

# Package ‘mastR’

April 9, 2025

**Title** Markers Automated Screening Tool in R

**Version** 1.7.0

**Description** mastR is an R package designed for automated screening of signatures of interest for specific research questions. The package is developed for generating refined lists of signature genes from multiple group comparisons based on the results from edgeR and limma differential expression (DE) analysis workflow. It also takes into account the background noise of tissue-specificity, which is often ignored by other marker generation tools. This package is particularly useful for the identification of group markers in various biological and medical applications, including cancer research and developmental biology.

**biocViews** Software, GeneExpression, Transcriptomics, DifferentialExpression, Visualization

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Depends** R (>= 4.3.0)

**Suggests** BiocManager, BiocStyle, BisqueRNA, clusterProfiler, ComplexHeatmap, depmap, enrichplot, ggrepel, ggvenn, gridExtra, jsonlite, knitr, rmarkdown, RobustRankAggreg, rvest, scuttle, singscore, splatter, testthat (>= 3.0.0), UpSetR

**Config/testthat/edition** 3

**Imports** AnnotationDbi, Biobase, dplyr, edgeR, ggplot2, ggpubr, graphics, grDevices, GSEABase, limma, Matrix, methods, msigdb, org.Hs.eg.db, patchwork, SeuratObject, SingleCellExperiment, stats, SummarizedExperiment, tidy, utils

**Collate** 'plot.R' 'DE\_functions.R' 'AllGenerics.R'  
'filter\_subset\_sig-methods.R' 'get\_de\_table-methods.R'  
'get\_degs-methods.R' 'get\_gsc\_sig-methods.R' 'get\_lm\_sig.R'  
'get\_panglao\_sig.R' 'gls2gsc-methods.R' 'gsc\_plot.R'  
'list\_panglao\_organs.R' 'list\_panglao\_types.R'

'mastR-package.R' 'merge\_markers.R' 'pca\_matrix\_plot-methods.R'  
 'pseudo\_samples-methods.R' 'remove\_bg\_exp-methods.R'  
 'sig\_boxplot-methods.R' 'sig\_gseaplot-methods.R'  
 'sig\_heatmap-methods.R' 'sig\_rankdensity\_plot-methods.R'  
 'sig\_scatter\_plot-methods.R' 'subset\_sig\_by\_step.R'

**URL** <https://davislaboratory.github.io/mastR>

**BugReports** <https://github.com/DavisLaboratory/mastR/issues>

**VignetteBuilder** knitr

**Language** en-US

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

**git\_branch** devel

**git\_last\_commit** 492c795

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

**Repository** Bioconductor 3.21

**Date/Publication** 2025-04-09

**Author** Jinjin Chen [aut, cre] (ORCID: <<https://orcid.org/0000-0001-7923-5723>>),  
 Ahmed Mohamed [aut, ctb] (ORCID:  
 <<https://orcid.org/0000-0001-6507-5300>>),  
 Chin Wee Tan [ctb] (ORCID: <<https://orcid.org/0000-0001-9695-7218>>)

**Maintainer** Jinjin Chen <[chen.j@wehi.edu.au](mailto:chen.j@wehi.edu.au)>

## Contents

ccl_2_wide . . . . .	3
ccl_crc_5 . . . . .	4
DEGs_Group . . . . .	5
DEGs_RP . . . . .	6
de_analysis . . . . .	7
filter_subset_sig . . . . .	8
get_degs . . . . .	11
get_de_table . . . . .	13
get_gsc_sig . . . . .	15
get_lm_sig . . . . .	16
get_panglao_sig . . . . .	17
gls2gsc . . . . .	18
gsc_plot . . . . .	18
im_data_6 . . . . .	19
list_panglao_organs . . . . .	19
list_panglao_types . . . . .	20
lm22 . . . . .	20
lm7 . . . . .	22
mastR_Package . . . . .	23
merge_markers . . . . .	23

msigdb_gobp_nk . . . . .	24
nk_markers . . . . .	24
pca_matrix_plot . . . . .	25
pca_matrix_plot_init . . . . .	28
plotPCAbiplot . . . . .	29
plot_diagnostics . . . . .	29
plot_mean_var . . . . .	30
process_data . . . . .	31
pseudo_samples . . . . .	34
pseudo_sample_list . . . . .	36
remove_bg_exp . . . . .	37
remove_bg_exp_mat . . . . .	42
select_sig . . . . .	43
sig_boxplot . . . . .	44
sig_gseaplot . . . . .	47
sig_heatmap . . . . .	50
sig_rankdensity_plot . . . . .	54
sig_scatter_plot . . . . .	56
voom_fit_treat . . . . .	59

**Index** **61**

ccl2\_wide *Convert CCL2 data from long data to wide data.*

**Description**

Convert CCL2 data downloaded by `depmap::depmap_TPM()` from long data into wide matrix, with row names are gene names and column names are depmap IDs.

**Usage**

```
ccl2_wide(ccl2)
```

**Arguments**

ccl2 CCL2 data downloaded by `depmap::depmap_TPM()`

**Value**

a matrix

## Examples

```
data("cclerc_5")
cclerc <- data.frame(
  gene_name = rownames(cclerc_5),
  cclerc_5$counts
) |>
  tidyr::pivot_longer(
    ~gene_name,
    names_to = "depmap_id",
    values_to = "rna_expression"
  )
cclerc_wide <- cclerc_2_wide(cclerc)
```

---

cclerc\_5

*RNA-seq TPM data of 5 CRC cell line samples from CCLE.*

---

## Description

A test DGEList object with RNA-seq RSEM quantified TPM data of 5 CRC cell line samples from CCLE `depmap::depmap_TPM()`.

## Usage

```
data(cclerc_5)
```

## Format

A DGEList of 19177 genes \* 5 samples.

## Value

DGEList

## Source

`depmap::depmap_TPM()`

---

DEGs_Group	<i>return DEGs UP and DOWN list based on intersection or union of comparisons</i>
------------	---

---

### Description

return DEGs UP and DOWN list based on intersection or union of comparisons

### Usage

```
DEGs_Group(
  tfit,
  lfc = NULL,
  p = 0.05,
  assemble = "intersect",
  Rank = "adj.P.Val",
  keep.top = NULL,
  keep.group = NULL,
  ...
)
```

### Arguments

tfit	MArrayLM object generated by <code>limma::treat()</code>
lfc	num, cutoff of logFC for DE analysis
p	num, cutoff of p value for DE analysis
assemble	'intersect' or 'union', whether to select intersected or union genes of different comparisons, default 'intersect'
Rank	character, the variable for ranking DEGs, can be 'logFC', 'adj.P.Val' ..., default 'adj.P.Val'
keep.top	NULL or num, whether to keep top n DEGs of specific comparison
keep.group	NULL or pattern, specify the top DEGs of which comparison or group to be kept
...	omitted

### Value

A list of "UP" and "DOWN" genes

---

DEGs\_RP

*return DEGs UP and DOWN list based on Rank Product*


---

### Description

return DEGs UP and DOWN list based on Rank Product

### Usage

```
DEGs_RP(
  tfit,
  lfc = NULL,
  p = 0.05,
  assemble = "intersect",
  Rank = "adj.P.Val",
  nperm = 1e+05,
  thres = 0.05,
  keep.top = NULL,
  keep.group = NULL,
  ...
)
```

### Arguments

tfit	MArrayLM object generated by <code>limma::treat()</code>
lfc	num, cutoff of logFC for DE analysis
p	num, cutoff of p value for DE analysis
assemble	'intersect' or 'union', whether to select intersected or union genes of different comparisons, default 'intersect'
Rank	character, the variable for ranking DEGs, can be 'logFC', 'adj.P.Val' ..., default 'adj.P.Val'
nperm	num, permutation runs of simulating the distribution
thres	num, cutoff for rank product permutation test if feature_selection = "rankproduct", default 0.05
keep.top	NULL or num, whether to keep top n DEGs of specific comparison
keep.group	NULL or pattern, specify the top DEGs of which comparison or group to be kept
...	omitted

### Value

A list of "UP" and "DOWN" genes

de\_analysis

*DE analysis pipeline***Description**

Standard DE analysis by using edgeR and limma::voom pipeline

**Usage**

```
de_analysis(
  dge,
  group_col,
  target_group,
  normalize = TRUE,
  group = FALSE,
  filter = c(10, 10),
  plot = FALSE,
  lfc = 0,
  p = 0.05,
  markers = NULL,
  gene_id = "SYMBOL",
  slot = "counts",
  batch = NULL,
  summary = TRUE,
  ...
)
```

**Arguments**

dge	DGEList object for DE analysis, including expr and samples info
group_col	character, column name of coldata to specify the DE comparisons
target_group	pattern, specify the group of interest, e.g. NK
normalize	logical, if the expr in data is raw counts needs to be normalized
group	logical, TRUE to separate samples into only 2 groups: 'target_group' and 'Others'; FALSE to set each level as a group
filter	a vector of 2 numbers, filter condition to remove low expression genes, the 1st for min.counts (if normalize = TRUE) or CPM/TPM (if normalize = FALSE), the 2nd for samples size 'large.n'
plot	logical, if to make plots to show QC before and after filtration
lfc	num, cutoff of logFC for DE analysis
p	num, cutoff of p value for DE analysis and permutation test if feature_selection = "rankproduct"
markers	vector, a vector of gene names, listed the gene symbols to be kept anyway after filtration. Default 'NULL' means no special genes need to be kept.

gene_id	character, specify the gene ID target_group of rownames of expression data when markers is not NULL, could be one of 'ENSEMBL', 'SYMBOL', 'ENTREZ' ..., default 'SYMBOL'
slot	character, specify which slot to use for DGEList, default 'counts'
batch	vector of character, column name(s) of coldata to be treated as batch effect factor, default NULL
summary	logical, if to show the summary of DE analysis
...	omitted

### Value

MArrayLM object generated by `limma::treat()`

### Examples

```
data("im_data_6")
dge <- edgeR::DGEList(
  counts = Biobase::exprs(im_data_6),
  samples = Biobase::pData(im_data_6)
)
de_analysis(dge, group_col = "celltype.ch1", target_group = "NK")
```

---

filter\_subset\_sig      *Filter specific cell type signature genes against other subsets.*

---

### Description

Specify the signature of the subset matched 'target\_group' against other subsets, either "union", "intersect" or "RRA" can be specified when input is a list of datasets to integrate the signatures into one.

### Usage

```
filter_subset_sig(
  data,
  group_col,
  target_group,
  markers = NULL,
  normalize = TRUE,
  dir = "UP",
  gene_id = "SYMBOL",
  feature_selection = c("auto", "rankproduct", "none"),
  comb = union,
  filter = c(10, 10),
  s_thres = 0.05,
  ...
)
```



```
)

## S4 method for signature 'list'
filter_subset_sig(
  data,
  group_col,
  target_group,
  markers = NULL,
  normalize = TRUE,
  dir = "UP",
  gene_id = "SYMBOL",
  feature_selection = c("auto", "rankproduct", "none"),
  comb = union,
  filter = c(10, 10),
  s_thres = 0.05,
  slot = "counts",
  batch = NULL,
  ...
)

## S4 method for signature 'DGEList'
filter_subset_sig(
  data,
  group_col,
  target_group,
  markers = NULL,
  normalize = TRUE,
  dir = "UP",
  gene_id = "SYMBOL",
  feature_selection = c("auto", "rankproduct", "none"),
  comb = union,
  filter = c(10, 10),
  s_thres = 0.05,
  ...
)

## S4 method for signature 'ANY'
filter_subset_sig(
  data,
  group_col,
  target_group,
  markers = NULL,
  normalize = TRUE,
  dir = "UP",
  gene_id = "SYMBOL",
  feature_selection = c("auto", "rankproduct", "none"),
  comb = union,
  filter = c(10, 10),
```

```

    s_thres = 0.05,
    ...
)

```

### Arguments

data	An expression data or a list of expression data objects
group_col	vector or character, specify the group factor or column name of coldata for DE comparisons
target_group	pattern, specify the group of interest, e.g. NK
markers	vector, a vector of gene names, listed the gene symbols to be kept anyway after filtration. Default 'NULL' means no special genes need to be kept.
normalize	logical, if the expr in data is raw counts needs to be normalized
dir	character, could be 'UP' or 'DOWN' to use only up- or down-expressed genes
gene_id	character, specify the gene ID target_group of rownames of expression data when markers is not NULL, could be one of 'ENSEMBL', 'SYMBOL', 'ENTREZ' ..., default 'SYMBOL'
feature_selection	one of "auto" (default), "rankproduct" or "none", choose if to use rank product or not to select DEGs from multiple comparisons of DE analysis, 'auto' uses 'rankproduct' but change to 'none' if final genes < 5 for both UP and DOWN
comb	'RRA' or Fun for combining sigs from multiple datasets, keep all passing genes or only intersected genes, could be union or intersect or setdiff or customized Fun, or could be 'RRA' to use Robust Rank Aggregation method for integrating multi-lists of sigs, default 'union'
filter	(list of) vector of 2 numbers, filter condition to remove low expression genes, the 1st for min.counts (if normalize = TRUE) or CPM/TPM (if normalize = FALSE), the 2nd for samples size 'large.n'
s_thres	num, threshold of score if comb = 'RRA'
...	other params for <code>get_degs()</code>
slot	character, specify which slot to use only for DGEList, sce or seurat object, optional, default 'counts'
batch	vector of character, column name(s) of coldata to be treated as batch effect factor, default NULL

### Value

a vector of gene symbols

### Examples

```

data("im_data_6", "nk_markers")
sigs <- filter_subset_sig(im_data_6, "celltype:ch1", "NK",
  markers = nk_markers$HGNC_Symbol,
  gene_id = "ENSEMBL"
)

```

---

`get_degs`*Get differentially expressed genes by comparing specified groups*

---

**Description**

This function uses edgeR and limma to get 'UP' and 'DOWN' DEG lists, for multiple comparisons, DEGs can be obtained from intersection of all DEGs or by using product of p value ranks for multiple comparisons. Filter out low expressed genes and extract DE genes by using `limma::voom` and `limma::treat`, and also create an object `proc_data` to store processed data.

**Usage**

```
get_degs(  
  data,  
  group_col,  
  target_group,  
  normalize = TRUE,  
  feature_selection = c("auto", "rankproduct", "none"),  
  slot = "counts",  
  batch = NULL,  
  ...  
)  
  
## S4 method for signature 'DGEList,character,character'  
get_degs(  
  data,  
  group_col,  
  target_group,  
  normalize = TRUE,  
  feature_selection = c("auto", "rankproduct", "none"),  
  slot = "counts",  
  batch = NULL,  
  ...  
)  
  
## S4 method for signature 'matrix,vector,character'  
get_degs(  
  data,  
  group_col,  
  target_group,  
  normalize = TRUE,  
  feature_selection = c("auto", "rankproduct", "none"),  
  slot = "counts",  
  batch = NULL,  
  ...  
)
```

```
## S4 method for signature 'Matrix,vector,character'
get_degs(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  feature_selection = c("auto", "rankproduct", "none"),
  slot = "counts",
  batch = NULL,
  ...
)

## S4 method for signature 'ExpressionSet,character,character'
get_degs(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  feature_selection = c("auto", "rankproduct", "none"),
  slot = "counts",
  batch = NULL,
  ...
)

## S4 method for signature 'SummarizedExperiment,character,character'
get_degs(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  feature_selection = c("auto", "rankproduct", "none"),
  slot = "counts",
  batch = NULL,
  ...
)

## S4 method for signature 'Seurat,character,character'
get_degs(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  feature_selection = c("auto", "rankproduct", "none"),
  slot = "counts",
  batch = NULL,
  ...
)
```

**Arguments**

data	expression object
group_col	vector or character, specify the group factor or column name of coldata for DE comparisons
target_group	pattern, specify the group of interest, e.g. NK
normalize	logical, if the expr in data is raw counts needs to be normalized
feature_selection	one of "auto" (default), "rankproduct" or "none", choose if to use rank product or not to select DEGs from multiple comparisons of DE analysis, 'auto' uses 'rankproduct' but change to 'none' if final genes < 5 for both UP and DOWN
slot	character, specify which slot to use only for DGEList, sce or seurat object, optional, default 'counts'
batch	vector of column name(s) or dataframe, specify the batch effect factor(s), default NULL
...	params for <a href="#">process_data()</a> and <a href="#">select_sig()</a>

**Value**

A list of 'UP', 'DOWN' gene set of all differentially expressed genes, and a DGEList 'proc\_data' containing data after process (filtration, normalization and voom fit). Both 'UP' and 'DOWN' are ordered by rank product or 'Rank' variable if keep.top is NULL

**Examples**

```
data("im_data_6")
DEGs <- get_degs(im_data_6,
  group_col = "celltype:ch1",
  target_group = "NK", gene_id = "ENSEMBL"
)
```

---

get\_de\_table

*Get DE analysis result table(s) with statistics*


---

**Description**

This function uses edgeR and limma to get DE analysis results lists for multiple comparisons. Filter out low expressed genes and obtain DE statistics by using limma::voom and limma::treat, and also create an object proc\_data to store processed data.

**Usage**

```

get_de_table(data, group_col, target_group, slot = "counts", ...)

## S4 method for signature 'DGEList,character,character'
get_de_table(data, group_col, target_group, slot = "counts", ...)

## S4 method for signature 'matrix,vector,character'
get_de_table(data, group_col, target_group, slot = "counts", ...)

## S4 method for signature 'Matrix,vector,character'
get_de_table(data, group_col, target_group, slot = "counts", ...)

## S4 method for signature 'ExpressionSet,character,character'
get_de_table(data, group_col, target_group, slot = "counts", ...)

## S4 method for signature 'SummarizedExperiment,character,character'
get_de_table(data, group_col, target_group, slot = "counts", ...)

## S4 method for signature 'Seurat,character,character'
get_de_table(data, group_col, target_group, slot = "counts", ...)

```

**Arguments**

data	expression object
group_col	vector or character, specify the group factor or column name of coldata for DE comparisons
target_group	pattern, specify the group of interest, e.g. NK
slot	character, specify which slot to use only for DGEList, sce or seurat object, optional, default 'counts'
...	params for function <a href="#">de_analysis()</a>

**Value**

A list of DE result table of all comparisons.

**Examples**

```

data("im_data_6")
DE_tables <- get_de_table(im_data_6, group_col = "celltype:ch1", target_group = "NK")

```

---

`get_gsc_sig`*Collect genes from MSigDB or provided GeneSetCollection.*

---

## Description

Collect gene sets from MSigDB or given GeneSetCollection, of which the gene-set names are matched to the given regex pattern by using `grep()` function. By setting `cat` and `subcat`, matching can be constrained in the union of given categories and subcategories if `gsc = 'msigdb'`.

## Usage

```
get_gsc_sig(  
  gsc = "msigdb",  
  pattern,  
  cat = NULL,  
  subcat = NULL,  
  species = c("hs", "mm"),  
  id = c("SYM", "EZID"),  
  version = msigdb::getMsigdbVersions(),  
  ...  
)  
  
## S4 method for signature 'GeneSetCollection,character'  
get_gsc_sig(  
  gsc = "msigdb",  
  pattern,  
  cat = NULL,  
  subcat = NULL,  
  species = c("hs", "mm"),  
  id = c("SYM", "EZID"),  
  version = msigdb::getMsigdbVersions(),  
  ...  
)  
  
## S4 method for signature 'character,character'  
get_gsc_sig(  
  gsc = "msigdb",  
  pattern,  
  cat = NULL,  
  subcat = NULL,  
  species = c("hs", "mm"),  
  id = c("SYM", "EZID"),  
  version = msigdb::getMsigdbVersions(),  
  ...  
)
```

**Arguments**

gsc	'msigdb' or GeneSetCollection to be searched
pattern	pattern pass to <code>grep()</code> , to match the MsigDB gene-set name of interest, e.g. 'NATURAL_KILLER_CELL_MEDIATED'
cat	character, stating the category(s) to be retrieved. The category(s) must be one from <code>msigdb::listCollections()</code> , see details in <code>msigdb::subsetCollection()</code>
subcat	character, stating the sub-category(s) to be retrieved. The sub-category(s) must be one from <code>msigdb::listSubCollections()</code> , see details in <code>msigdb::subsetCollection()</code>
species	character, species of interest, can be 'hs' or 'mm'
id	a character, representing the ID type to use ("SYM" for gene SYMBOLs and "EZID" for ENTREZ IDs)
version	a character, stating the version of MSigDB to be retrieved (should be $\geq 7.2$ ). See <code>msigdb::getMsigdbVersions()</code> .
...	params for <code>grep()</code> , used to match pattern to gene-set names

**Value**

A GeneSet object containing all matched gene-sets in MSigDB

**Examples**

```
data("msigdb_gobp_nk")
get_gsc_sig(
  gsc = msigdb_gobp_nk,
  pattern = "natural_killer_cell_mediated",
  subcat = "GO:BP",
  ignore.case = TRUE
)
```

---

```
get_lm_sig
```

---

*Extract specific subset markers from LM7 or/and LM22*

---

**Description**

Extract markers for subsets matched to the given pattern from LM7/LM22, and save the matched genes in 'GeneSet' class object, if both pattern are provided, the output would be a 'GeneSetCollection' class object with setName: LM7, LM22.

**Usage**

```
get_lm_sig(lm7.pattern, lm22.pattern, ...)
```



**Arguments**

lm7.pattern	character string containing a regular expression, to be matched in the given subsets in LM7
lm22.pattern	character string containing a regular expression, to be matched in the given subsets in LM22
...	params for function <code>grep()</code>

**Value**

A `GeneSet` or `GeneSetCollection` for matched subsets in LM7 and/or LM22

**Examples**

```
data("lm7", "lm22")
get_lm_sig(lm7.pattern = "NK", lm22.pattern = "NK cells")
```

---

get_panglao_sig	<i>Extract immune subset markers from PanglaoDB website.</i>
-----------------	--

---

**Description**

Extract specific immune subset markers for 'Hs' or 'Mm', the markers are retrieved from up-to-date PanglaoDB website.

**Usage**

```
get_panglao_sig(type, species = c("Hs", "Mm", "Mm Hs"))
```

**Arguments**

type	character vector, cell type name(s) of interest, available subsets could be listed by <code>list_panglao_types()</code>
species	character, default 'Hs', could be 'Hs', 'Mm' or 'Mm Hs', specify the species of interest

**Value**

a 'GeneSet' class object containing genes of given type(s)

**Examples**

```
get_panglao_sig(type = "NK cells")
get_panglao_sig(type = c("NK cells", "T cells"))
```

---

`gls2gsc`*Convert gene-set list into GeneSetCollection*

---

**Description**

Convert gene-set list into GeneSetCollection

**Usage**

```
gls2gsc(...)  
  
## S4 method for signature 'list'  
gls2gsc(...)  
  
## S4 method for signature 'vector'  
gls2gsc(...)
```

**Arguments**

... vector of genes or list of genes

**Value**

GeneSetCollection

**Examples**

```
data("msigdb_gobp_nk")  
gls2gsc(GSEABase::geneIds(msigdb_gobp_nk[1:3]))
```

---

`gsc_plot`*Make upset plot for given gene sets*

---

**Description**

Plot upset diagram for overlapping genes among given gene-sets.

**Usage**

```
gsc_plot(...)
```

**Arguments**

... GeneSet or GeneSetCollection

**Value**

upset plot object

**Examples**

```
data("msigdb_gobp_nk")
gsc_plot(msigdb_gobp_nk[1:3])
```

---

im\_data\_6

*RNA-seq TMM normalized counts data of 6 sorted immune subsets.*

---

**Description**

An ExpressionSet objects containing 6 immune subsets (B-cells, CD4, CD8, Monocytes, Neutrophils, NK) from healthy individuals.

**Usage**

```
data(im_data_6)
```

**Format**

An ExpressionSet objects of 6\*4 samples.

**Value**

ExpressionSet

**Source**

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE60424>

---

list\_panglao\_organisms

*Show the summary info of available organs in PanglaoDB.*

---

**Description**

Show the name of organs available in PanglaoDB. Help users know which organs could be retrieved by PanglaoDB.

**Usage**

```
list_panglao_organisms()
```

**Value**

a vector of available organ types or cell types in PanglaoDB

**Examples**

```
list_panglao_organs()
```

---

```
list_panglao_types
```

*Show the summary info of available cell types in PanglaoDB.*

---

**Description**

Show the name and number of each cell type in PanglaoDB. Help users know which subset(s) marker list(s) could be retrieved by PanglaoDB.

**Usage**

```
list_panglao_types(organ)
```

**Arguments**

organ                    character, specify the tissue or organ label to list cell types

**Value**

a vector of available cell types of the organ in PanglaoDB

**Examples**

```
list_panglao_types(organ = "Immune system")
```

---

```
lm22
```

*LM22 matrix for CIBERSORT.*

---

**Description**

A dataset containing 547 marker genes expression of 22 immune subsets which is generated for CIBERSORT.

**Usage**

```
data(lm22)
```

**Format**

A data frame with 547 rows 23 variables:

**Gene** gene symbols

**B cells naive** 0 or 1, represents if the gene is significantly up-regulated in the subset

**B cells memory** 0 or 1

**Plasma cells** 0 or 1

**T cells CD8** 0 or 1

**T cells CD4 naive** 0 or 1

**T cells CD4 memory resting** 0 or 1

**T cells CD4 memory activated** 0 or 1

**T cells follicular helper** 0 or 1

**T cells regulatory (Tregs)** 0 or 1

**T cells gamma delta** 0 or 1

**NK cells resting** 0 or 1

**NK cells activated** 0 or 1

**Monocytes** 0 or 1

**Macrophages M0** 0 or 1

**Macrophages M1** 0 or 1

**Macrophages M2** 0 or 1

**Dendritic cells resting** 0 or 1

**Dendritic cells activated** 0 or 1

**Mast cells resting** 0 or 1

**Mast cells activated** 0 or 1

**Eosinophils** 0 or 1

**Neutrophils** 0 or 1

**Value**

data frame

**Source**

<https://cibersort.stanford.edu/>

---

`lm7`*LM7 matrix for CIBERSORT.*

---

**Description**

A dataset containing 375 marker genes expression of 7 immune subsets which is generated for CIBERSORT.

**Usage**

```
data(lm7)
```

**Format**

A data frame with 375 rows 9 variables:

**Gene** gene symbols

**Subset** immune subset of the marker gene

**B cells** gene median expression in B cells

**T CD4** gene median expression in T CD4 cells

**T CD8** gene median expression in T CD8 cells

**T gamma delta** gene median expression in T gamma delta cells

**NK** gene median expression in NK cells

**MoMaDC** gene median expression in MoMaDC cells

**granulocytes** gene median expression in granulocytes

**Value**

data frame

**Source**

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5384348/>

---

 mastR\_Package

*Screen Immune Cells Signature for Specific Cancer or Tissue Type*


---

**Description**

mastR This package enables automated screening of group specific signature for specific tissues. The package is developed for generating refined lists of signature genes from multiple group comparisons based on the results from edgeR and limma differential expression (DE) analysis workflow. It also takes into account the background expression of tissue-specificity, which is often ignored by other markers generation tools. This package also provides pseudo bulking function to deal with scRNA-seq data. Multiple visualization functions are implemented in this package.

**Value**

Automated screened signature

**Author(s)**

Jinjin Chen <chen.j@wehi.edu.au>

---

 merge\_markers

*Merge markers list into one.*


---

**Description**

Merge markers collected from different DB into one 'GeneSet' object, saved a data.frame in json format under longDescription with 'TRUE' and '-' to indicate which DB each gene is from, this can be shown via `jsonlite::fromJSON()`.

**Usage**

```
merge_markers(...)
```

**Arguments**

... GeneSet or GeneSetCollection object to be merged

**Value**

A GeneSet class of union genes in the given list

**Examples**

```
data("msigdb_gobp_nk")
Markers <- merge_markers(msigdb_gobp_nk[1:3])
jsonlite::fromJSON(GSEABase::longDescription(Markers))
```

---

msigdb_gobp_nk	<i>Sub-collection of MSigDB gene sets.</i>
----------------	--

---

**Description**

A small GeneSetCollection object, contains gene sets with gene set name matched to 'NATURAL\_KILLER' from GO:BP MSigDB v7.4 database.

**Usage**

```
data(msigdb_gobp_nk)
```

**Format**

A GeneSetCollection of 55 gene sets.

**Value**

GeneSetCollection

**Source**

```
msigdb::getMsigdb()
```

---

nk_markers	<i>NK cell markers combination.</i>
------------	-------------------------------------

---

**Description**

A dataset containing 114 NK cell markers from LM22, LM7 and human orthologs in mice.

**Usage**

```
data(nk_markers)
```

**Format**

A data frame with 114 rows and at least 4 variables:

**HGNC\_Symbol** gene symbols

**LM22** if included in LM22

**LM7** if included in LM7

**Huntington** if included in orthologs



**Value**

data frame

**Source**

<https://cancerimmunolres.aacrjournals.org/content/7/7/1162.long>

---

pca\_matrix\_plot

*Make a matrix plot of PCA with top PCs*

---

**Description**

Make a matrix plot of PCA with top PCs

**Usage**

```
pca_matrix_plot(  
  data,  
  features = "all",  
  slot = "counts",  
  group_by = NULL,  
  scale = TRUE,  
  n = 4,  
  loading = FALSE,  
  n_loadings = 10,  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'matrix'  
pca_matrix_plot(  
  data,  
  features = "all",  
  group_by = NULL,  
  scale = TRUE,  
  n = 4,  
  loading = FALSE,  
  n_loadings = 10,  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'Matrix'  
pca_matrix_plot(  
  data,  
  features = "all",  
  group_by = NULL,  
  scale = TRUE,  
  n = 4,
```

```
    loading = FALSE,
    n_loadings = 10,
    gene_id = "SYMBOL"
)

## S4 method for signature 'data.frame'
pca_matrix_plot(
  data,
  features = "all",
  group_by = NULL,
  scale = TRUE,
  n = 4,
  loading = FALSE,
  n_loadings = 10,
  gene_id = "SYMBOL"
)

## S4 method for signature 'ExpressionSet'
pca_matrix_plot(
  data,
  features = "all",
  group_by = NULL,
  scale = TRUE,
  n = 4,
  loading = FALSE,
  n_loadings = 10,
  gene_id = "SYMBOL"
)

## S4 method for signature 'DGEList'
pca_matrix_plot(
  data,
  features = "all",
  slot = "counts",
  group_by = NULL,
  scale = TRUE,
  n = 4,
  loading = FALSE,
  n_loadings = 10,
  gene_id = "SYMBOL"
)

## S4 method for signature 'SummarizedExperiment'
pca_matrix_plot(
  data,
  features = "all",
  slot = "counts",
  group_by = NULL,
```

```

    scale = TRUE,
    n = 4,
    loading = FALSE,
    n_loadings = 10,
    gene_id = "SYMBOL"
  )

  ## S4 method for signature 'Seurat'
  pca_matrix_plot(
    data,
    features = "all",
    slot = "counts",
    group_by = NULL,
    scale = TRUE,
    n = 4,
    loading = FALSE,
    n_loadings = 10,
    gene_id = "SYMBOL"
  )

```

### Arguments

data	expression data, can be matrix, eSet, seurat...
features	vector of gene symbols or 'all', specify the genes used for PCA, default 'all'
slot	character, specify the slot name of expression to be used, optional
group_by	character, specify the column to be grouped and colored, default NULL
scale	logical, if to scale data for PCA, default TRUE
n	num, specify top n PCs to plot
loading	logical, if to plot and label loadings of PCA, default 'FALSE'
n_loadings	num, top n loadings to plot; or a vector of gene IDs; only work when loading = TRUE
gene_id	character, specify which column of IDs used to calculate TPM, also indicate the ID type of expression data's rowname, could be one of 'ENSEMBL', 'SYMBOL', 'ENTREZ' ..., default 'SYMBOL'

### Value

matrix plot of PCA

### Examples

```

data("im_data_6")
pca_matrix_plot(data = im_data_6, scale = FALSE)

```

---

pca\_matrix\_plot\_init *Make a matrix plot of PCA with top PCs*

---

### Description

Make a matrix plot of PCA with top PCs

### Usage

```
pca_matrix_plot_init(  
  data,  
  features = "all",  
  group_by = NULL,  
  scale = TRUE,  
  n = 4,  
  loading = FALSE,  
  n_loadings = 10,  
  gene_id = "SYMBOL"  
)
```

### Arguments

data	expression matrix
features	vector of gene symbols or 'all', specify the genes used for PCA, default 'all'
group_by	character, specify the column to be grouped and colored, default NULL
scale	logical, if to scale data for PCA, default TRUE
n	num, specify top n PCs to plot
loading	logical, if to plot and label loadings of PCA, default 'FALSE'
n_loadings	num, top n loadings to plot; or a vector of gene IDs; only work when loading = TRUE
gene_id	character, specify which column of IDs used to calculate TPM, also indicate the ID type of expression data's rowname, could be one of 'ENSEMBL', 'SYMBOL', 'ENTREZ' ..., default 'SYMBOL'

### Value

matrix plot of PCA

---

plotPCAbiplot      *Single PCA plot function*

---

### Description

Single PCA plot function

### Usage

```
plotPCAbiplot(
  prcomp,
  loading = FALSE,
  n_loadings = 10,
  dims = c(1, 2),
  group_by = NULL
)
```

### Arguments

prcomp	prcomp object generated by <code>stats::prcomp()</code>
loading	logical, if to plot and label loadings of PCA, default 'FALSE'
n_loadings	num, top n loadings to plot; or a vector of gene IDs; only work when loading = TRUE
dims	a vector of 2 elements, specifying PCs to plot
group_by	character, specify the column to be grouped and colored, default NULL

### Value

ggplot of PCA

---

plot\_diagnostics      *plot diagnostics before and after `process_data()`*

---

### Description

plot diagnostics before and after `process_data()`

### Usage

```
plot_diagnostics(expr1, expr2, group_col, abl = 2)
```

**Arguments**

expr1	expression matrix 1 for original data
expr2	expression matrix 2 for processed data
group_col	vector of group of samples
abl	num, cutoff line

**Value**

multiple plots

**Examples**

```
data("im_data_6")
dge <- edgeR::DGEList(
  counts = Biobase::exprs(im_data_6),
  samples = Biobase::pData(im_data_6)
)
dge$logCPM <- edgeR::cpm(dge, log = TRUE)
proc_data <- process_data(dge,
  group_col = "celltype.ch1",
  target_group = "NK"
)
plot_diagnostics(proc_data$logCPM, proc_data$svfit$E,
  group_col = proc_data$samples$group
)
```

---

plot_mean_var	<i>plot Mean-variance trend after voom and after final linear fit</i>
---------------	---

---

**Description**

plot Mean-variance trend after voom and after final linear fit

**Usage**

```
plot_mean_var(proc_data, span = 0.5)
```

**Arguments**

proc_data	processed data returned by <code>process_data()</code>
span	num, span for <code>lowess()</code>

**Value**

comparison plot of mean-variance of voom and final model

**Examples**

```
data("im_data_6")
proc_data <- process_data(
  im_data_6,
  group_col = "celltype:ch1",
  target_group = "NK"
)
plot_mean_var(proc_data)
```

---

process_data	<i>process data</i>
--------------	---------------------

---

**Description**

filter low expression genes, normalize data by 'TMM' and apply `limma::voom()`, `limma::lmFit()` and `limma::treat()` on normalized data

**Usage**

```
process_data(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  filter = c(10, 10),
  lfc = 0,
  p = 0.05,
  markers = NULL,
  gene_id = "SYMBOL",
  slot = "counts",
  ...
)

## S4 method for signature 'DGEList,character,character'
process_data(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  filter = c(10, 10),
  lfc = 0,
  p = 0.05,
  markers = NULL,
  gene_id = "SYMBOL",
  slot = "counts",
  ...
)
```

```
## S4 method for signature 'matrix,vector,character'
process_data(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  filter = c(10, 10),
  lfc = 0,
  p = 0.05,
  markers = NULL,
  gene_id = "SYMBOL",
  batch = NULL,
  ...
)

## S4 method for signature 'Matrix,vector,character'
process_data(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  filter = c(10, 10),
  lfc = 0,
  p = 0.05,
  markers = NULL,
  gene_id = "SYMBOL",
  batch = NULL,
  ...
)

## S4 method for signature 'ExpressionSet,character,character'
process_data(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  filter = c(10, 10),
  lfc = 0,
  p = 0.05,
  markers = NULL,
  gene_id = "SYMBOL",
  batch = NULL,
  ...
)

## S4 method for signature 'SummarizedExperiment,character,character'
process_data(
```



```

    data,
    group_col,
    target_group,
    normalize = TRUE,
    filter = c(10, 10),
    lfc = 0,
    p = 0.05,
    markers = NULL,
    gene_id = "SYMBOL",
    slot = "counts",
    batch = NULL,
    ...
)

## S4 method for signature 'Seurat,character,character'
process_data(
  data,
  group_col,
  target_group,
  normalize = TRUE,
  filter = c(10, 10),
  lfc = 0,
  p = 0.05,
  markers = NULL,
  gene_id = "SYMBOL",
  slot = "counts",
  batch = NULL,
  ...
)

```

### Arguments

data	expression object
group_col	character, column name of coldata to specify the DE comparisons
target_group	pattern, specify the group of interest, e.g. NK
normalize	logical, if the expr in data is raw counts needs to be normalized
filter	a vector of 2 numbers, filter condition to remove low expression genes, the 1st for min.counts (if normalize = TRUE) or CPM/TPM (if normalize = FALSE), the 2nd for samples size 'large.n'
lfc	num, cutoff of logFC for DE analysis
p	num, cutoff of p value for DE analysis and permutation test if feature_selection = "rankproduct"
markers	vector, a vector of gene names, listed the gene symbols to be kept anyway after filtration. Default 'NULL' means no special genes need to be kept.
gene_id	character, specify the gene ID target_group of rownames of expression data when markers is not NULL, could be one of 'ENSEMBL', 'SYMBOL', 'ENTREZ' ..., default 'SYMBOL'

slot	character, specify which slot to use only for DGEList, sce or seurat object, optional, default 'counts'
...	params for <code>voom_fit_treat()</code>
batch	vector of character, column name(s) of coldata to be treated as batch effect factor, default NULL

**Value**

A DGEList containing vfit by `limma::voom()` (if `normalize = TRUE`) and tfit by `limma::treat()`

**Examples**

```
data("im_data_6")
proc_data <- process_data(
  im_data_6,
  group_col = "celltype:ch1",
  target_group = "NK"
)
```

---

pseudo_samples	<i>Aggregate single cells to pseudo-samples according to specific factors</i>
----------------	---

---

**Description**

Gather cells for each group according to specified factors, then randomly assign and aggregate cells to each pseudo-samples with randomized cell size. (`min.cells <= size <= max.cells`)

**Usage**

```
pseudo_samples(
  data,
  by,
  fun = c("sum", "mean"),
  scale = NULL,
  min.cells = 0,
  max.cells = Inf,
  slot = "counts"
)

## S4 method for signature 'matrix,data.frame'
pseudo_samples(
  data,
  by,
  fun = c("sum", "mean"),
  scale = NULL,
  min.cells = 0,
  max.cells = Inf,
```

```

    slot = "counts"
  )

## S4 method for signature 'matrix,vector'
pseudo_samples(
  data,
  by,
  fun = c("sum", "mean"),
  scale = NULL,
  min.cells = 0,
  max.cells = Inf,
  slot = "counts"
)

## S4 method for signature 'Seurat,character'
pseudo_samples(
  data,
  by,
  fun = c("sum", "mean"),
  scale = NULL,
  min.cells = 0,
  max.cells = Inf,
  slot = "counts"
)

## S4 method for signature 'SummarizedExperiment,character'
pseudo_samples(
  data,
  by,
  fun = c("sum", "mean"),
  scale = NULL,
  min.cells = 0,
  max.cells = Inf,
  slot = "counts"
)

```

### Arguments

data	a matrix or Seurat/SCE object containing expression and metadata
by	a vector of group names or dataframe for aggregation
fun	chr, methods used to aggregate cells, could be 'sum' or 'mean', default 'sum'
scale	a num or NULL, if to multiply a scale to the average expression
min.cells	num, default 300, the minimum size of cells aggregating to each pseudo-sample
max.cells	num, default 600, the maximum size of cells aggregating to each pseudo-sample
slot	chr, specify which slot of seurat object to aggregate, can be 'counts', 'data', 'scale.data'..., default is 'counts'

**Value**

An expression matrix after aggregating cells on specified factors

**Examples**

```
counts <- matrix(abs(rnorm(10000, 10, 10)), 100)
rownames(counts) <- 1:100
colnames(counts) <- 1:100
meta <- data.frame(
  subset = rep(c("A", "B"), 50),
  level = rep(1:4, each = 25)
)
rownames(meta) <- 1:100
scRNA <- SeuratObject::CreateSeuratObject(counts = counts, meta.data = meta)
pseudo_samples(scRNA,
  by = c("subset", "level"),
  min.cells = 10, max.cells = 20
)
```

---

pseudo\_sample\_list      *Split cells according to specific factors*

---

**Description**

Gathering cells to make the pool according to specific factors, and randomly assign the cells from the pool to pseudo-sample with the randomized cell size. (min.cells <= size <= max.cells)

**Usage**

```
pseudo_sample_list(data, by, min.cells = 0, max.cells = Inf)
```

**Arguments**

data	matrix or data.frame or other single cell expression object
by	a vector or data.frame contains factor(s) for aggregation
min.cells	num, default 0, the minimum size of cells aggregating to each pseudo-sample
max.cells	num, default Inf, the maximum size of cells aggregating to each pseudo-sample

**Value**

A list of cell names for each pseudo-sample

**Examples**

```

counts <- matrix(abs(rnorm(10000, 10, 10)), 100)
rownames(counts) <- 1:100
colnames(counts) <- 1:100
meta <- data.frame(
  subset = rep(c("A", "B"), 50),
  level = rep(1:4, each = 25)
)
rownames(meta) <- 1:100
scRNA <- SeuratObject::CreateSeuratObject(counts = counts, meta.data = meta)
pseudo_sample_list(scRNA,
  by = c("subset", "level"),
  min.cells = 10, max.cells = 20
)

```

---

remove\_bg\_exp

*Remove markers with high signal in background data.*


---

**Description**

Specify signatures against specific tissues or cell lines by removing genes with high expression in the background.

**Usage**

```

remove_bg_exp(
  sig_data,
  bg_data = "CCLE",
  markers,
  s_group_col = NULL,
  s_target_group = NULL,
  b_group_col = NULL,
  b_target_group = NULL,
  snr = 1,
  ...,
  filter = NULL,
  gene_id = "SYMBOL",
  s_slot = "counts",
  b_slot = "counts",
  ccle_tpm = NULL,
  ccle_meta = NULL
)

## S4 method for signature 'matrix,matrix,vector'
remove_bg_exp(
  sig_data,
  bg_data = "CCLE",

```

```
markers,
s_group_col = NULL,
s_target_group = NULL,
b_group_col = NULL,
b_target_group = NULL,
snr = 1,
...,
filter = NULL,
gene_id = "SYMBOL",
s_slot = "counts",
b_slot = "counts",
ccle_tpm = NULL,
ccle_meta = NULL
)

## S4 method for signature 'DGEList,matrix,vector'
remove_bg_exp(
  sig_data,
  bg_data = "CCLE",
  markers,
  s_group_col = NULL,
  s_target_group = NULL,
  b_group_col = NULL,
  b_target_group = NULL,
  snr = 1,
  ...,
  filter = NULL,
  gene_id = "SYMBOL",
  s_slot = "counts",
  b_slot = "counts",
  ccle_tpm = NULL,
  ccle_meta = NULL
)

## S4 method for signature 'ANY,DGEList,vector'
remove_bg_exp(
  sig_data,
  bg_data = "CCLE",
  markers,
  s_group_col = NULL,
  s_target_group = NULL,
  b_group_col = NULL,
  b_target_group = NULL,
  snr = 1,
  ...,
  filter = NULL,
  gene_id = "SYMBOL",
  s_slot = "counts",
```

```
    b_slot = "counts",
    ccle_tpm = NULL,
    ccle_meta = NULL
  )

## S4 method for signature 'ANY,ExpressionSet,vector'
remove_bg_exp(
  sig_data,
  bg_data = "CCLE",
  markers,
  s_group_col = NULL,
  s_target_group = NULL,
  b_group_col = NULL,
  b_target_group = NULL,
  snr = 1,
  ...,
  filter = NULL,
  gene_id = "SYMBOL",
  s_slot = "counts",
  b_slot = "counts",
  ccle_tpm = NULL,
  ccle_meta = NULL
)

## S4 method for signature 'ANY,SummarizedExperiment,vector'
remove_bg_exp(
  sig_data,
  bg_data = "CCLE",
  markers,
  s_group_col = NULL,
  s_target_group = NULL,
  b_group_col = NULL,
  b_target_group = NULL,
  snr = 1,
  ...,
  filter = NULL,
  gene_id = "SYMBOL",
  s_slot = "counts",
  b_slot = "counts",
  ccle_tpm = NULL,
  ccle_meta = NULL
)

## S4 method for signature 'ANY,Seurat,vector'
remove_bg_exp(
  sig_data,
  bg_data = "CCLE",
  markers,
```

```
s_group_col = NULL,  
s_target_group = NULL,  
b_group_col = NULL,  
b_target_group = NULL,  
snr = 1,  
...,  
filter = NULL,  
gene_id = "SYMBOL",  
s_slot = "counts",  
b_slot = "counts",  
ccle_tpm = NULL,  
ccle_meta = NULL  
)  
  
## S4 method for signature 'ANY,character,vector'  
remove_bg_exp(  
  sig_data,  
  bg_data = "CCLE",  
  markers,  
  s_group_col = NULL,  
  s_target_group = NULL,  
  b_group_col = NULL,  
  b_target_group = NULL,  
  snr = 1,  
  ...,  
  filter = NULL,  
  gene_id = "SYMBOL",  
  s_slot = "counts",  
  b_slot = "counts",  
  ccle_tpm = NULL,  
  ccle_meta = NULL  
)  
  
## S4 method for signature 'ANY,missing,vector'  
remove_bg_exp(  
  sig_data,  
  bg_data = "CCLE",  
  markers,  
  s_group_col = NULL,  
  s_target_group = NULL,  
  b_group_col = NULL,  
  b_target_group = NULL,  
  snr = 1,  
  ...,  
  filter = NULL,  
  gene_id = "SYMBOL",  
  s_slot = "counts",  
  b_slot = "counts",
```



```

    ccle_tpm = NULL,
    ccle_meta = NULL
)

## S4 method for signature 'ANY,ANY,vector'
remove_bg_exp(
  sig_data,
  bg_data = "CCLE",
  markers,
  s_group_col = NULL,
  s_target_group = NULL,
  b_group_col = NULL,
  b_target_group = NULL,
  snr = 1,
  ...,
  filter = NULL,
  gene_id = "SYMBOL",
  s_slot = "counts",
  b_slot = "counts",
  ccle_tpm = NULL,
  ccle_meta = NULL
)

```

### Arguments

sig_data	log-transformed expression object, can be matrix or DGEList, as signal data
bg_data	'CCLE' or log-transformed expression object as background data
markers	vector, a vector of gene names, listed the gene symbols to be filtered. Must be gene SYMBOLs
s_group_col	vector or character, to specify the group of signal target_groups, or column name of group, default NULL
s_target_group	pattern, specify the target group of interest in sig_data, default NULL
b_group_col	vector or character, to specify the group of background target_groups, or column name of <code>depmap::depmap_metadata()</code> , e.g. 'primary_disease', default NULL
b_target_group	pattern, specify the target_group of interest in bg_data, e.g. 'colorectal', default NULL
snr	num, the cutoff of SNR to screen markers which are not or lowly expressed in bg_data
...	params for <code>grep()</code> to find matched cell lines in bg_data
filter	NULL or a vector of 2 num, filter condition to remove low expression genes in bg_data, the 1st for logcounts, the 2nd for samples size
gene_id	character, specify the gene ID type of rownames of expression data, could be one of 'ENSEMBL', 'SYMBOL', 'ENTREZ'..., default 'SYMBOL'
s_slot	character, specify which slot to use of DGEList, sce or seurat object for sig_data, optional, default 'counts'

b_slot	character, specify which slot to use of DGEList, sce or seurat object for bg_data, optional, default 'counts'
ccle_tpm	ccle_tpm data from <code>depmap::depmap_TPM()</code> , only used when data = 'CCLE', default NULL
ccle_meta	ccle_meta data from <code>depmap::depmap_metadata()</code> , only used when data = 'CCLE', default NULL

### Value

a vector of genes after filtration

### Examples

```
data("im_data_6", "nk_markers", "ccle_crc_5")
remove_bg_exp(
  sig_data = Biobase::exprs(im_data_6),
  bg_data = ccle_crc_5,
  im_data_6$`celltype:ch1`, "NK", ## for sig_data
  "cancer", "CRC", ## for bg_data
  markers = nk_markers$HGNC_Symbol[40:50],
  filter = c(1, 2),
  gene_id = c("ENSEMBL", "SYMBOL")
)
```

---

remove_bg_exp_mat	<i>Remove genes show high signal in the background expression data from markers.</i>
-------------------	--

---

### Description

Remove genes show high signal in the background expression data from markers.

### Usage

```
remove_bg_exp_mat(sig_mat, bg_mat, markers, snr = 1, gene_id = "SYMBOL")
```

### Arguments

sig_mat	log-transformed expression matrix of interested signal data
bg_mat	log-transformed expression matrix of interested background data
markers	vector, a vector of gene names, listed the gene symbols to be filtered. Must be gene SYMBOLs.
snr	num, the cutoff of SNR to screen markers which are not or lowly expressed in bg_data
gene_id	character, specify the gene ID types of row names of sig_mat and bg_mat data, could be one of 'ENSEMBL', 'SYMBOL', 'ENTREZ'..., default 'SYMBOL'

**Value**

a vector of genes after filtration

**Examples**

```
data("im_data_6", "nk_markers", "cclle_crc_5")
remove_bg_exp_mat(
  sig_mat = Biobase::exprs(im_data_6),
  bg_mat = cclle_crc_5$counts,
  markers = nk_markers$HGNC_Symbol[30:40],
  gene_id = c("ENSEMBL", "SYMBOL")
)
```

---

select_sig	<i>select DEGs from multiple comparisons</i>
------------	--

---

**Description**

select DEGs from multiple comparisons

**Usage**

```
select_sig(tfit, feature_selection = c("auto", "rankproduct", "none"), ...)
```

**Arguments**

tfit	processed tfit by <code>limma::treat()</code> or processed data returned by <code>process_data()</code>
feature_selection	one of "auto" (default), "rankproduct" or "none", choose if to use rank product or not to select DEGs from multiple comparisons of DE analysis, 'auto' uses 'rankproduct' but change to 'none' if final genes < 5 for both UP and DOWN
...	params for <code>DEGs_RP()</code> or <code>DEGs_Group()</code>

**Value**

GeneSetCollection contains UP and DOWN gene sets

**Examples**

```
data("im_data_6")
proc_data <- process_data(
  im_data_6,
  group_col = "celltype:ch1",
  target_group = "NK"
)
select_sig(proc_data$tfit)
```

---

`sig_boxplot`*Boxplot of median expression or scores of signature*

---

**Description**

Make boxplot and show expression or score level of signature across subsets.

**Usage**

```
sig_boxplot(  
  data,  
  sigs,  
  group_col,  
  target_group,  
  type = c("score", "expression"),  
  method = "t.test",  
  slot = "counts",  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'matrix,vector,vector,character'  
sig_boxplot(  
  data,  
  sigs,  
  group_col,  
  target_group,  
  type = c("score", "expression"),  
  method = "t.test",  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'Matrix,vector,vector,character'  
sig_boxplot(  
  data,  
  sigs,  
  group_col,  
  target_group,  
  type = c("score", "expression"),  
  method = "t.test",  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'data.frame,vector,vector,character'  
sig_boxplot(  
  data,  
  sigs,  
  group_col,
```

```
target_group,
type = c("score", "expression"),
method = "t.test",
gene_id = "SYMBOL"
)

## S4 method for signature 'DGEList,vector,character,character'
sig_boxplot(
  data,
  sigs,
  group_col,
  target_group,
  type = c("score", "expression"),
  method = "t.test",
  slot = "counts",
  gene_id = "SYMBOL"
)

## S4 method for signature 'ExpressionSet,vector,character,character'
sig_boxplot(
  data,
  sigs,
  group_col,
  target_group,
  type = c("score", "expression"),
  method = "t.test",
  gene_id = "SYMBOL"
)

## S4 method for signature 'Seurat,vector,character,character'
sig_boxplot(
  data,
  sigs,
  group_col,
  target_group,
  type = c("score", "expression"),
  method = "t.test",
  slot = "counts",
  gene_id = "SYMBOL"
)

## S4 method for signature 'SummarizedExperiment,vector,character,character'
sig_boxplot(
  data,
  sigs,
  group_col,
  target_group,
  type = c("score", "expression"),
```

```

    method = "t.test",
    slot = "counts",
    gene_id = "SYMBOL"
  )

  ## S4 method for signature 'list,vector,character,character'
  sig_boxplot(
    data,
    sigs,
    group_col,
    target_group,
    type = c("score", "expression"),
    method = "t.test",
    slot = "counts",
    gene_id = "SYMBOL"
  )

```

### Arguments

data	expression data, can be matrix, DGEList, eSet, seurat, sce...
sigs	a vector of signature (Symbols)
group_col	character or vector, specify the column name to compare in coldata
target_group	pattern, specify the group of interest as reference
type	one of "score" and "expression", to plot score or expression of the signature
method	a character string indicating which method to be used for <code>stat_compare_means()</code> to compare the means across groups, could be "t.test", 'wilcox.test', 'anova' ..., default "t.test"
slot	character, indicate which slot used as expression, optional
gene_id	character, indicate the ID type of rowname of expression data's , could be one of 'ENSEMBL', 'SYMBOL', ... default 'SYMBOL'

### Value

patchwork or ggplot of boxplot

### Examples

```

data("im_data_6", "nk_markers")
p <- sig_boxplot(
  im_data_6,
  sigs = nk_markers$HGNC_Symbol[1:30],
  group_col = "celltype:ch1", target_group = "NK",
  gene_id = "ENSEMBL"
)

```

---

sig_gseaplot	<i>Visualize GSEA result with input list of gene symbols.</i>
--------------	---

---

## Description

Visualize GSEA result with multiple lists of genes by using clusterProfiler.

## Usage

```
sig_gseaplot(  
  data,  
  sigs,  
  group_col,  
  target_group,  
  gene_id = "SYMBOL",  
  slot = "counts",  
  method = c("dotplot", "gseaplot"),  
  col = "-log10(p.adjust)",  
  size = "enrichmentScore",  
  pvalue_table = FALSE,  
  digits = 2,  
  rank_stat = "logFC",  
  ...  
)  
  
## S4 method for signature 'MArrayLM,vector'  
sig_gseaplot(  
  data,  
  sigs,  
  group_col,  
  target_group,  
  gene_id = "SYMBOL",  
  slot = "counts",  
  method = c("dotplot", "gseaplot"),  
  col = "-log10(p.adjust)",  
  size = "enrichmentScore",  
  pvalue_table = FALSE,  
  digits = 2,  
  rank_stat = "logFC",  
  ...  
)  
  
## S4 method for signature 'MArrayLM,list'  
sig_gseaplot(  
  data,  
  sigs,  
  group_col,
```

```
target_group,
gene_id = "SYMBOL",
slot = "counts",
method = c("dotplot", "gseaplot"),
col = "-log10(p.adjust)",
size = "enrichmentScore",
pvalue_table = FALSE,
digits = 2,
rank_stat = "logFC",
...
)

## S4 method for signature 'DGEList,ANY'
sig_gseaplot(
  data,
  sigs,
  group_col,
  target_group,
  gene_id = "SYMBOL",
  slot = "counts",
  method = c("dotplot", "gseaplot"),
  col = "-log10(p.adjust)",
  size = "enrichmentScore",
  pvalue_table = FALSE,
  digits = 2,
  rank_stat = "logFC",
  ...
)

## S4 method for signature 'ANY,ANY'
sig_gseaplot(
  data,
  sigs,
  group_col,
  target_group,
  gene_id = "SYMBOL",
  slot = "counts",
  method = c("dotplot", "gseaplot"),
  col = "-log10(p.adjust)",
  size = "enrichmentScore",
  pvalue_table = FALSE,
  digits = 2,
  rank_stat = "logFC",
  ...
)

## S4 method for signature 'list,ANY'
sig_gseaplot(
```



```

    data,
    sigs,
    group_col,
    target_group,
    gene_id = "SYMBOL",
    slot = "counts",
    method = c("dotplot", "gseaplot"),
    col = "-log10(p.adjust)",
    size = "enrichmentScore",
    pvalue_table = FALSE,
    digits = 2,
    rank_stat = "logFC",
    ...
)

```

### Arguments

data	expression data, can be matrix, DGEList, eSet, seurat, sce...
sigs	a vector of signature (Symbols) or a list of signatures
group_col	character or vector, specify the column name to compare in coldata
target_group	pattern, specify the group of interest as reference
gene_id	character, indicate the ID type of rowname of expression data's , could be one of 'ENSEMBL', 'SYMBOL', ... default 'SYMBOL'
slot	character, indicate which slot used as expression, optional
method	one of "gseaplot" and "dotplot", how to plot GSEA result
col	column name of <code>clusterProfiler::GSEA()</code> result, used for dot col when method = "dotplot"
size	column name of <code>clusterProfiler::GSEA()</code> result, used for dot size when method = "dotplot"
pvalue_table	logical, if to add p value table if method = "gseaplot"
digits	num, specify the number of significant digits of pvalue table
rank_stat	character, specify which metric used to rank for GSEA, default "logFC"
...	params for function <code>get_de_table()</code> and function <code>enrichplot::gseaplot2()</code>

### Value

patchwork object for all comparisons

### Examples

```

data("im_data_6", "nk_markers")
sig_gseaplot(
  sigs = list(
    A = nk_markers$HGNC_Symbol[1:15],
    B = nk_markers$HGNC_Symbol[20:40],
    C = nk_markers$HGNC_Symbol[60:75]
  )
)

```

```

),
data = im_data_6, group_col = "celltype:ch1",
target_group = "NK", gene_id = "ENSEMBL"
)

```

---

sig\_heatmap

*Heatmap original markers and screened signature*


---

## Description

Compare the heatmap before and after screening.

## Usage

```

sig_heatmap(
  data,
  sigs,
  group_col,
  markers,
  scale = c("none", "row", "column"),
  gene_id = "SYMBOL",
  ranks_plot = FALSE,
  slot = "counts",
  ...
)

## S4 method for signature 'matrix,character,vector,missing'
sig_heatmap(
  data,
  sigs,
  group_col,
  markers,
  scale = c("none", "row", "column"),
  gene_id = "SYMBOL",
  ranks_plot = FALSE,
  slot = "counts",
  ...
)

## S4 method for signature 'matrix,character,vector,vector'
sig_heatmap(
  data,
  sigs,
  group_col,
  markers,
  scale = c("none", "row", "column"),

```

```
    gene_id = "SYMBOL",
    ranks_plot = FALSE,
    slot = "counts",
    ...
)

## S4 method for signature 'matrix,list,vector,missing'
sig_heatmap(
  data,
  sigs,
  group_col,
  markers,
  scale = c("none", "row", "column"),
  gene_id = "SYMBOL",
  ranks_plot = FALSE,
  slot = "counts",
  ...
)

## S4 method for signature 'Matrix,ANY,vector,ANY'
sig_heatmap(
  data,
  sigs,
  group_col,
  markers,
  scale = c("none", "row", "column"),
  gene_id = "SYMBOL",
  ranks_plot = FALSE,
  slot = "counts",
  ...
)

## S4 method for signature 'data.frame,ANY,vector,ANY'
sig_heatmap(
  data,
  sigs,
  group_col,
  markers,
  scale = c("none", "row", "column"),
  gene_id = "SYMBOL",
  ranks_plot = FALSE,
  slot = "counts",
  ...
)

## S4 method for signature 'DGEList,ANY,character,ANY'
sig_heatmap(
  data,
```

```
sigs,  
group_col,  
markers,  
scale = "none",  
gene_id = "SYMBOL",  
ranks_plot = FALSE,  
slot = "counts",  
...  
)  
  
## S4 method for signature 'ExpressionSet,ANY,character,ANY'  
sig_heatmap(  
  data,  
  sigs,  
  group_col,  
  markers,  
  scale = c("none", "row", "column"),  
  gene_id = "SYMBOL",  
  ranks_plot = FALSE,  
  slot = "counts",  
  ...  
)  
  
## S4 method for signature 'Seurat,ANY,character,ANY'  
sig_heatmap(  
  data,  
  sigs,  
  group_col,  
  markers,  
  scale = "none",  
  gene_id = "SYMBOL",  
  ranks_plot = FALSE,  
  slot = "counts",  
  ...  
)  
  
## S4 method for signature 'SummarizedExperiment,ANY,character,ANY'  
sig_heatmap(  
  data,  
  sigs,  
  group_col,  
  markers,  
  scale = "none",  
  gene_id = "SYMBOL",  
  ranks_plot = FALSE,  
  slot = "counts",  
  ...  
)
```

```
## S4 method for signature 'list,ANY,character,ANY'
sig_heatmap(
  data,
  sigs,
  group_col,
  markers,
  scale = "none",
  gene_id = "SYMBOL",
  ranks_plot = FALSE,
  slot = "counts",
  ...
)
```

### Arguments

data	expression data, can be matrix, DGEList, eSet, seurat, sce...
sigs	a vector of signature (Symbols) or a list of signatures
group_col	character or vector, specify the column name to compare in coldata
markers	a vector of gene names, listed the gene symbols of original markers pool
scale	could be one of 'none' (default), 'row' or 'column'
gene_id	character, indicate the ID type of rowname of expression data's , could be one of 'ENSEMBL', 'SYMBOL', ... default 'SYMBOL'
ranks_plot	logical, if to use ranks instead of expression of genes to draw heatmap
slot	character, indicate which slot used as expression, optional
...	params for <a href="#">ComplexHeatmap::Heatmap()</a>

### Value

patchwork object of heatmap

### Examples

```
data("im_data_6", "nk_markers")
sig_heatmap(
  data = im_data_6, sigs = nk_markers$HGNC_Symbol[1:10],
  group_col = "celltype:ch1",
  gene_id = "ENSEMBL"
)
```

---

sig\_rankdensity\_plot *Plot rank density*

---

### Description

Show the rank density of given signature in the given comparison.

### Usage

```
sig_rankdensity_plot(  
  data,  
  sigs,  
  group_col,  
  aggregate = FALSE,  
  slot = "counts",  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'matrix,vector,vector'  
sig_rankdensity_plot(  
  data,  
  sigs,  
  group_col,  
  aggregate = FALSE,  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'Matrix,vector,vector'  
sig_rankdensity_plot(  
  data,  
  sigs,  
  group_col,  
  aggregate = FALSE,  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'data.frame,vector,vector'  
sig_rankdensity_plot(  
  data,  
  sigs,  
  group_col,  
  aggregate = FALSE,  
  gene_id = "SYMBOL"  
)  
  
## S4 method for signature 'DGEList,vector,character'  
sig_rankdensity_plot(  
  data,  
  sigs,  
  group_col,  
  aggregate = FALSE,  
  gene_id = "SYMBOL"  
)
```

```
    data,
    sigs,
    group_col,
    aggregate = FALSE,
    slot = "counts",
    gene_id = "SYMBOL"
)

## S4 method for signature 'ExpressionSet,vector,character'
sig_rankdensity_plot(
  data,
  sigs,
  group_col,
  aggregate = FALSE,
  gene_id = "SYMBOL"
)

## S4 method for signature 'Seurat,vector,character'
sig_rankdensity_plot(
  data,
  sigs,
  group_col,
  aggregate = FALSE,
  slot = "counts",
  gene_id = "SYMBOL"
)

## S4 method for signature 'SummarizedExperiment,vector,character'
sig_rankdensity_plot(
  data,
  sigs,
  group_col,
  aggregate = FALSE,
  slot = "counts",
  gene_id = "SYMBOL"
)

## S4 method for signature 'list,vector,character'
sig_rankdensity_plot(
  data,
  sigs,
  group_col,
  aggregate = FALSE,
  slot = "counts",
  gene_id = "SYMBOL"
)
```

**Arguments**

data	expression data, can be matrix, DGEList, eSet, seurat, sce...
sigs	a vector of signature (Symbols)
group_col	character or vector, specify the column name to compare in coldata
aggregate	logical, if to aggregate expression according to group_col, default FALSE
slot	character, indicate which slot used as expression, optional
gene_id	character, indicate the ID type of rowname of expression data's , could be one of 'ENSEMBL', 'SYMBOL', ... default 'SYMBOL'

**Value**

ggplot or patchwork

**Examples**

```
data("im_data_6", "nk_markers")
sig_rankdensity_plot(
  data = im_data_6, sigs = nk_markers$HGNC_Symbol[1:10],
  group_col = "celltype:ch1", gene_id = "ENSEMBL"
)
```

---

sig_scatter_plot	<i>Scatter plot of signature for specific subset vs others</i>
------------------	--

---

**Description**

Scatter plot depicts mean expression for each signature gene in the specific subset against other cell types.

**Usage**

```
sig_scatter_plot(
  data,
  sigs,
  group_col,
  target_group,
  slot = "counts",
  xint = 1,
  yint = 1,
  gene_id = "SYMBOL"
)

## S4 method for signature 'matrix,vector,vector,character'
sig_scatter_plot(
  data,
```



```
    sigs,
    group_col,
    target_group,
    xint = 1,
    yint = 1,
    gene_id = "SYMBOL"
)

## S4 method for signature 'Matrix,vector,vector,character'
sig_scatter_plot(
  data,
  sigs,
  group_col,
  target_group,
  xint = 1,
  yint = 1,
  gene_id = "SYMBOL"
)

## S4 method for signature 'DGEList,vector,character,character'
sig_scatter_plot(
  data,
  sigs,
  group_col,
  target_group,
  slot = "counts",
  xint = 1,
  yint = 1,
  gene_id = "SYMBOL"
)

## S4 method for signature 'ExpressionSet,vector,character,character'
sig_scatter_plot(
  data,
  sigs,
  group_col,
  target_group,
  xint = 1,
  yint = 1,
  gene_id = "SYMBOL"
)

## S4 method for signature 'Seurat,vector,character,character'
sig_scatter_plot(
  data,
  sigs,
  group_col,
  target_group,
```

```

    slot = "counts",
    xint = 1,
    yint = 1,
    gene_id = "SYMBOL"
)

## S4 method for signature 'SummarizedExperiment,vector,character,character'
sig_scatter_plot(
  data,
  sigs,
  group_col,
  target_group,
  slot = "counts",
  xint = 1,
  yint = 1,
  gene_id = "SYMBOL"
)

## S4 method for signature 'list,vector,character,character'
sig_scatter_plot(
  data,
  sigs,
  group_col,
  target_group,
  slot = "counts",
  xint = 1,
  yint = 1,
  gene_id = "SYMBOL"
)

```

### Arguments

data	expression data, can be matrix, DGEList, eSet, seurat, sce...
sigs	a vector of signature (Symbols)
group_col	character or vector, specify the column name to compare in coldata
target_group	pattern, specify the group of interest as reference
slot	character, indicate which slot used as expression, optional
xint	intercept of vertical dashed line, default 1
yint	intercept of horizontal dashed line, default 1
gene_id	character, indicate the ID type of rowname of expression data's , could be one of 'ENSEMBL', 'SYMBOL', ... default 'SYMBOL'

### Value

patchwork or ggplot of scatter plot of median expression

**Examples**

```
data("im_data_6", "nk_markers")
sig_scatter_plot(
  sigs = nk_markers$HGNC_Symbol, data = im_data_6,
  group_col = "celltype:ch1", target_group = "NK",
  gene_id = "ENSEMBL"
)
```

---

voom_fit_treat	<i>return DGEList containing vfit by limma::voom (if normalize = TRUE) and tfit by limma::treat</i>
----------------	---

---

**Description**

return DGEList containing vfit by limma::voom (if normalize = TRUE) and tfit by limma::treat

**Usage**

```
voom_fit_treat(
  dge,
  group_col,
  target_group,
  normalize = TRUE,
  group = FALSE,
  lfc = 0,
  p = 0.05,
  batch = NULL,
  summary = TRUE,
  ...
)
```

**Arguments**

dge	DGEList object for DE analysis, including expr and samples info
group_col	character, column name of coldata to specify the DE comparisons
target_group	pattern, specify the group of interest, e.g. NK
normalize	logical, if the expr in data is raw counts needs to be normalized
group	logical, TRUE to separate samples into only 2 groups: 'target_group' and 'Others'; FALSE to set each level as a group
lfc	num, cutoff of logFC for DE analysis
p	num, cutoff of p value for DE analysis and permutation test if feature_selection = "rankproduct"
batch	vector of character, column name(s) of coldata to be treated as batch effect factor, default NULL
summary	logical, if to show the summary of DE analysis
...	omitted

**Value**

A DGEList containing vfit and tfit

# Index

## \* datasets

- cclle\_crc\_5, 4
- im\_data\_6, 19
- lm22, 20
- lm7, 22
- msigdb\_gobp\_nk, 24
- nk\_markers, 24

## \* internal

- mastR\_Package, 23

- cclle\_2\_wide, 3
- cclle\_crc\_5, 4
- clusterProfiler::GSEA(), 49
- ComplexHeatmap::Heatmap(), 53

- de\_analysis, 7
- de\_analysis(), 14
- DEGs\_Group, 5
- DEGs\_Group(), 43
- DEGs\_RP, 6
- DEGs\_RP(), 43
- depmap::depmap\_metadata(), 41, 42
- depmap::depmap\_TPM(), 3, 4, 42

- enrichplot::gseaplot2(), 49

- filter\_subset\_sig, 8
- filter\_subset\_sig, ANY-method  
(filter\_subset\_sig), 8
- filter\_subset\_sig, DGEList-method  
(filter\_subset\_sig), 8
- filter\_subset\_sig, list-method  
(filter\_subset\_sig), 8

- get\_de\_table, 13
- get\_de\_table(), 49
- get\_de\_table, DGEList, character, character-method  
(get\_de\_table), 13
- get\_de\_table, ExpressionSet, character, character-method  
(get\_de\_table), 13

- get\_de\_table, Matrix, vector, character-method  
(get\_de\_table), 13
- get\_de\_table, matrix, vector, character-method  
(get\_de\_table), 13
- get\_de\_table, Seurat, character, character-method  
(get\_de\_table), 13
- get\_de\_table, SummarizedExperiment, character, character-method  
(get\_de\_table), 13
- get\_degs, 11
- get\_degs(), 10
- get\_degs, DGEList, character, character-method  
(get\_degs), 11
- get\_degs, ExpressionSet, character, character-method  
(get\_degs), 11
- get\_degs, Matrix, vector, character-method  
(get\_degs), 11
- get\_degs, matrix, vector, character-method  
(get\_degs), 11
- get\_degs, Seurat, character, character-method  
(get\_degs), 11
- get\_degs, SummarizedExperiment, character, character-method  
(get\_degs), 11
- get\_gsc\_sig, 15
- get\_gsc\_sig, character, character-method  
(get\_gsc\_sig), 15
- get\_gsc\_sig, GeneSetCollection, character-method  
(get\_gsc\_sig), 15
- get\_lm\_sig, 16
- get\_panglao\_sig, 17
- gls2gsc, 18
- gls2gsc, list-method (gls2gsc), 18
- gls2gsc, vector-method (gls2gsc), 18
- grep(), 15–17, 41
- gsc\_plot, 18
- im\_data\_6, 19
- jsonlite::fromJSON(), 23
- limma::lmFit(), 31

- limma::`treat()`, [5](#), [6](#), [8](#), [31](#), [34](#), [43](#)
- limma::`voom()`, [31](#), [34](#)
- list\_panglao\_organs, [19](#)
- list\_panglao\_types, [20](#)
- list\_panglao\_types(), [17](#)
- lm22, [20](#)
- lm7, [22](#)
- lowess(), [30](#)
  
- mastR (mastR\_Package), [23](#)
- mastR\_Package, [23](#)
- merge\_markers, [23](#)
- msigdb::`getMsigdb()`, [24](#)
- msigdb::`getMsigdbVersions()`, [16](#)
- msigdb::`listCollections()`, [16](#)
- msigdb::`listSubCollections()`, [16](#)
- msigdb::`subsetCollection()`, [16](#)
- msigdb\_gobp\_nk, [24](#)
  
- nk\_markers, [24](#)
  
- pca\_matrix\_plot, [25](#)
- pca\_matrix\_plot, data.frame-method  
(pca\_matrix\_plot), [25](#)
- pca\_matrix\_plot, DGEList-method  
(pca\_matrix\_plot), [25](#)
- pca\_matrix\_plot, ExpressionSet-method  
(pca\_matrix\_plot), [25](#)
- pca\_matrix\_plot, Matrix-method  
(pca\_matrix\_plot), [25](#)
- pca\_matrix\_plot, matrix-method  
(pca\_matrix\_plot), [25](#)
- pca\_matrix\_plot, Seurat-method  
(pca\_matrix\_plot), [25](#)
- pca\_matrix\_plot, SummarizedExperiment-method  
(pca\_matrix\_plot), [25](#)
- pca\_matrix\_plot\_init, [28](#)
- plot\_diagnostics, [29](#)
- plot\_mean\_var, [30](#)
- plotPCAbiplot, [29](#)
- process\_data, [31](#)
- process\_data(), [13](#), [29](#), [30](#), [43](#)
- process\_data, DGEList, character, character-method  
(process\_data), [31](#)
- process\_data, ExpressionSet, character, character-method  
(process\_data), [31](#)
- process\_data, Matrix, vector, character-method  
(process\_data), [31](#)
- process\_data, matrix, vector, character-method  
(process\_data), [31](#)
- process\_data, Seurat, character, character-method  
(process\_data), [31](#)
- process\_data, SummarizedExperiment, character, character-method  
(process\_data), [31](#)
- pseudo\_sample\_list, [36](#)
- pseudo\_samples, [34](#)
- pseudo\_samples, matrix, data.frame-method  
(pseudo\_samples), [34](#)
- pseudo\_samples, matrix, vector-method  
(pseudo\_samples), [34](#)
- pseudo\_samples, Seurat, character-method  
(pseudo\_samples), [34](#)
- pseudo\_samples, SummarizedExperiment, character-method  
(pseudo\_samples), [34](#)
  
- remove\_bg\_exp, [37](#)
- remove\_bg\_exp, ANY, ANY, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, ANY, character, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, ANY, DGEList, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, ANY, ExpressionSet, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, ANY, missing, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, ANY, Seurat, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, ANY, SummarizedExperiment, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, DGEList, matrix, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp, matrix, matrix, vector-method  
(remove\_bg\_exp), [37](#)
- remove\_bg\_exp\_mat, [42](#)
  
- select\_sig, [43](#)
- select\_sig(), [13](#)
- sig\_boxplot, [44](#)
- sig\_boxplot, data.frame, vector, vector, character-method  
(sig\_boxplot), [44](#)
- sig\_boxplot, DGEList, vector, character, character-method  
(sig\_boxplot), [44](#)
- sig\_boxplot, ExpressionSet, vector, character, character-method  
(sig\_boxplot), [44](#)
- sig\_boxplot, list, vector, character, character-method  
(sig\_boxplot), [44](#)

- sig\_boxplot,Matrix,vector,vector,character-method (sig\_rankdensity\_plot), 54  
(sig\_boxplot), 44
- sig\_boxplot,matrix,vector,vector,character-method (sig\_rankdensity\_plot), 54  
(sig\_boxplot), 44
- sig\_boxplot,Seurat,vector,character,character-method (sig\_rankdensity\_plot), 54  
(sig\_boxplot), 44
- sig\_boxplot,SummarizedExperiment,vector,character,character-method (sig\_rankdensity\_plot), 54  
(sig\_boxplot), 44
- sig\_gseaplot, 47
- sig\_gseaplot,ANY,ANY-method  
(sig\_gseaplot), 47
- sig\_gseaplot,DGEList,ANY-method  
(sig\_gseaplot), 47
- sig\_gseaplot,list,ANY-method  
(sig\_gseaplot), 47
- sig\_gseaplot,MArrayLM,list-method  
(sig\_gseaplot), 47
- sig\_gseaplot,MArrayLM,vector-method  
(sig\_gseaplot), 47
- sig\_heatmap, 50
- sig\_heatmap,data.frame,ANY,vector,ANY-method  
(sig\_heatmap), 50
- sig\_heatmap,DGEList,ANY,character,ANY-method  
(sig\_heatmap), 50
- sig\_heatmap,ExpressionSet,ANY,character,ANY-method  
(sig\_heatmap), 50
- sig\_heatmap,list,ANY,character,ANY-method  
(sig\_heatmap), 50
- sig\_heatmap,Matrix,ANY,vector,ANY-method  
(sig\_heatmap), 50
- sig\_heatmap,matrix,character,vector,missing-method  
(sig\_heatmap), 50
- sig\_heatmap,matrix,character,vector,vector-method  
(sig\_heatmap), 50
- sig\_heatmap,matrix,list,vector,missing-method  
(sig\_heatmap), 50
- sig\_heatmap,Seurat,ANY,character,ANY-method  
(sig\_heatmap), 50
- sig\_heatmap,SummarizedExperiment,ANY,character,ANY-method  
(sig\_heatmap), 50
- sig\_rankdensity\_plot, 54
- sig\_rankdensity\_plot,data.frame,vector,vector-method  
(sig\_rankdensity\_plot), 54
- sig\_rankdensity\_plot,DGEList,vector,character-method  
(sig\_rankdensity\_plot), 54
- sig\_rankdensity\_plot,ExpressionSet,vector,character-method  
(sig\_rankdensity\_plot), 54
- sig\_rankdensity\_plot,list,vector,character-method  
(sig\_rankdensity\_plot), 54
- sig\_rankdensity\_plot,Matrix,vector,vector-method  
(sig\_rankdensity\_plot), 54
- sig\_rankdensity\_plot,matrix,vector,vector-method  
(sig\_rankdensity\_plot), 54
- sig\_rankdensity\_plot,Seurat,vector,character-method  
(sig\_rankdensity\_plot), 54
- sig\_rankdensity\_plot,SummarizedExperiment,vector,character-method  
(sig\_rankdensity\_plot), 54
- sig\_scatter\_plot, 56
- sig\_scatter\_plot,DGEList,vector,character,character-method  
(sig\_scatter\_plot), 56
- sig\_scatter\_plot,ExpressionSet,vector,character,character-method  
(sig\_scatter\_plot), 56
- sig\_scatter\_plot,list,vector,character,character-method  
(sig\_scatter\_plot), 56
- sig\_scatter\_plot,Matrix,vector,vector,character-method  
(sig\_scatter\_plot), 56
- sig\_scatter\_plot,matrix,vector,vector,character-method  
(sig\_scatter\_plot), 56
- sig\_scatter\_plot,Seurat,vector,character,character-method  
(sig\_scatter\_plot), 56
- sig\_scatter\_plot,SummarizedExperiment,vector,character,character-method  
(sig\_scatter\_plot), 56
- stats::prcomp(), 29
- voom\_fit\_treat, 59
- voom\_fit\_treat(), 34