metrics(x) <- value

```

### Arguments

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs
result	Name of result from <a href="#">result_list</a> to assign the metrics. Used when result is a <a href="#">musica</a> object.
modality	Modality to assign the metrics. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the metrics. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
value	List of metrics

### Value

The metrics for the model

### Examples

```

data(res)
metrics(res, "result", "SBS96", "res")

```

---

modality	<i>Retrieve modality from a <a href="#">result_model</a>, <a href="#">result_collection</a>, or <a href="#">musica</a> object</i>
----------	---

---

### Description

The modality

## Usage

```
modality(x, ...)  
  
## S4 method for signature 'result_collection'  
modality(x, modality, model_id)  
  
## S4 method for signature 'result_model'  
modality(x)  
  
modality(x, ...) <- value  
  
## S4 replacement method for signature 'musica,matrix'  
modality(x, result, modality, model_id) <- value  
  
## S4 replacement method for signature 'result_collection,matrix'  
modality(x, modality, model_id) <- value  
  
## S4 replacement method for signature 'result_model,matrix'  
modality(x) <- value
```

## Arguments

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs <a href="#">result_collection</a> object.
modality	Modality to assign the modality. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the modality. Used when result is a <a href="#">musica</a> or
value	A modality
result	Name of result from <a href="#">result_list</a> to assign the modality. Used when result is a <a href="#">musica</a> object.

## Value

The modality for the model

## Examples

```
data(res)  
modality(res, "result", "SBS96", "res")
```

---

model_id	<i>Retrieve model_id from a result_model, result_collection, or musica object</i>
----------	---

---

**Description**

Model identifier

**Usage**

```

model_id(x, ...)

## S4 method for signature 'musica'
model_id(x, result, modality, model_id)

## S4 method for signature 'result_collection'
model_id(x, modality, model_id)

## S4 method for signature 'result_model'
model_id(x)

model_id(x, ...) <- value

## S4 replacement method for signature 'musica,matrix'
model_id(x, result, modality, model_id) <- value

## S4 replacement method for signature 'result_collection,matrix'
model_id(x, modality, model_id) <- value

## S4 replacement method for signature 'result_model,matrix'
model_id(x) <- value

## S4 method for signature 'musica'
modality(x, result, modality, model_id)

```

**Arguments**

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs
result	Name of result from result_list to assign the model_id. Used when result is a <a href="#">musica</a> object.
modality	Modality to assign the model_id. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the model_id. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
value	Model identifier

**Value**

The `model_id` for the model

**Examples**

```
data(res)
model_id(res, "result", "SBS96", "res")
```

---

musica	<i>musica</i>
--------	---------------

---

**Description**

A musica created for testing that includes SBS variants

**Usage**

```
data(musica)
```

**Format**

An object of class musica See [`create_musica_from_variants()`] and [`create_musica_from_counts()`].

---

musica-class	<i>The primary object that contains variants, count_tables, and samples annotations</i>
--------------	---

---

**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)  
`result_list` Results from various algorithms, modalities, and models



---

musicatk	<i>Starts the musicatk interactive Shiny app</i>
----------	--

---

**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**

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

---

musica_annot	<i>musica_annot</i>
--------------	---------------------

---

**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\_from\_variants()] and [create\_musica\_from\_counts()].

---

musica_sbs96	<i>musica_sbs96</i>
--------------	---------------------

---

**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()].

---

musica_sbs96_tiny	<i>musica_sbs96_tiny</i>
-------------------	--------------------------

---

**Description**

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

**Usage**

```
data(musica_sbs96_tiny)
```

**Format**

An object of class musica See [build\_standard\_table()].

---

name_signatures	<i>Rename signatures for a model</i>
-----------------	--------------------------------------

---

**Description**

Rename signatures for a model

**Usage**

```
name_signatures(
  musica,
  model_id,
  name_vector,
  modality = "SBS96",
  result_name = "result"
)
```

**Arguments**

musica	A <code>musica</code> object containing a mutational signature discovery or prediction.
model_id	The name of the model to rename signatures for.
name_vector	Vector of user-defined signature names
modality	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
result_name	Name of the result list entry containing the model. Default "result".

**Value**

Musica object with user-defined signatures names

**Examples**

```
data(res)
name_signatures(res,
  model_id = "res",
  name_vector = c("smoking", "apobec", "unknown")
)
```

---

num_signatures	<i>Retrieve num_signatures from a result_model, result_collection, or musica object</i>
----------------	---

---

**Description**

The number of signatures in a model

**Usage**

```
num_signatures(x, ...)

## S4 method for signature 'musica'
num_signatures(x, result, modality, model_id)

## S4 method for signature 'result_collection'
num_signatures(x, modality, model_id)

## S4 method for signature 'result_model'
num_signatures(x)

num_signatures(x, ...) <- value

## S4 replacement method for signature 'musica,matrix'
num_signatures(x, result, modality, model_id) <- value
```

```
## S4 replacement method for signature 'result_collection,matrix'
num_signatures(x, modality, model_id) <- value
```

```
## S4 replacement method for signature 'result_model,matrix'
num_signatures(x) <- value
```

### Arguments

x	A <code>musica</code> , <code>result_collection</code> , or <code>result_model</code> object generated by a mutational discovery or prediction tool.
...	Other inputs
result	Name of result from <code>result_list</code> to assign the <code>num_signatures</code> . Used when result is a <code>musica</code> object.
modality	Modality to assign the <code>num_signatures</code> . Used when result is a <code>musica</code> or <code>result_collection</code> object.
model_id	Model identifier to assign the <code>num_signatures</code> . Used when result is a <code>musica</code> or <code>result_collection</code> object.
value	Number of signatures in the model

### Value

The number of signatures in a model

### Examples

```
data(res)
num_signatures(res, "result", "SBS96", "res")
```

---

other_parameters	<i>Retrieve other_parameters from a result_model, result_collection, or musica object</i>
------------------	---

---

### Description

Parameters for the model

### Usage

```
other_parameters(x, ...)

## S4 method for signature 'musica'
other_parameters(x, result, modality, model_id)

## S4 method for signature 'result_collection'
other_parameters(x, modality, model_id)
```

```
## S4 method for signature 'result_model'
other_parameters(x)

other_parameters(x, ...) <- value

## S4 replacement method for signature 'musica,matrix'
other_parameters(x, result, modality, model_id) <- value

## S4 replacement method for signature 'result_collection,matrix'
other_parameters(x, modality, model_id) <- value

## S4 replacement method for signature 'result_model,matrix'
other_parameters(x) <- value
```

### Arguments

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs
result	Name of result from result_list to assign the other_parameters. Used when result is a <a href="#">musica</a> object.
modality	Modality to assign the other_parameters. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the other_parameters. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
value	List of other parameters

### Value

The other parameters for the model

### Examples

```
data(res)
other_parameters(res, "result", "SBS96", "res")
```

---

parameter	<i>Retrieve parameter from a musica or result_collection object</i>
-----------	---

---

### Description

The parameter contains input parameters used in the model

**Usage**

```

parameter(x, ...)

## S4 method for signature 'musica'
parameter(x, result)

## S4 method for signature 'result_collection'
parameter(x)

parameter(x, ...) <- value

## S4 replacement method for signature 'result_collection,list'
parameter(x) <- value

## S4 replacement method for signature 'musica,list'
parameter(x, result) <- value

```

**Arguments**

x	A <a href="#">result_model</a> or <a href="#">result_collection</a> object
...	Other inputs
result	The name of the result_list entry.
value	a list of input parameters

**Value**

a list of parameters

**Examples**

```

data(res)
parameter(res, "result")

```

---

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**

```
plot_cluster(  
  musica,  
  model_name,  
  modality = "SBS96",  
  result_name = "result",  
  clusters,  
  group = "signature",  
  annotation = NULL,  
  plotly = TRUE  
)
```

**Arguments**

musica	A <a href="#">musica</a> object containing a mutational discovery or prediction. A two-dimensional UMAP has to be stored in this object.
model_name	The name of the desired model.
modality	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
result_name	Name of the result list entry containing desired model. Default "result".
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**

```
set.seed(123)  
data(res_annot)  
# Get clustering result  
clust_out <- cluster_exposure(  
  musica = res_annot, model_name = "res_annot",  
  nclust = 2, iter.max = 15  
)  
# UMAP  
create_umap(musica = res_annot, model_name = "res_annot")  
# generate cluster X signature plot  
plot_cluster(  
  musica = res_annot, model_name = "res_annot",  
  modality = "SBS96", result_name = "result",  
  clusters = clust_out, group = "signature",  
  annotation = "res_annot", plotly = TRUE  
)
```

```
    musica = res_annot, model_name = "res_annot",
    clusters = clust_out, group = "signature"
  )
  # generate cluster X annotation plot
  plot_cluster(
    musica = res_annot, model_name = "res_annot",
    clusters = clust_out, group = "annotation",
    annotation = "Tumor_Subtypes"
  )
  # generate a single UMAP plot
  plot_cluster(
    musica = res_annot, model_name = "res_annot",
    clusters = clust_out, group = "none"
  )
)
```

---

plot\_differential\_analysis

*Compare exposures of annotated samples*

---

### 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,
  model_name = "res_annot",
  annotation = "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(  
  musica,  
  model_name,  
  modality = "SBS96",  
  result_name = "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

<code>musica</code>	A <code>musica</code> object containing a mutational discovery or prediction.
<code>model_name</code>	The name of the desired model.
<code>modality</code>	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
<code>result_name</code>	Name of the result list entry containing desired model. Default "result".
<code>plot_type</code>	One of "bar", "box", or "violin". Default "bar".
<code>proportional</code>	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 <code>mixedsort</code> 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 <code>plotly</code> . Default FALSE.

**Value**

Generates a ggplot or plotly object

**Examples**

```
data(res_annot)
plot_exposures(res_annot,
  model_name = "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.

## Usage

```
plot_heatmap(  
  musica,  
  model_name,  
  modality = "SBS96",  
  result_name = "result",  
  proportional = FALSE,  
  show_column_names = FALSE,  
  show_row_names = TRUE,  
  scale = TRUE,  
  subset_tumor = NULL,  
  subset_signatures = NULL,  
  annotation = NULL,  
  ...  
)
```

## Arguments

<code>musica</code>	A <code>musica</code> object containing a mutational discovery or prediction.
<code>model_name</code>	The name of the desired model.
<code>modality</code>	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
<code>result_name</code>	Name of the result list entry containing desired model. Default "result".
<code>proportional</code>	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.
<code>show_column_names</code>	Boolean check. If True, column names are shown. Otherwise, they aren't. Default FALSE
<code>show_row_names</code>	Boolean check. If True, row names are shown. Otherwise, they aren't. Default FALSE
<code>scale</code>	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**

```
data(res_annot)
plot_heatmap(
  musica = res_annot, model_name = "res_annot",
  proportional = TRUE, scale = TRUE, annotation = "Tumor_Subtypes"
)
```

---

plot\_k\_comparison      *Plot k comparison*

---

**Description**

Plot the results of comparing k values

**Usage**

```
plot_k_comparison(k_comparison)
```

**Arguments**

k\_comparison      data.frame with k value comparisons returned from the [compare\\_k\\_vals](#) function.

**Value**

a ggplot figure

### Examples

```
data(musica)
k_comparison <- compare_k_vals(musica, "SBS96",
  reps = 3, min_k = 1,
  max_k = 5
)
plot_k_comparison(k_comparison)
```

---

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,
  modality = "SBS96",
  text_size = 10,
  show_x_labels = TRUE,
  show_y_labels = TRUE,
  same_scale = TRUE,
  annotation = NULL,
  table_name = NULL
)
```

### Arguments

musica	A <a href="#">musica</a> object.
sample_names	Names of the samples to plot.
modality	Name of table used for plotting counts. Default "SBS96".
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.
table_name	Use modality instead

**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(
  musica,
  sample,
  model_id,
  modality = "SBS96",
  result_name = "result",
  plotly = FALSE
)
```

**Arguments**

musica	A <a href="#">musica</a> object.
sample	Name of the sample within the <a href="#">musica</a> object.
model_id	The name of the desired model.
modality	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
result_name	Name of the result list entry containing desired model. Default "result".
plotly	If TRUE, the the plot will be made interactive using <a href="#">plotly</a> . Default FALSE.

**Value**

Generates a ggplot or plotly object

## Examples

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

---

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(  
  musica,  
  model_id,  
  modality = "SBS96",  
  result_name = "result",  
  color_variable = NULL,  
  color_mapping = NULL,  
  text_size = 10,  
  show_x_labels = TRUE,  
  show_y_labels = TRUE,  
  same_scale = FALSE,  
  y_max = NULL,  
  annotation = NULL,  
  percent = TRUE,  
  plotly = FALSE  
)
```

## Arguments

<code>musica</code>	A <a href="#">musica</a> object containing a mutational discovery or prediction.
<code>model_id</code>	The name of the model to plot.
<code>modality</code>	The modality of the signatures to plot. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
<code>result_name</code>	Name of the result list entry containing the signatures to plot. Default "result".
<code>color_variable</code>	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 <a href="#">musica</a> 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 color_variable to a color. The items in color_mapping correspond to the colors. The names of the items in color_mapping should correspond to the unique items in color_variable. If NULL, then the default 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 FALSE.
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.
plotly	If TRUE, the the plot will be made interactive using <a href="#">plotly</a> . Default FALSE.

**Value**

Generates a ggplot or plotly object

**Examples**

```
data(res)
plot_signatures(res, model_id = "res")
```

---

plot\_umap

*Plot a UMAP from a musica result*

---

**Description**

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

**Usage**

```
plot_umap(
  musica,
  model_name,
  modality = "SBS96",
  result_name = "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

<code>musica</code>	A <code>musica</code> object containing a mutational signature discovery or prediction.
<code>model_name</code>	The name of the desired model.
<code>modality</code>	The modality of the model. Must be "SBS96", "DBS78", or "IND83". Default "SBS96".
<code>result_name</code>	Name of the result list entry containing the model. Default "result".
<code>color_by</code>	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".
<code>proportional</code>	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.
<code>annotation</code>	Sample annotation used to color the points. One used when <code>color_by = "annotation"</code> . Default NULL.
<code>point_size</code>	Scatter plot point size. Default 0.7.
<code>same_scale</code>	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 <code>color_by = "signature"</code> . Setting to FALSE is most useful when the maximum value of various signatures are vastly different from one another. Default TRUE.
<code>add_annotation_labels</code>	If TRUE, labels for each group in the annotation variable will be displayed. Only used if <code>color_by = "annotation"</code> . 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.
<code>annotation_label_size</code>	Size of annotation labels. Only used if <code>color_by = "annotation"</code> and <code>add_annotation_labels = TRUE</code> . Default 3.

annotation_text_box	Place a white box around the annotation labels to improve readability. Only used if color_by = "annotation" and add_annotation_labels = TRUE. Default TRUE.
plotly	If TRUE, the the plot will be made interactive using <a href="#">plotly</a> . 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, "res_annot")
plot_umap(res_annot, "res_annot", color_by = "none")
```

---

predict_exposure	<i>Prediction of exposures in new samples using pre-existing signatures</i>
------------------	---

---

**Description**

Exposures for samples will be predicted using an existing set of signatures stored in a [result\\_model](#) object. Algorithms available for prediction include a modify version of "lda", and "decompTumor2Sig".

**Usage**

```
predict_exposure(
  musica,
  modality,
  signature_res,
  algorithm = c("lda", "decompTumor2Sig"),
  result_name = "result",
  model_id = NULL,
  signatures_to_use = seq_len(ncol(signatures(signature_res))),
  verbose = FALSE,
  make_copy = FALSE,
  table_name = NULL
)
```

**Arguments**

musica	A <a href="#">musica</a> object.
modality	Modality 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 be stored in a <a href="#">result_model</a> object.
algorithm	Algorithm to use for prediction of exposures. One of "lda" or "decompTumor2Sig".
result_name	Name for result_list entry to save the results to. Default "result".
model_id	Identifier for the result. If NULL, will be automatically set to the algorithm and number of signatures. Default NULL.
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.
make_copy	If FALSE, the inputted <a href="#">musica</a> object is updated and nothing is returned. If TRUE, a new <a href="#">musica</a> object is created and returned. Default FALSE.
table_name	Use modality instead

**Value**

Returns nothing or a new [musica](#) object, depending on the make\_copy parameter.

**Examples**

```

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

```

---

rc	<i>Reverse complement of a string using biostrings</i>
----	--

---

**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	<i>Replication Timing Data as GRanges Object</i>
-----------	--

---

**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**

GEO, <<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE134225>>

**References**

Sriramachandran, A. M. et al. (2020) Genome-wide Nucleotide-Resolution Mapping of DNA Replication Patterns, Single-Strand Breaks, and Lesions by GLOE-Seq. ([Molecular Cell] (doi:10.1016/j.molcel.2020.03.027

---

res	res
-----	-----

---

**Description**

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

**Usage**

data(res)

**Format**

An object of class musica See [discover\_signatures()].

---

result\_collection-class

*The Result Collection object that contains modality, input parameters, prior hyperparameters*

---

**Description**

The Result Collection object that contains modality, input parameters, prior hyperparameters

**Slots**

modality a list contains model results for different modality

parameter a list contains input parameters

hyperparameter a list contains prior and tuning parameters

---

result\_list

*Retrieve result\_list from a musica object*

---

**Description**

The result\_list contains results from various runs

**Usage**

```

result_list(object)

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

result_list(musica) <- value

## S4 replacement method for signature 'musica,SimpleList'
result_list(musica) <- value

```

**Arguments**

object	A <i>musica</i> object generated by the <a href="#">create_musica_from_variants</a> or <a href="#">create_musica_from_counts</a> function.
musica	A <i>musica</i> object generated by the <a href="#">create_musica_from_variants</a> or <a href="#">create_musica_from_counts</a> function
value	A list of results

**Value**

A list of results

**Examples**

```

data(res)
result_list(res)

```

---

result_model-class	<i>Object that contains results for a single model</i>
--------------------	--

---

**Description**

Object that contains results for a single model

**Slots**

signatures	A matrix of signatures by mutational motifs
exposures	A matrix of samples by signature weights
num_signatures	Number of signatures in the model
other_parameters	Parameters relevant to the model
credible_intervals	Credible intervals for parameters
metrics	Performance metrics for the model
umap	List of umap data.frames for plotting and analysis
model_id	Model identifier
modality	Modality of result (SBS96, DBS78, IND83)

---

res_annot	<i>res_annot</i>
-----------	------------------

---

**Description**

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

**Usage**

```
data(res_annot)
```

**Format**

An object of class musica See [discover\_signatures()].

---

sample_names	<i>Retrieve sample names from a musica object</i>
--------------	---

---

**Description**

Sample names were included in the sample column in the variant object passed to [create\\_musica\\_from\\_variants](#), or in the colnames of the count table object passed to [create\\_musica\\_from\\_counts](#). This returns a unique list of samples names in the order they are inside the musica object.

**Usage**

```
sample_names(object)

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

**Arguments**

object	A <i>musica</i> object generated by the <a href="#">create_musica_from_variants</a> or <a href="#">create_musica_from_counts</a> function.
--------	--

**Value**

A character vector of sample names

**Examples**

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

---

samp_annot	<i>Get or set sample annotations from a musica object</i>
------------	---

---

### 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

```
samp_annot(object)

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

samp_annot(object, name) <- value

## S4 replacement method for signature 'musica,character,vector'
samp_annot(object, name) <- value
```

### Arguments

object	A <a href="#">musica</a> object generated by the <a href="#">create_musica_from_variants</a> or <a href="#">create_musica_from_counts</a> function.
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](#) 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	<i>Helper function to load common human or mouse genomes</i>
---------------	--

---

**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	<i>Retrieve signatures from a result_model, result_collection, or musica object</i>
------------	---

---

**Description**

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

**Usage**

```
signatures(x, ...)

## S4 method for signature 'musica'
signatures(x, result, modality, model_id)

## S4 method for signature 'result_collection'
signatures(x, modality, model_id)

## S4 method for signature 'result_model'
signatures(x)
```

```

signatures(x, ...) <- value

## S4 replacement method for signature 'musica,matrix'
signatures(x, result, modality, model_id) <- value

## S4 replacement method for signature 'result_collection,matrix'
signatures(x, modality, model_id) <- value

## S4 replacement method for signature 'result_model,matrix'
signatures(x) <- value

```

### Arguments

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs
result	Name of result from result_list to assign the signatures. Used when result is a <a href="#">musica</a> object.
modality	Modality to assign the signatures. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the signatures. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
value	A matrix of motifs counts by samples

### Value

A matrix of mutational signatures

### Examples

```

data(res)
signatures(res, "result", "SBS96", "res")
data(res)
signatures(res, "result", "SBS96", "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

### Usage

```
subset_musica_by_annotation(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**

```

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**

```
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

**Examples**

```
data(musica_sbs96)
subset_musica_by_counts(musica_sbs96, "SBS96", 20)
```

---

```
subset_variants_by_samples
```

*Return sample from musica\_variant object*

---

**Description**

Return sample from musica\_variant object

**Usage**

```
subset_variants_by_samples(musica, sample_name)
```

**Arguments**

musica	A <a href="#">musica</a> object.
sample_name	Sample name to subset by

**Value**

Returns sample data.frame subset to a single sample

**Examples**

```
data(musica)
subset_variants_by_samples(musica, "TCGA-94-7557-01A-11D-2122-08")
```

---

```
subset_variant_by_type
```

*Subsets a variant table based on Variant Type*

---

**Description**

Subsets a variant table based on Variant Type

**Usage**

```
subset_variant_by_type(tab, type)
```

**Arguments**

tab	Input variant table
type	Variant type to return e.g. "SBS", "INS", "DEL", "DBS"

**Value**

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

**Examples**

```
data(musica)
annotate_variant_type(musica)
subset_variant_by_type(variants(musica), "SBS")
```

---

tables	<i>Retrieve the list of count_tables from a musica object</i>
--------	---

---

**Description**

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

**Usage**

```
tables(object)

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

tables(musica) <- value

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

**Arguments**

object	A <a href="#">musica</a> object generated by the <a href="#">create_musica_from_variants</a> or <a href="#">create_musica_from_counts</a> function.
musica	A <a href="#">musica</a> object generated by the <a href="#">create_musica_from_variants</a> or <a href="#">create_musica_from_counts</a> function.
value	A list of <a href="#">count_table</a> objects representing counts of motifs in samples

**Value**

A list of count\_tables

**Examples**

```
data(res)
tables(res)
```

---

table_96	<i>Generates a 96 motif table based on input counts for plotting</i>
----------	--

---

**Description**

Generates a 96 motif table based on input counts for plotting

**Usage**

```
table_96(sample_df)
```

**Arguments**

sample\_df      Input counts table

**Value**

Returns a 96 motif summary table

---

table_selected	<i>Retrieve table name used for plotting from a result_model object</i>
----------------	---

---

**Description**

The table name

**Usage**

```
table_selected(result)
```

```
## S4 method for signature 'result_model'  
table_selected(result)
```

**Arguments**

result      A [result\\_model](#) object generated by a mutational discovery or prediction tool.

**Value**

Table name used for plotting

**Examples**

```
data(res)  
model <- get_model(res, "result", "SBS96", "res")  
table_selected(model)
```

---

umap	<i>Retrieve umap list from a result_model, result_collection, or musica object</i>
------	--

---

### Description

The umap dataframes for the model

### Usage

```
umap(x, ...)

## S4 method for signature 'musica'
umap(x, result, modality, model_id)

## S4 method for signature 'result_collection'
umap(x, modality, model_id)

## S4 method for signature 'result_model'
umap(x)

umap(x, ...) <- value

## S4 replacement method for signature 'musica,matrix'
umap(x, result, modality, model_id) <- value

## S4 replacement method for signature 'result_collection,matrix'
umap(x, modality, model_id) <- value

## S4 replacement method for signature 'result_model,matrix'
umap(x) <- value
```

### Arguments

x	A <a href="#">musica</a> , <a href="#">result_collection</a> , or <a href="#">result_model</a> object generated by a mutational discovery or prediction tool.
...	Other inputs
result	Name of result from result_list to assign the umap. Used when result is a <a href="#">musica</a> object.
modality	Modality to assign the umap. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
model_id	Model identifier to assign the umap. Used when result is a <a href="#">musica</a> or <a href="#">result_collection</a> object.
value	A list of umap dataframes

**Value**

A list of umap dataframes

**Examples**

```
data(res)
umap(res, "result", "SBS96", "res")
```

---

variants

*Retrieve variants from a musica object*

---

**Description**

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

**Usage**

```
variants(object)

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

variants(musica) <- value

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

**Arguments**

object	A <code>musica</code> object generated by the <code>create_musica_from_variants</code> or <code>create_musica_from_counts</code> function.
musica	A <code>musica</code> object generated by the <code>create_musica_from_variants</code> or <code>create_musica_from_counts</code> function
value	A <code>data.table</code> of mutational variants and variant-level annotations

**Value**

A `data.table` of variants

**Examples**

```
data(res)
variants(res)
```



---

%>%

*Pipe operator*

---

**Description**

Pipe operator

**Usage**

lhs %>% rhs

**Value**

NA

**Examples**

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

# Index

- \* **datasets**
  - [cosmic\\_v2\\_sigs](#), [22](#)
  - [cosmic\\_v3\\_dbs\\_sigs](#), [23](#)
  - [cosmic\\_v3\\_indel\\_sigs](#), [24](#)
  - [cosmic\\_v3\\_sbs\\_sigs](#), [24](#)
  - [cosmic\\_v3\\_sbs\\_sigs\\_exome](#), [25](#)
  - [dbs\\_musica](#), [33](#)
  - [indel\\_musica](#), [51](#)
  - [musica](#), [56](#)
  - [musica\\_annot](#), [57](#)
  - [musica\\_sbs96](#), [58](#)
  - [musica\\_sbs96\\_tiny](#), [58](#)
  - [rep\\_range](#), [76](#)
  - [res](#), [77](#)
  - [res\\_annot](#), [79](#)
- \* **internal**
  - [.jsd](#), [4](#)
  - [%>%](#), [89](#)
  - [add\\_variant\\_type](#), [6](#)
  - [auto\\_subset\\_sigs](#), [10](#)
  - [create\\_dbs78\\_table](#), [26](#)
  - [create\\_sbs192\\_table](#), [30](#)
  - [create\\_sbs96\\_table](#), [30](#)
  - [table\\_96](#), [86](#)
  - [.jsd](#), [4](#)
  - [%>%](#), [89](#)
  - [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](#), [13](#), [28](#), [30](#)
  - [build\\_custom\\_table](#), [11](#)
  - [build\\_standard\\_table](#), [12](#), [33](#), [34](#)
  - [built\\_tables](#), [14](#)
  - [built\\_tables,musica-method](#)  
([built\\_tables](#)), [14](#)
  - [cluster\\_exposure](#), [14](#)
  - [CollapsedVCF](#), [39](#)
  - [combine\\_count\\_tables](#), [16](#)
  - [combine\\_predict\\_grid](#), [17](#)
  - [compare\\_cosmic\\_v2](#), [18](#)
  - [compare\\_cosmic\\_v3](#), [19](#)
  - [compare\\_k\\_vals](#), [20](#), [68](#)
  - [compare\\_results](#), [21](#)
  - [cosmic\\_v2\\_sigs](#), [22](#)
  - [cosmic\\_v2\\_subtype\\_map](#), [23](#)
  - [cosmic\\_v3\\_dbs\\_sigs](#), [23](#)
  - [cosmic\\_v3\\_indel\\_sigs](#), [24](#)
  - [cosmic\\_v3\\_sbs\\_sigs](#), [24](#)
  - [cosmic\\_v3\\_sbs\\_sigs\\_exome](#), [25](#)
  - [count\\_table](#), [47](#), [85](#)
  - [count\\_table-class](#), [25](#)
  - [create\\_dbs78\\_table](#), [26](#)
  - [create\\_musica\\_from\\_counts](#), [14](#), [27](#), [33](#), [49](#),  
[78–80](#), [85](#), [88](#)
  - [create\\_musica\\_from\\_variants](#), [14](#), [27](#), [33](#),  
[49](#), [78–80](#), [85](#), [88](#)
  - [create\\_result\\_model](#), [29](#)
  - [create\\_sbs192\\_table](#), [30](#)
  - [create\\_sbs96\\_table](#), [30](#)
  - [create\\_umap](#), [31](#), [63](#), [74](#)
  - [credible\\_intervals](#), [32](#)
  - [credible\\_intervals,musica-method](#)  
([credible\\_intervals](#)), [32](#)
  - [credible\\_intervals,result\\_collection-method](#)  
([credible\\_intervals](#)), [32](#)
  - [credible\\_intervals,result\\_model-method](#)  
([credible\\_intervals](#)), [32](#)
  - [credible\\_intervals<-](#)  
([credible\\_intervals](#)), [32](#)
  - [credible\\_intervals<-,musica,matrix-method](#)  
([credible\\_intervals](#)), [32](#)

- credible\_intervals<- ,result\_collection,matrix-method (credible\_intervals), 32
- credible\_intervals<- ,result\_model,matrix-method (credible\_intervals), 32
- data.table, 88
- dbs\_musica, 33
- discover\_signatures, 33
- drop\_annotation, 35
- ExpandedVCF, 39
- exposure\_differential\_analysis, 37
- exposures, 35
- exposures,musica-method (exposures), 35
- exposures,result\_collection-method (exposures), 35
- exposures,result\_model-method (exposures), 35
- exposures<- (exposures), 35
- exposures<- ,musica,matrix-method (exposures), 35
- exposures<- ,result\_collection,matrix-method (exposures), 35
- exposures<- ,result\_model,matrix-method (exposures), 35
- extract\_count\_tables, 38
- extract\_variants, 38
- extract\_variants\_from\_maf, 41
- extract\_variants\_from\_maf\_file, 41
- extract\_variants\_from\_matrix, 42
- extract\_variants\_from\_vcf, 39, 43
- extract\_variants\_from\_vcf\_file, 39, 40, 44
- fviz\_nbclust, 52
- generate\_result\_grid, 46
- get\_count\_table, 47
- get\_modality, 48
- get\_modality,musica-method (get\_modality), 48
- get\_modality,result\_collection-method (get\_modality), 48
- get\_model, 48
- get\_model,musica-method (get\_model), 48
- get\_model,result\_collection-method (get\_model), 48
- get\_result\_list\_entry, 49
- get\_result\_list\_entry,musica,character-method (get\_result\_list\_entry), 49
- hyperparameter, 50
- hyperparameter,musica-method (hyperparameter), 50
- hyperparameter,result\_collection-method (hyperparameter), 50
- hyperparameter<- (hyperparameter), 50
- hyperparameter<- ,musica,list-method (hyperparameter), 50
- hyperparameter<- ,result\_collection,list-method (hyperparameter), 50
- indel\_musica, 51
- k\_select, 51
- kmeans, 15
- list, 50
- MAF, 39
- metrics, 52
- metrics,musica-method (metrics), 52
- metrics,result\_collection-method (metrics), 52
- metrics,result\_model-method (metrics), 52
- metrics<- (metrics), 52
- metrics<- ,musica,SimpleList-method (metrics), 52
- metrics<- ,result\_collection,SimpleList-method (metrics), 52
- metrics<- ,result\_model,SimpleList-method (metrics), 52
- mixedsort, 66
- modality, 53
- modality,musica-method (model\_id), 55
- modality,result\_collection-method (modality), 53
- modality,result\_model-method (modality), 53
- modality<- (modality), 53
- modality<- ,musica,matrix-method (modality), 53
- modality<- ,result\_collection,matrix-method (modality), 53
- modality<- ,result\_model,matrix-method (modality), 53
- model\_id, 55
- model\_id,musica-method (model\_id), 55
- model\_id,result\_collection-method (model\_id), 55

- model\_id,result\_model-method  
(model\_id), 55
- model\_id<- (model\_id), 55
- model\_id<- ,musica,matrix-method  
(model\_id), 55
- model\_id<- ,result\_collection,matrix-method  
(model\_id), 55
- model\_id<- ,result\_model,matrix-method  
(model\_id), 55
- musica, 6–8, 10–22, 26, 27, 30–38, 46–49, 51,  
53–55, 56, 59–61, 63, 65, 67, 69–71,  
73, 75, 78–80, 82–85, 87, 88
- musica-class, 56
- musica\_annot, 57
- musica\_sbs96, 58
- musica\_sbs96\_tiny, 58
- musicatk, 57
  
- name\_signatures, 58
- num\_signatures, 59
- num\_signatures,musica-method  
(num\_signatures), 59
- num\_signatures,result\_collection-method  
(num\_signatures), 59
- num\_signatures,result\_model-method  
(num\_signatures), 59
- num\_signatures<- (num\_signatures), 59
- num\_signatures<- ,musica,matrix-method  
(num\_signatures), 59
- num\_signatures<- ,result\_collection,matrix-method  
(num\_signatures), 59
- num\_signatures<- ,result\_model,matrix-method  
(num\_signatures), 59
  
- other\_parameters, 60
- other\_parameters,musica-method  
(other\_parameters), 60
- other\_parameters,result\_collection-method  
(other\_parameters), 60
- other\_parameters,result\_model-method  
(other\_parameters), 60
- other\_parameters<- (other\_parameters),  
60
- other\_parameters<- ,musica,matrix-method  
(other\_parameters), 60
- other\_parameters<- ,result\_collection,matrix-method  
(other\_parameters), 60
- other\_parameters<- ,result\_model,matrix-method  
(other\_parameters), 60
  
- parameter, 61
- parameter,musica-method (parameter), 61
- parameter,result\_collection-method  
(parameter), 61
- parameter<- (parameter), 61
- parameter<- ,musica,list-method  
(parameter), 61
- parameter<- ,result\_collection,list-method  
(parameter), 61
- plot\_cluster, 62
- plot\_differential\_analysis, 64
- plot\_exposures, 65, 80
- plot\_heatmap, 67
- plot\_k\_comparison, 68
- plot\_sample\_counts, 69
- plot\_sample\_reconstruction\_error, 70
- plot\_signatures, 71
- plot\_umap, 32, 72, 80
- plotly, 66, 70, 72, 74
- predict\_exposure, 74
  
- rc, 76
- rep\_range, 76
- res, 77
- res\_annot, 79
- result\_collection, 32, 33, 36, 48–50,  
53–55, 60–62, 82, 87
- result\_collection-class, 77
- result\_list, 77
- result\_list,musica-method  
(result\_list), 77
- result\_list<- (result\_list), 77
- result\_list<- ,musica,SimpleList-method  
(result\_list), 77
- result\_model, 29, 31, 32, 36, 48–50, 53–55,  
60–62, 74, 75, 82, 86, 87
- result\_model-class, 78
  
- samp\_annot, 80
- samp\_annot,musica-method (samp\_annot),  
80
- samp\_annot<- (samp\_annot), 80
- samp\_annot<- ,musica,character,vector-method  
(samp\_annot), 80
- sample\_names, 79, 80
- sample\_names,musica-method  
(sample\_names), 79
- select\_genome, 81
- seqlevelsStyle, 28

signatures, [81](#)  
signatures,musica-method (signatures),  
    [81](#)  
signatures,result\_collection-method  
    (signatures), [81](#)  
signatures,result\_model-method  
    (signatures), [81](#)  
signatures<- (signatures), [81](#)  
signatures<-,musica,matrix-method  
    (signatures), [81](#)  
signatures<-,result\_collection,matrix-method  
    (signatures), [81](#)  
signatures<-,result\_model,matrix-method  
    (signatures), [81](#)  
subset\_musica\_by\_annotation, [82](#)  
subset\_musica\_by\_counts, [83](#)  
subset\_variant\_by\_type, [84](#)  
subset\_variants\_by\_samples, [84](#)  
  
table\_96, [86](#)  
table\_selected, [86](#)  
table\_selected,result\_model-method  
    (table\_selected), [86](#)  
tables, [85](#)  
tables,musica-method (tables), [85](#)  
tables<- (tables), [85](#)  
tables<-,musica,list-method (tables), [85](#)  
  
umap, [31](#), [32](#), [87](#)  
umap,musica-method (umap), [87](#)  
umap,result\_collection-method (umap), [87](#)  
umap,result\_model-method (umap), [87](#)  
umap<- (umap), [87](#)  
umap<-,musica,matrix-method (umap), [87](#)  
umap<-,result\_collection,matrix-method  
    (umap), [87](#)  
umap<-,result\_model,matrix-method  
    (umap), [87](#)  
  
variants, [88](#)  
variants,musica-method (variants), [88](#)  
variants<- (variants), [88](#)  
variants<-,musica,data.table-method  
    (variants), [88](#)