

# Package ‘deconica’

April 25, 2018

**Type** Package

**Title** Deconvolution of transcriptome through Immune Component Analysis

**Version** 0.1.0

**Maintainer** The package maintainer <urszula.czerwinska@cri-paris.org>

**URL** <https://github.com/UrszulaCzerwinska/DeconICA>

**BugReports** <https://github.com/UrszulaCzerwinska/DeconICA/issues>

**Description** Deconvolution of transcriptome through Immune Component Analysis aims to provide an analytical pipeline that can be applied to complex mixtures, i.e. transcriptomes in order to extract latent immune variables and provide a tool to study biological insights. It requires mixture data, additional data like .gmt for enrichment analysis and pure profiles of signals. It also allows simulation of gene expression data and comparison with other tools.

**Depends** R (>= 3.4.1)

**License** GPL

**Encoding** UTF-8

**LazyData** yes

**Language** en-US

**RoxygenNote** 6.0.1

**Imports** fastICA (>= 1.2-1),  
stats (>= 3.4.1),  
Hmisc (>= 4.0.3),  
utils (>= 3.4.1),  
gtools (>= 3.5.0)

**Suggests** edgeR (>= 3.18.1),  
testthat,  
pheatmap,  
knitr,  
rmarkdown,  
CellMix,  
prettydoc,  
kableExtra,  
analytics,  
ACSNMineR (>= 0.16.8.25),  
matlabr (>= 1.5.0),  
ggplot2 (>= 2.2.1),  
corrplot (>= 0.84),

reshape ( $\geq 0.8.7$ ),  
 png ( $\geq 0.1.7$ ),  
 grDevices ( $\geq 3.4.1$ ),  
 NMF ( $\geq 0.20.6$ ),  
 MCPcounter

## biocViews

VignetteBuilder knitr

## R topics documented:

add_path . . . . .	3
assign_metagenes . . . . .	3
BEK_ica_overdecompose . . . . .	4
Biton.list . . . . .	5
cell_voting_immgen . . . . .	5
correlate_metagenes . . . . .	6
deconica . . . . .	7
dist_test_samples . . . . .	8
doICA . . . . .	9
doICABatch . . . . .	10
Example_ds . . . . .	11
export_for_correlation_java . . . . .	12
export_for_ICA . . . . .	13
generate_basis . . . . .	14
generate_markers . . . . .	15
gene_enrichment_test . . . . .	16
get_matlab_2 . . . . .	17
get_max_correlations . . . . .	18
get_scores . . . . .	19
identify_immune_comp . . . . .	20
ImmgenHUGO . . . . .	20
import_ICA_res . . . . .	21
is_logscale . . . . .	22
LM22.list . . . . .	22
lolypop_plot_corr . . . . .	23
make_list . . . . .	24
most_variant_IC . . . . .	24
plot_dist_test . . . . .	25
prepare_data_for_ica . . . . .	26
radar_plot_corr . . . . .	27
run_fastica . . . . .	28
run_fastica_import . . . . .	30
run_matlab_code_2 . . . . .	31
run_matlab_script_2 . . . . .	32
scores_corr_plot . . . . .	32
simulate_gene_expresssion . . . . .	33
stacked_proportions_plot . . . . .	34
TIMER_cellTypes . . . . .	35

## Index

---

add_path	<i>Create PATHs to add to MATLAB PATHs</i>
----------	--

---

**Description**

Create PATHs to add to MATLAB PATHs

**Usage**

```
add_path(path)
```

**Arguments**

path	path to add
------	-------------

**Value**

A character vector

**Examples**

```
add_path("~/")
```

---

assign_metagenes	<i>Assign components to a metagene through mutual reciprocity</i>
------------------	---

---

**Description**

Attributes labels to components under condition of mutual reciprocal correlation

**Usage**

```
assign_metagenes(r, exclude_name = "M8_IMMUNE")
```

**Arguments**

r	the correlation matrix, r matrix, can be generated from <a href="#">correlate_metagenes</a> function
exclude_name	name of the components (present in r) to be excluded from this analysis (for example immune), by default "M8_IMMUNE" is excluded

**Details**

This function assign a component to a metagene/profile through verification if the component's the maximal correlation points to a given profile and if for this profile the maximal correlation points back the that component. In mathematical terms, given correlations between the set of profiles/metagenes  $A = A_1, \dots, A_m$  and  $S$  components matrix  $S = IC1, \dots, ICN$ , if

$$S_i = \operatorname{argmax}_i(\operatorname{corr}(A_j, S))$$

and

$$A_j = \operatorname{argmax}_j(\operatorname{corr}(S_i, A))$$

**Value**

returns a `data.frame` with component name in the first column and assigned profile/metagene name in second column

**See Also**

[get\\_max\\_correlations](#), [correlate\\_metagenes](#)

**Examples**

```
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 5,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$S,
  gene.names = res_run_ica$names)

assign_metagenes(corr$r)
```

---

BEK\_ica\_overdecompose *Decomposition of a transcriptome data*

---

**Description**

A dataset overdecomposed (into 100 components). Data were downloaded from GEO, then [run\\_fastica](#) using MATLAB algorithm with stabilization as applied.

**Usage**

```
BEK_ica_overdecompose
```

**Format**

a list containing

**A** A ICA matrix (sample scores)

**S** S ICA matrix (gene scores)

**names** gene names

**samples** sample names

**counts** raw counts (non centered)

**log.counts** log2 counts (non centered)

**Details**

Source: Bekhouche I, Finetti P, Adelaïde J, Ferrari A et al. High-resolution comparative genomic hybridization of inflammatory breast cancer and identification of candidate genes. PLoS One 2011 Feb 9;6(2):e16950. PMID: 21339811

**Source**

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE23720>

---

Biton.list	<i>Array of all Metagenes</i>
------------	-------------------------------

---

**Description**

list of metagenes

**Usage**

```
Biton.list
```

**Format**

list of 11 elements

**Source**

[http://www.cell.com/cell-reports/abstract/S2211-1247\(14\)00904-8](http://www.cell.com/cell-reports/abstract/S2211-1247(14)00904-8)

---

cell_voting_immgen	<i>Attribute cell type to a component</i>
--------------------	---

---

**Description**

From [gene\\_enrichment\\_test](#) result constructs a summary table counting percentage of a certain cell type attributed to a component. Works only with Immgen signatures

**Usage**

```
cell_voting_immgen(enrich, n = 10)
```

**Arguments**

enrich	enrichment results from <a href="#">gene_enrichment_test</a>
n	n top results taken into account, 10 by default

**Value**

list of `data.frame` for each non NULL result of enrichment list from [gene\\_enrichment\\_test](#)

**See Also**

[gene\\_enrichment\\_test](#)

## Examples

```
set.seed(123)
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = TRUE,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$S,
  gene.names = res_run_ica$names)

assign <- assign_metagenes(corr$r)
immune_c<- identify_immune_comp(corr$r[, "M8_IMMUNE"], assign[, "component"], threshold = 0.1)

enrichment <- gene_enrichment_test(
  res_run_ica$S,
  res_run_ica$names,
  names(immune_c),
  alternative = "greater",
  p.adjust.method = "none",
  n = 50,
  n.consider = 100,
  p.value.threshold = 0.005
)

cell_voting_immgen(enrichment$enrichment)
```

---

correlate_metagenes	<i>Correlate components with known ranked lists of genes</i>
---------------------	--

---

## Description

Components obtained, for example, with [run\\_fastica](#) can be characterized through correlation with known ranked list (metagenes or profiles), by default this function is using metagenes from Biton et al. (2015), Cell. It is using [rcorr](#) function for correlations

## Usage

```
correlate_metagenes(S, gene.names, metagenes = Biton.list, threshold = -Inf,
  n.genes.intersect = 30, orient.long = TRUE, orient.max = FALSE, ...)
```

## Arguments

S	S matrix of components
gene.names	list of gene names, needs to be of the same length as nrow of S, for ICA it is recommended to run <a href="#">run_fastica</a> with <code>with.names = TRUE</code> to assure compatibility
metagenes	named list of datasets, each with two columns 1st - gene names, 2nd - ranks, by default 11 metagenes from Biton et al. (2015), Cell
threshold	threshold for components (columns of S) to be applied before correlation, default set to -Inf (all ranks are kept)

`n.genes.intersect` minimum of genes that should intersect between a component and a metagene to keep the component in correlation matrix

`orient.long` orient by long tails, default TRUE

`orient.max` orient by maximal correlation, default FALSE, can be used if there is no long tails

`...` additional params you can pass to [rcorr](#)

### Value

a correlation matrix with correlation coefficient  $r$ , p.values  $P$  and number of overlapping genes  $n$ , oriented  $S$  matrix

### See Also

[rcorr](#) [run\\_fastica](#) [make\\_list](#)

### Examples

```
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 5,
  with.names = TRUE
)
correlate_metagenes(
  S = res_run_ica$S,
  gene.names = res_run_ica$names)
```

---

deconica

*deconICA: Deconvolution of transcriptome through Immune Component Analysis*

---

### Description

deconICA is a package to perform unsupervised deconvolution of complex mixtures, it contains functions implementing the pipeline of data interpretation

### Details

See the README on [CRAN](#) or [GitHub](#)

### deconICA functions

NA

---

dist_test_samples	<i>Test impact of each Independent Component</i>
-------------------	--

---

## Description

This function is applying distribution statistical test (i.e. `t.test`, `wilcox.test`) to evaluate which ICs have highest impact on differences between samples

## Usage

```
dist_test_samples(A, sample.names, quant = c(0.1, 0.9), X.counts, test.type,
  thr = 0.1, isLog = NULL, return = "p.value", wide = TRUE)
```

## Arguments

<code>A</code>	result of <code>run_fastica</code> the A matrix
<code>sample.names</code>	names of samples, should correspond to number of columns of A
<code>quant</code>	quantiles to use, in form of <code>c(x, y)</code>
<code>X.counts</code>	expression data
<code>test.type</code>	test of distributions to perform
<code>thr</code>	threshold of maximal p.value considered 0.1 by default
<code>isLog</code>	by default NULL, if X is not counts but log, provide the base of log, for natural logarithm use <code>exp(1)</code>
<code>return</code>	if you want to return p.values select
<code>wide</code>	should the output matrix be in wide format (FALSE preferable for plotting)

## Value

returns a matrix (in long or wide) format

## Examples

```
# numerical matrix
set.seed(123)
S <- matrix(stats::rnbino(10000, mu = 6, size = 10), 500, 80)
dat <- matrix(runif(1600,min =1, max=10 ), 80, 80, byrow = TRUE)
A <- dat / rowSums(dat)
X <- data.frame(S %>% A)
res_run_ica <- run_fastica(X, row.center = TRUE, n.comp = 5, overdecompose = FALSE)

#stats::t.test
dist_test_samples(A = res_run_ica$A,
  sample.names = res_run_ica$samples,
  X.counts = res_run_ica$log.counts,
  test.type = "t.test",
  isLog = 2,
  return = "p.value",
  thr= 0.5)

#edgeR::exactTest
```



```

dist_test_samples(A = res_run_ica$A,
sample.names = res_run_ica$samples,
X.counts = res_run_ica$log.counts,
test.type = "exactTest",
isLog = 2,
return = "p.value",
thr= 0.5)

#for plotting
res.ttest <- dist_test_samples(A = res_run_ica$A,
sample.names = res_run_ica$samples,
X.counts = res_run_ica$log.counts,
test.type = "t.test",
isLog = 2,
return = "p.value",
thr= 0.5,
wide = FALSE)

plot_dist_test(res.ttest, plot.type = "density")
plot_dist_test(res.ttest, plot.type = "line")

```

doICA

*Call doICA matlab function*

## Description

function used inside [run\\_fastica](#) to run fastICA with icasso stabilization. Matlab engine is necessary

## Usage

```

doICA(df.scaled.t, names, samples, path_global = getwd(), n, name = FALSE,
export.corr = FALSE, corr_folder = "CORRELATION", matlbpth = NULL,
fasticaph = paste0(path.package("deconica"), quiet = TRUE), "/fastica++"))

```

## Arguments

df.scaled.t	scaled numerical data matrix
names	gene names, no duplicates
samples	sample names
path_global	path where files will be saved
n	number of components
name	FALSE by default, name of dataset is used, you can put your name
export.corr	FALSE by default, if you want to use a java correlation function later or select TRUE
corr_folder	"CORRELATION" by default, only if you selected export.corr = TRUE
matlbpth	is found automatically with <a href="#">get_matlab_2</a> function, replace if not functional
fasticaph	path to fastica++ repository with MATLAB scripts

Value

it returns A, S matrices of ICA and names and samples for coherence

See Also

[get\\_matlab](#), [run\\_fastica](#), [export\\_for\\_ICA](#), [run\\_matlab\\_code](#), [import\\_ICA\\_res](#), [codeexport\\_for\\_correlation\\_java](#)

Examples

```
## Not run:
data(Example_ds)
res.pre <-
  prepare_data_for_ica(Example_ds[, -1], names = Example_ds[, 1])
res.do <- doICA(
  df.scaled.t = res.pre$df.scaled,
  names = res.pre$names,
  samples = res.pre$samples,
  path_global = getwd(),
  n = 5,
  name = "test",
  export.corr = FALSE
)

## End(Not run)
```

---

doICABatch	doBatchICA
------------	------------

---

Description

prepares the data (scales and removes duplicates), runs doBatchICA.m MATLAB script

Usage

```
doICABatch(df, vec, path_global = getwd(), names, samples, name = FALSE,
  matlbpth = NULL, fasticaph = paste0(path.package("deconica", quiet =
  TRUE), "/fastica++"))
```

Arguments

df	numerical data matrix
vec	vector of values for which ICA should be computed
path_global	path were files will be saved, current directory by default
names	gene names
samples	sample names
name	name of the dataset, if not provided, name of R variable
matlbpth	path to matlab, found automatically with <a href="#">get_matlab_2</a>
fasticaph	path to fastica++

**Value**

plots of stability and MSTD if possible

**Examples**

```
## Not run:
data(Example_ds)
doICABatch(
  Example_ds[, -1],
  seq(2, 4, 1),
  names = Example_ds[, 1],
  samples = colnames(Example_ds[, -1]),
  name = "test",
  fasticaph = paste0(path.package("deconica", quiet = FALSE), "/fastica++")
)

## End(Not run)
```

---

Example_ds	<i>Example of a cancer dataset</i>
------------	------------------------------------

---

**Description**

A sample 60 randomly selected samples from transcriptome of inflammatory breast cancer (IBC). Data were centred and in transformed in log2 before sampling

**Usage**

Example\_ds

**Format**

a dataframe with the

**rows** 21320

**columns** 61

**first column** is related to GENE names

**Details**

Bekhouche I, Finetti P, Adelaïde J, Ferrari A et al. High-resolution comparative genomic hybridization of inflammatory breast cancer and identification of candidate genes. PLoS One 2011 Feb 9;6(2):e16950. PMID: 21339811

**Source**

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE23720>

---

export\_for\_correlation\_java

*Exports S ICA matrix in a specific format*

---

## Description

needed for an external function in java

## Usage

```
export_for_correlation_java(corr_folder = "CORRELATION", names, S, samples, A,
  ncomp, name, path_global_1 = getwd())
```

## Arguments

corr_folder	export folder name "CORRELATION" by default
names	gene names
S	S ICA matrix
samples	sample names
A	A ICA matrix
ncomp	number of computed components
name	name of the dataset
path_global_1	absolute path

## Value

saves on the drive in corr\_folder exported files

## See Also

[run\\_fastica](#), [import\\_ICA\\_res](#), [doICA](#), [export\\_for\\_ICA](#)

## Examples

```
## Not run:
data(Example_ds)
res.pre <-
  prepare_data_for_ica(Example_ds[, -1], names = Example_ds[, 1])
res.do <- doICA(
  df.scaled.t = res.pre$df.scaled,
  names = res.pre$names,
  samples = res.pre$samples,
  path_global = getwd(),
  n = 5,
  name = "test",
  export.corr = FALSE
)
export_for_correlation_java(
  S = res.do$S,
  A = t(res.do$A),
  names = res.do$names,
```

```

samples = res.do$samples,
name = "test",
ncomp = 5
)

## End(Not run)

```

---

export\_for\_ICA

*Export files*


---

## Description

export files in right format to run fastICA in MATLAB or BiodICA

## Usage

```

export_for_ICA(df.scaled.t, names, samples, path_global = getwd(),
  name = FALSE, n = "")

```

## Arguments

df.scaled.t	scaled numerical matrix
names	gene names, vector of character string
samples	sample names, vector of character string
path_global	path to export files, current directory by default
name	name of the dataset
n	number of components

## Value

writes files on the drive in indicated location

## See Also

[run\\_fastica](#), [import\\_ICA\\_res](#), [doICA](#), [export\\_for\\_correlation\\_java](#)

## Examples

```

## Not run:
data(Example_ds)
res.pre <-
  prepare_data_for_ica(Example_ds[, -1], names = Example_ds[, 1])
export_for_ICA(res.pre$df.scaled,
  res.pre$names,
  res.pre$samples,
  path_global = getwd(),
  name = "test",
  n = 5)

## End(Not run)

```

---

generate_basis	<i>Generate basis matrix</i>
----------------	------------------------------

---

## Description

It generates a basis matrix that can be used for regression from list of weighted markers

## Usage

```
generate_basis(df, sel.comp, markers, orient.long = TRUE)
```

## Arguments

df	output of run_fastica containing at least S and names elements
sel.comp	components identified as specific sources (i.e. immune cells), by default it takes all components of S matrix, can be provided as valid column names or numeric index
markers	list of markers that should be used for basis matrix (i.e. "gene.list" from generate_markers), can be also simple vector or list of gene names
orient.long	TRUE by default, if you modified S matrix and you don't want it to be oriented select FALSE

## Value

it returns a data.frame (basis matrix) that can be used for regression or visualization purposes

## Examples

```
set.seed(123)
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 20,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$S,
  gene.names = res_run_ica$names)

assign <- assign_metagenes(corr$r)

immune <- identify_immune_comp(corr$r[, "M8_IMMUNE"], assign[, "component"], threshold = 0.1)

markers <- generate_markers(df = res_run_ica, n = 10, sel.comp= names(immune), return= "gene.list")
basis <- generate_basis(df = res_run_ica, sel.comp= names(immune), markers= markers )
pheatmap::pheatmap(basis )
```

---

generate_markers	<i>Generate markers from components</i>
------------------	---

---

## Description

It extracts from set of components (i.e. ICA S matrix) the n top genes (with weights if needed) to use as marker list or markers with weights for estimation of abundance through [get\\_scores](#)

## Usage

```
generate_markers(df, n = 30, thr = Inf, sel.comp = paste("IC",
  1:ncol(df$S), sep = ""), return = "gene.list", orient.long = TRUE)
```

## Arguments

df	list (usually output of run_fastica) containing at least S and names elements
n	number of top genes considered from each signature, n = 30 by default
thr	max gene expression, if removal of outliers is necessary, Inf (no threshold) by default.
sel.comp	components of interest (i.e. identified as specific to some profiles/metagenes (i.e. immune cells)), by default it takes all columns of S matrix, can be provided as valid column names or numeric index
return	return gene.list or gene.ranked
orient.long	TRUE by default, if S is oriented change to FALSE

## Value

function returns either list of gene markers gene.list for each component or list of gene.ranked which are gene names with weights

## See Also

run\_fastica, [get\\_scores](#)

## Examples

```
set.seed(123)
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 20,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$S,
  gene.names = res_run_ica$names)

assign <- assign_metagenes(corr$r)

immune <- identify_immune_comp(corr$r[, "M8-IMMUNE"], assign[, "component"], threshold = 0.1)

generate_markers(df = res_run_ica, n = 10, sel.comp= names(immune))
generate_markers(df = res_run_ica, n = 10, sel.comp= names(immune), return= "gene.ranked")
```

---

gene\_enrichment\_test    *Enrichment analysis*

---

## Description

Computes an enrichment score (fisher exact test) in provided signatures for selected components

## Usage

```
gene_enrichment_test(S, gene.names, immune.ics, gmt = ImmgenHUGO,
  alternative = c("greater", "lower"), p.adjust.method = c("holm",
    "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"), n = 100,
  n.consider = 500, min_module_size = 5, max_module_size = 500,
  p.value.threshold = 0.05, orient.long = TRUE)
```

## Arguments

<code>S</code>	matrix of components, dim n corresponding to genes, m corresponding to number of components, use oriented matrix
<code>gene.names</code>	character vector of gene names, length needs to be equal to n
<code>immune.ics</code>	vector of character names of components to use for enrichment test
<code>gmt</code>	data.frame obtained from gmt file with a function <a href="#">format_from_gmt</a> , by default Immgen signatures <a href="http://Immgen.org">http://Immgen.org</a>
<code>alternative</code>	greater will check for enrichment, less will check for depletion
<code>p.adjust.method</code>	correction method
<code>n</code>	number of top genes that will be used to test signature
<code>n.consider</code>	number of genes from the positive end to be considered
<code>min_module_size</code>	minimal module size from gmt file to be considered in enrichment
<code>max_module_size</code>	maximum module size from gmt file to be considered in enrichment
<code>p.value.threshold</code>	maximal p-value (corrected if correction is enabled) that will be displayed
<code>orient.long</code>	TRUE by default, in case you applied transformation to your S components, select FALSE.

## Details

gene\_enrichment\_test runs enrichment of a component (or any ranked list) in known (i.e. immune cell types) signatures. It was designed to use S matrix from [run\\_fastica\\_fisher.test](#) only on components identified as correlated with immune metagene through function [identify\\_immune\\_ic](#) and it searches in Immgen signatures <http://Immgen.org>.

## Value

returns value if there is an enrichment in provided signatures:

**metagenes** interpreted metagene gene ranking

**enrichment** full results of the enrichment analysis sorted by corrected p.value

**genes.list** list of genes used for enrichment



**See Also**

[identify\\_immune\\_comp](#) identifying immune related components, [run\\_fastica](#) for running Independent Components Analysis, and [enrichment](#) for enrichment in gmt files

**Examples**

```
set.seed(123)
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 41,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$$,
  gene.names = res_run_ica$names)

assign <- assign_metagenes(corr$r)

immune_c<- identify_immune_comp(corr$r[, "M8_IMMUNE"], assign[, "component"], threshold = 0.1)

gene_enrichment_test(
  res_run_ica$$,
  res_run_ica$names,
  names(immune_c),
  alternative = "greater",
  p.adjust.method = "none",
  n = 50,
  n.consider = 100,
  p.value.threshold = 0.005
)
```

---

get\_matlab\_2

Find matlab path

---

**Description**

This tries to find matlab's path using a system which command, and then, if not found, looks at `getOption("matlab.path")`. If not path is found, it fails.

**Usage**

```
get_matlab_2(try_defaults = TRUE, desktop = FALSE, splash = FALSE,
  display = FALSE, wait = TRUE, mpath = NULL)
```

**Arguments**

<code>try_defaults</code>	(logical) If matlab is not found from <code>Sys.which</code> , and <code>matlab.path</code> not found, then try some default PATHs for Linux and OS X.
<code>desktop</code>	Should desktop be active for MATLAB?
<code>splash</code>	Should splash be active for MATLAB?

display	Should display be active for MATLAB?
wait	Should R wait for the command to finish. Both passed to <a href="#">system</a> and adds the -wait flag.
mpath	path to matlab if known

**Value**

Character of command for matlab

**Examples**

```
if (matlabr::have_matlab()) {
  get_matlab_2()
}
```

---

get\_max\_correlations    *Assign through maximal correlations*

---

**Description**

It assigns maximal correlations between set of correlated vectors

**Usage**

```
get_max_correlations(corr)
```

**Arguments**

corr                      list of correlation matrices with correlation coefficients and p-values, can be obtained from [correlate\\_metagenes](#) or [rcorr](#)

**Value**

data.frame with matched column names, Pearson correlation coefficient, p.value

**See Also**

[rcorr](#), [correlate\\_metagenes](#), [assign\\_metagenes](#)

**Examples**

```
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 5,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$S,
  gene.names = res_run_ica$names)

get_max_correlations(corr)
```

---

get_scores	<i>Get abundance scores</i>
------------	-----------------------------

---

## Description

It calculates abundance scores through a mean of marker genes

## Usage

```
get_scores(df, markers.list, summary = "mean", ...)
```

## Arguments

df	gene matrix with samples in columns and genes in rows with named rows
markers.list	list of genes or list of genes with weights
summary	can be any type of mean i.e. mean, gm_mean (geometric mean), harmonic_mean, weighted.mean. For weighted mean weights are needed along with gene names
...	optional parameters for the mean function

## Value

Function returns numerical value for each column (sample) of provided data frame

## Examples

```
set.seed(123)
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 20,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$$,
  gene.names = res_run_ica$names)

assign <- assign_metagenes(corr$r)

immune <- identify_immune_comp(corr$r[, "M8_IMMUNE"], assign[, "component"], threshold = 0.1)
counts.abs <- (2^res_run_ica$log.counts)-1
row.names(counts.abs) <- res_run_ica$names

markers <- generate_markers(df = res_run_ica, n = 10,
                           sel.comp= names(immune),
                           return= "gene.list")
get_scores (counts.abs, markers, summary = "mean", na.rm = TRUE)

markers <- generate_markers(df = res_run_ica, n = 10,
                           sel.comp= names(immune),
                           return= "gene.ranked")
get_scores (counts.abs, markers, summary = "weighted.mean", na.rm = TRUE)
```

---

`identify_immune_comp`    *Identify components related to immune signal*

---

### Description

Identify components related to immune signal

### Usage

```
identify_immune_comp(x, l, threshold = 0.1)
```

### Arguments

<code>x</code>	the correlation with immune metagene can be retrieved from <a href="#">correlate_metagenes</a> output
<code>l</code>	vector of names of assigned components
<code>threshold</code>	lower bound for filtering correlation [0,1]

### Value

it returns data frame of component names and correlations passing the threshold

### Examples

```
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 20,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$S,
  gene.names = res_run_ica$names)

assign <- assign_metagenes(corr$r)

identify_immune_comp(corr$r[, "M8_IMMUNE"], assign[, "component"], threshold = 0.1)
```

---

ImmgenHUGO

*Cell type signatures*

---

### Description

Imported in correct format with [format\\_from\\_gmt](#) and parsed

### Usage

```
ImmgenHUGO
```

**Format**

a dataframe with the

**module** first column

**module length** second column

**gene names** third column

**Source**

<http://www.immgen.org>

---

import_ICA_res	<i>Import results of ICA</i>
----------------	------------------------------

---

**Description**

imports files run in Matlab or precomputed

**Usage**

```
import_ICA_res(name, ncomp, path_global_1)
```

**Arguments**

name	name of the dataset
ncomp	number of components
path_global_1	absolute path of the files

**Value**

imports A and S ICA matrix

**See Also**

[run\\_fastica](#), [export\\_for\\_ICA](#), [doICA](#), [export\\_for\\_correlation\\_java](#)

**Examples**

```
## Not run:
data(Example_ds)
res.pre <-
  prepare_data_for_ica(Example_ds[, -1], names = Example_ds[, 1])
res.do <- doICA(
  df.scaled.t = res.pre$df.scaled,
  names = res.pre$names,
  samples = res.pre$samples,
  path_global = getwd(),
  n = 5,
  name = "test",
  export.corr = FALSE
)
import_ICA_res("test_5", 5, paste0(getwd(), "/test_5/"))

## End(Not run)
```

---

is_logscale	<i>Verify if data is in log scale</i>
-------------	---------------------------------------

---

**Description**

Verify if data is in log scale

**Usage**

```
is_logscale(x)
```

**Arguments**

x	data.frame or matrix
---	----------------------

**Value**

TRUE or FALSE

**Examples**

```
M <- matrix(sample(-1:14, 100, replace = TRUE), 10, 10, byrow = TRUE)
is_logscale(M)
M2 <- 2^M
is_logscale(M2)
```

---

LM22.list	<i>Array of all Metagenes</i>
-----------	-------------------------------

---

**Description**

list of 22 immune cell type profiles

**Usage**

```
LM22.list
```

**Format**

list of 22 data.frames

**Source**

[http://www.cell.com/cell-reports/abstract/S2211-1247\(14\)00904-8](http://www.cell.com/cell-reports/abstract/S2211-1247(14)00904-8)

---

lolypop_plot_corr	<i>Lolypop plot for correlations</i>
-------------------	--------------------------------------

---

## Description

Plot correlations between one metagene or known profile and all components in a form of linear plot which is a variant of a signal plot. Wrapper using ggplot2.

## Usage

```
lolypop_plot_corr(r, col, head.size = 10, head.color = "value",
  digits = 2, head.text.size = 3.5, head.text.color = "white",
  vertical = TRUE)
```

## Arguments

r	correlation matrix r matrix of output <a href="#">correlate_metagenes</a>
col	select column either index or column name
head.size	size of the point of correlation
head.color	by default colored by correlation values, if you want one color provide color name
digits	parameter of <a href="#">round</a> for the correlation showed on the plot. integer indicating the number of decimal places (round) or significant digits (signif) to be used.
head.text.size	size of the correlation text font
head.text.color	color of the correlation text font
vertical	TRUE for vertical plot, FALSE for horizontal plot

## Details

Values are order from highest correlation to lowest correlation. Colors and fonts can be overwritten. To see all correlations simultaneously choose [radar\\_plot\\_corr](#)

## Value

returns [ggplot](#)

## See Also

[ggplot](#), [aes\\_string](#), [theme\\_bw](#), [geom\\_point](#), [labs](#), [scale\\_color\\_distiller](#), [coord\\_flip](#), [geom\\_segment](#)

## Examples

```
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 20,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$S,
```

```

    gene.names = res_run_ica$names)
#horizontal
lolypop_plot_corr(corr$r,2, vertical =FALSE)
# vertical
lolypop_plot_corr(corr$r,"M8_IMMUNE")
#change colors
lolypop_plot_corr(corr$r,"M8_IMMUNE",head.color = "black" , head.text.color = "green")
#remove title
lolypop_plot_corr(corr$r,"M8_IMMUNE")+ ggplot2::labs(title="",subtitle="")

```

---

make_list	<i>Make list of weighted markers</i>
-----------	--------------------------------------

---

### Description

Transforms a data frame with multiple columns into a named list of weighted markers with gene names in the first column and values in the second column.

### Usage

```
make_list(df)
```

### Arguments

df                      data.frame to be transformed with gene names in the row.names

### Value

named list of data.frames with gene names in the first column and values in the second column.

### Examples

```

X <- as.data.frame(matrix(runif(10000), 50, 10))
row.names(X) <- paste("A",1:nrow(X), sep="")
make_list(X)

```

---

most_variant_IC	<i>Compute variance explained by each Independent Component</i>
-----------------	---

---

### Description

Compute variance explained by each Independent Component

### Usage

```
most_variant_IC(S, A, X, n = 5)
```



**Arguments**

S	result of <a href="#">run_fastica</a> the S matrix
A	result of <a href="#">run_fastica</a> the A matrix
X	data, either post-PCA data of <a href="#">run_fastica</a> X matrix
n	number of top ICs if n = "all" then fraction of variance explained for all ICs is returned

**Value**

returns a data frame with n top ICs numbers ranked by their fraction of variance explained

**Examples**

```
set.seed(123)
res_fastica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 20,
  with.names = TRUE
)
most_variant_IC(res_fastica$S, res_fastica$A, res_fastica$X, n =3)

res <- most_variant_IC(res_fastica$S, res_fastica$A, res_fastica$X, n =5)
barplot(as.matrix(t(res)))
```

---

plot_dist_test	<i>Plot results of density test</i>
----------------	-------------------------------------

---

**Description**

Wrapper over [ggplot](#) plotting either rank or density versus selected value in [dist\\_test\\_samples](#) (p.value or test statistics)

**Usage**

```
plot_dist_test(df, plot.type = c("line", "density"))
```

**Arguments**

df	data.frame in long format
plot.type	can be either "line" or "density"

**Value**

returns a line or density plot of p.value or test statistics versus rank or density

**See Also**

[ggplot](#), [stat\\_density](#), [theme\\_bw](#), [aes](#), [geom\\_line](#)

## Examples

```
#numerical matrix
set.seed(134)
S <- matrix(stats::rnbino(10000, mu = 6, size = 10), 500, 80)
dat <- matrix(runif(1600,min =1, max=10 ), 80, 80, byrow = TRUE)
A <- dat / rowSums(dat)
X <- data.frame(S %*% A)
res_run_ica <- run_fastica(X, row.center = TRUE, n.comp = 5, overdecompose = FALSE)

#run the funtion selecting wide = FALSE
res.ttest <- dist_test_samples(A = res_run_ica$A,
sample.names = res_run_ica$samples,
X.counts = res_run_ica$log.counts,
test.type = "t.test",
thr=0.5,
isLog = 2,
return = "p.value",
wide = FALSE)

#plot results
plot_dist_test(res.ttest, plot.type = "density")
plot_dist_test(res.ttest, plot.type = "line")
```

---

prepare\_data\_for\_ica    *Formats data for ICA in MATLAB*

---

## Description

Formats data for ICA in MATLAB

## Usage

```
prepare_data_for_ica(df, names, samples = NULL, isLog = TRUE)
```

## Arguments

df	numerical data matrix
names	gene names character vector
samples	if not provided column names will be used

## Value

df.scaled	scaled data without duplicates
names	gene names without duplicates
non.scaled	non scaled data without duplicates
samples	sample names

## See Also

[run\\_fastica](#), [import\\_ICA\\_res](#), [doICA](#), [doICABatch](#)

## Examples

```
data(Example_ds)
prepare_data_for_ica(Example_ds[, -1], names = Example_ds[, 1])
```

---

radar_plot_corr	<i>Radar plot of correlations</i>
-----------------	-----------------------------------

---

## Description

Wrapper using ggplot2 to plot correlations between components and given metagenes or pure profiles

## Usage

```
radar_plot_corr(df, ax.size = NULL, size.el.txt = 15, point.size = 5)
```

## Arguments

df	output of function <a href="#">correlate_metagenes</a> - correlation matrix with correlation and p-values
ax.size	define size of axis labels, adapts automatically by default
size.el.txt	define general size of letters, 15 by default
point.size	size parameter in <a href="#">geom_point</a>

## Value

Radar plots for correlations of each input component with matagene/profile, Returns a list containing the data.frame df used to generate the plot - long format - and the plot itself p.

## See Also

[ggplot](#), [geom\\_point](#), [coord\\_polar](#), [theme\\_bw](#), [facet\\_wrap](#), [scale\\_color\\_distiller](#), [theme](#), [element\\_text](#)

## Examples

```
res_run_ica <- run_fastica (
  Example_ds,
  overdecompose = FALSE,
  n.comp = 20,
  with.names = TRUE
)
corr <- correlate_metagenes(
  S = res_run_ica$$,
  gene.names = res_run_ica$names)

radar_plot_corr(corr)
data <- radar_plot_corr(corr)$df

#change plot
radar_plot_corr(corr)$p +
```

```

ggplot2::labs(title="11 Biton et al. metagenes vs my ICA components",
              subtitle="Pearson correlation coefficients")

radar_plot_corr(corr, point.size = 1)

radar_plot_corr(corr, point.size = 0)$p +
ggplot2::geom_point(size = 8, alpha = 0.4)

```

run\_fastica

*Decompose dataset with ICA.*

## Description

This is a wrapper of [fastICA](#). It allows compute number of ICs overdecompose for over decomposition for immune deconvolution.

## Usage

```

run_fastica(X, overdecompose = TRUE, row.center = TRUE,
  with.names = FALSE, gene.names = NULL, samples = NULL,
  alg.typ = "parallel", method = "C", n.comp = 100, isLog = TRUE,
  R = TRUE, path_global = getwd(), matlbpth = NULL,
  fasticaph = paste0(path.package("deconica", quiet = TRUE), "/fastica++"),
  export.corr = FALSE, name = NULL, ...)

```

## Arguments

X	a data matrix with n rows representing observations and p columns representing variables, place gene names in the first column and select with.names = TRUE
overdecompose	check TRUE to let select best number of components for deconvolution, for datasets >120 columns, n.comp will be set to 100, if <120 then number of components will be selected according to Kaiser Rule (90 percent of variance explained)
row.center	if TRUE subtract row mean from data
with.names	if first column of X is row.names please indicate TRUE, in case of duplicated names, the transcript with highest variance will be kept, names need to be HUGO names, if names are not provided at this step, you can provide them later
gene.names	character vector of row names - gene names
samples	if samples names different from column names
alg.typ	if alg.typ == "parallel" the components are extracted simultaneously (the default). if alg.typ == "deflation" the components are extracted one at a time.
method	if method == "R" then computations are done exclusively in R (default). The code allows the interested R user to see exactly what the algorithm does. if method == "C" then C code is used to perform most of the computations, which makes the algorithm run faster. During compilation the C code is linked to an optimized BLAS library if present, otherwise stand-alone BLAS routines are compiled.

n.comp	number of components to be extracted
isLog	if data is in log TRUE if data is in counts FALSE
R	if TRUE (default) the R version of fastICA is running, else the matlab version (you need to provide parameters of your matlab engine)
path_global	only if R = FALSE, the global path where files will be written, current directory by default
matlbpth	only if R = FALSE, the path to matlab engine, it uses <code>get_matlab</code> to find path to your matlab automatically
fasticapth	path to repository of source matlab code, it is set by default as coming with the package
export.corr	TRUE if you need to export S matrix in a specific format for correlation in external java app
name	important for Matlab version, defines the name of your files
...	other possible parameters for <code>fastICA</code>

### Value

A list containing the following components as in `fastICA`

**X** pre-processed data matrix (after PCA)

**K** pre-whitening matrix that projects data onto the first n.comp principal components.

**W** estimated un-mixing matrix (see definition in details)

**A** estimated mixing matrix

**S** estimated source matrix

**names** if with.names = TRUE will contain row names list

**counts** if isLog = FALSE will contain initial matrix without duplicated genes

**log.counts** initial matrix without duplicated genes in  $\log_2(x+1)$  before centering

**samples** sample names as provided

### See Also

`fastICA` <https://cran.r-project.org/web/packages/fastICA/index.html>

### Examples

```
# numerical matrix
S <- matrix(runif(10000), 10, 2)
A <- matrix(sample(-3:3, 16, replace = TRUE), 2, 8, byrow = TRUE)
X <- data.frame(S %*% A)
run_fastica(X, row.center = TRUE, n.comp = 2, overdecompose = FALSE)

#matlab
## Not run:
run_fastica(X, row.center = TRUE, n.comp = 3, overdecompose = FALSE, R = FALSE)

## End(Not run)
# matrix with gene names
S <- matrix(runif(10000), 5000, 2)
A <- matrix(c(1, 1, -1, 3), 2, 2, byrow = TRUE)
X <- data.frame(S %*% A)
```

```
names <- paste("A", 1:nrow(X), sep="")
X <- cbind(names, X)
run_fastica(X, row.center = TRUE, n.comp = 2, overdecompose = FALSE, with.names = TRUE)
```

---

run_fastica_import	<i>Reproduce process of run_fastica</i>
--------------------	---

---

## Description

Applies preprocessing of [run\\_fastica](#) but instead of running ICA it imports matlab output files. It is handy if you run the matlab idenpendly or if you lost R session data

## Usage

```
run_fastica_import(X, overdecompose = TRUE, row.center = TRUE,
  with.names = FALSE, gene.names = NULL, n.comp = 100, isLog = TRUE,
  import = TRUE, path_global = getwd(), name = NULL, ...)
```

## Arguments

X	a data matrix with n rows representing observations and p columns representing variables, place gene names in the first column and select with.names = TRUE
overdecompose	check TRUE to let select best number of components for deconvolution, for datasets >120 columns, n.comp will be set to 100, if <120 then number of components will be selected according to Kaiser Rule (90 percent of variance explained)
row.center	if TRUE subtract row mean from data
with.names	if first column of X is row.names please indicate TRUE, in case of duplicated names, the transcript with highest variance will be kept, names need to be HUGO names, if names are not provided at this step, you can provide them later
gene.names	character vector of row names - gene names
n.comp	number of components to be extracted
isLog	if data is in log TRUE if data is in counts FALSE
import	imports data only if TRUE
path_global	only if R = FALSE, the global path where files will be written, current directory by default
name	important for Matlab version, defines the name of your files
...	other possible parameters for <a href="#">fastICA</a>

## Value

an object as [run\\_fastica](#)

**Examples**

```
## Not run:

# numerical matrix
S <- matrix(runif(10000), 10, 2)
A <- matrix(sample(-3:3, 16, replace = TRUE), 2, 8, byrow = TRUE)
X <- data.frame(S %*% A)
#matlab
run_fastica(X, row.center = TRUE, n.comp = 2, overdecompose = FALSE, R = FALSE)
run_fastica_import(X, row.center = TRUE, n.comp = 2, overdecompose = FALSE, import=TRUE)

## End(Not run)
```

---

run_matlab_code_2	<i>Runs matlab code</i>
-------------------	-------------------------

---

**Description**

This function takes in matlab code, where the last line must end with a `;`, and returns the exit status, slightly modified version of [run\\_matlab\\_code](#)

**Usage**

```
run_matlab_code_2(code, matlbpth = NULL, endlines = TRUE, verbose = TRUE,
  add_clear_all = FALSE, paths_to_add = NULL, ...)
```

**Arguments**

<code>code</code>	Character vector of code.
<code>matlbpth</code>	path to matlab engine
<code>endlines</code>	Logical of whether the semicolon ( <code>;</code> ) should be pasted to each element of the vector.
<code>verbose</code>	Print out filename to run
<code>add_clear_all</code>	Add <code>clear all;</code> to the beginning of code
<code>paths_to_add</code>	Character vector of PATHs to add to the script using <a href="#">add_path</a>
<code>...</code>	Options passed to <a href="#">run_matlab_script</a>

**Value**

Exit status of matlab code

**Examples**

```
if (matlabr::have_matlab()){
  run_matlab_code_2("disp(version)", matlbpth = "matlbpth")
  run_matlab_code_2("disp(version)", paths_to_add = "~/")
  run_matlab_code_2(c("disp('The version of the matlab is: ')", "disp(version)"))
  run_matlab_code_2(c("x = 5", "disp(['The value of x is ', num2str(x)])"))
}
```

---

run_matlab_script_2	<i>Run matlab script</i>
---------------------	--------------------------

---

### Description

This function runs a matlab script, and returns exit statuses, slightly modified version of [run\\_matlab\\_script](#)

### Usage

```
run_matlab_script_2(fname, matlbpth = NULL, verbose = TRUE,
  desktop = FALSE, splash = FALSE, display = FALSE, wait = TRUE, ...)
```

### Arguments

fname	Filename of matlab script (.m file)
matlbpth	path to matlab engine
verbose	print diagnostic messages
desktop	Should desktop be active for MATLAB?
splash	Should splash be active for MATLAB?
display	Should display be active for MATLAB?
wait	Should R wait for the command to finish. Both passed to <a href="#">system</a> and adds the -wait flag.
...	Options passed to <a href="#">system</a>

### Value

Exit status of matlab code

---

scores_corr_plot	<i>Correlation plot of abundance scores</i>
------------------	---

---

### Description

Produces correlation plot of abundance scores estimated versus expected

### Usage

```
scores_corr_plot(x, y, ...)
```

### Arguments

x	matrix or data.frame of abundance scores, samples in rows and cell types in columns
y	matrix or data.frame of expected (or to compare) abundance scores, samples in rows and cell types in columns
...	additional parameters for method from <a href="#">corrplot</a>



**Details**

correlation plot between different abundance scores of cell types in samples, correlates both matrices with each other merging two data.frames by row.names, on [corrplot](#) is.corr parameter is set to FALSE

**Value**

**correlation plot** based on [corrplot](#)

corr.full full correlation matrix

corr.filtere correlation without correlation with itself

**See Also**

[rcorr](#), [corrplot](#)

**Examples**

```
x <- matrix(runif(1000), ncol = 10, nrow = 10)
y <- matrix(runif(1000), ncol = 10, nrow = 10)
row.names(x) <- row.names(y) <- paste0("S", 1:10)
colnames(x) <- paste0("CL_", 1:10, "_estimated")
colnames(y) <- paste0("CL_", 1:10, "_expected")
scores_corr_plot(x,y, method = "number", tl.col = "black")
scores_corr_plot(x,y, method = "square", tl.col = "black")
```

---

simulate\_gene\_expression

*Simulate gene expression*

---

**Description**

Function simulating gene expression of mixed cell types with a perturbator (i.e. proliferation, stress)

**Usage**

```
simulate_gene_expression(x, n, p, z = 0, dist.cells = list(dist =
  stats::rnbino, size = 3, mu = 5), markers = NULL, mfold = 2,
  CLnames = NULL, genes = NULL, dist.noise.sources = list(dist =
  stats::rnorm, mean = 0, sd = 0.05), alpha = 1,
  dist.noise.global = list(dist = stats::rgamma, shape = 5, scale = 1),
  perturb = .pos.gaussian, pargs = list(p = p, mean = 0.5, sd = 0.2, lwr =
  0, upr = 1))
```

**Arguments**

x	number of cell types
n	number of genes
p	number of samples
z	number of perturbators
dist.cells	distribution and parameters from which cell profiles will be drawn

markers	number of markers that will distinguish cell types, can be a number (the same number of marker genes for cell types and perturbator), can be a vector of length $x+z$ , it will be set to $\text{ceiling}(n/20)$ if not provided
mfold	number of fold change between gene markers and other genes
CLnames	column names (cell and perturbator)
genes	gene names
dist.noise.sources	noise that will be added to each column of basis matrix (to each source)
alpha	parameter for the dirichlet distribution from which are drawn the cell proportions, using <code>rdirichlet</code> .
dist.noise.global	distribution and parameters of global noise (added to each sample one mixture is obtained)
perturb	function of distribution
pargs	arguments of perturbation function

### Value

**expression** mixed expression matrix  
**marker.genes** list of marker genes per cell type  
**basis\_matrix** pure cell type and perturbator profile  
**prop** pure cell type and perturbator proportions (from 0 to 1)

### Examples

```
res <- simulate_gene_expression(3, 30, 10, 2, markers = c(4,5,5,3,4))
#visualise the basis matrix
pheatmap::pheatmap(res$basis_matrix)
#visualize expression
pheatmap::pheatmap(res$expression)
#observe distribution of signals
par(mfrow=c(2,2))
apply(res$basis_matrix, 2, hist)
```

---

stacked\_proportions\_plot  
*Plot cell proportions*

---

### Description

Plots scores for all samples as a fraction of one in each samples

### Usage

```
stacked_proportions_plot(dat)
```

### Arguments

**dat** scores data.frame with cell types in lines and samples in columns

**Value**

a stacked bar plot based on ggplot2

**Examples**

```
#random matrix y
y <- data.frame(matrix(runif(10000), ncol = 100, nrow = 10))
#plot
stacked_proportions_plot(y)
```

---

TIMER_cellTypes	<i>TIMER signatures</i>
-----------------	-------------------------

---

**Description**

Signatures of 9 cell types published as part of TIMER tool

**Usage**

TIMER\_cellTypes

**Format**

a dataframe with the

**module** first column

**module length** second column

**gene names** third column

**Details**

Li, Bo, et al. "Comprehensive analyses of tumor immunity: implications for cancer immunotherapy." Genome biology 17.1 (2016): 174.

**Source**

<http://cistrome.org/TIMER/>

# Index

## \*Topic **datasets**

- BEK\_ica\_overdecompose, [4](#)
- Biton.list, [5](#)
- Example\_ds, [11](#)
- ImmgenHUGO, [20](#)
- LM22.list, [22](#)
- TIMER\_cellTypes, [35](#)
  
- add\_path, [3](#), [31](#)
- aes, [25](#)
- aes\_string, [23](#)
- assign\_metagenes, [3](#), [18](#)
  
- BEK\_ica\_overdecompose, [4](#)
- Biton.list, [5](#)
  
- cell\_voting\_immgen, [5](#)
- coord\_flip, [23](#)
- coord\_polar, [27](#)
- correlate\_metagenes, [3](#), [4](#), [6](#), [18](#), [20](#), [23](#), [27](#)
- corrplot, [32](#), [33](#)
  
- deconica, [7](#)
- deconica-package (deconica), [7](#)
- dist\_test\_samples, [8](#), [25](#)
- doICA, [9](#), [12](#), [13](#), [21](#), [26](#)
- doICABatch, [10](#), [26](#)
  
- element\_text, [27](#)
- enrichment, [17](#)
- Example\_ds, [11](#)
- export\_for\_correlation\_java, [10](#), [12](#), [13](#), [21](#)
- export\_for\_ICA, [10](#), [12](#), [13](#), [21](#)
  
- facet\_wrap, [27](#)
- fastICA, [28–30](#)
- fisher.test, [16](#)
- format\_from\_gmt, [16](#), [20](#)
  
- gene\_enrichment\_test, [5](#), [16](#)
- generate\_basis, [14](#)
- generate\_markers, [15](#)
- geom\_line, [25](#)
- geom\_point, [23](#), [27](#)
  
- geom\_segment, [23](#)
- get\_matlab, [10](#), [29](#)
- get\_matlab\_2, [9](#), [10](#), [17](#)
- get\_max\_correlations, [4](#), [18](#)
- get\_scores, [15](#), [19](#)
- ggplot, [23](#), [25](#), [27](#)
  
- identify\_immune\_comp, [17](#), [20](#)
- ImmgenHUGO, [20](#)
- import\_ICA\_res, [10](#), [12](#), [13](#), [21](#), [26](#)
- is\_logscale, [22](#)
  
- labs, [23](#)
- LM22.list, [22](#)
- lolypop\_plot\_corr, [23](#)
  
- make\_list, [7](#), [24](#)
- most\_variant\_IC, [24](#)
  
- plot\_dist\_test, [25](#)
- prepare\_data\_for\_ica, [26](#)
  
- radar\_plot\_corr, [23](#), [27](#)
- rcorr, [6](#), [7](#), [18](#), [33](#)
- round, [23](#)
- run\_fastica, [4](#), [6–10](#), [12](#), [13](#), [16](#), [17](#), [21](#), [25](#), [26](#), [28](#), [30](#)
- run\_fastica\_import, [30](#)
- run\_matlab\_code, [10](#), [31](#)
- run\_matlab\_code\_2, [31](#)
- run\_matlab\_script, [31](#), [32](#)
- run\_matlab\_script\_2, [32](#)
  
- scale\_color\_distiller, [23](#), [27](#)
- scores\_corr\_plot, [32](#)
- simulate\_gene\_expresssion, [33](#)
- stacked\_proportions\_plot, [34](#)
- stat\_density, [25](#)
- system, [18](#), [32](#)
  
- theme, [27](#)
- theme\_bw, [23](#), [25](#), [27](#)
- TIMER\_cellTypes, [35](#)