Package: liana (via r-universe)

May 30, 2026

Type Package

Title LIANA: a LIgand-receptor ANalysis frAmework

Version 0.1.14.9000

Description LIANA provides a number of methods and resources for ligand-receptor interaction inference from scRNA-seq data. This version is specifically designed for Seurat v4 and SeuratObject v4 compatibility. It includes improved Python environment handling and better integration with existing workflows.

License GPL-3 + file LICENSE

URL https://zaoqu-liu.github.io/liana/, https://github.com/Zaoqu-Liu/liana

BugReports https://github.com/Zaoqu-Liu/liana/issues

Encoding UTF-8

VignetteBuilder knitr

Depends R(>= 4.0)

Imports utils, reticulate, stringr, magrittr, purrr, tidyr, tibble, dplyr, readr, rlang, OmnipathR, SingleCellExperiment, scran, scater, scuttle, SeuratObject (>= 4.0.0), tidyselect, ggplot2, ComplexHeatmap, RColorBrewer, methods

Suggests CellChat, knitr, BiocStyle, rmarkdown, markdown, testthat (>= 3.0.0), sessioninfo, Seurat (>= 4.0.0), entropy, circlize, furrr (>= 0.2.0), future, tidyverse, devtools, decoupleR, broom, igraph, patchwork, basilisk, basilisk.utils

biocViews scater, scran, SingleCellExperiment, ComplexHeatmap

Remotes sqjin/CellChat, saezlab/OmnipathR, saezlab/decoupleR, Zaoqu-Liu/Connectome

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev perl python3 libx11-dev zlib1g-dev

Repository https://zaoqu-liu.r-universe.dev

Date/Publication 2026-01-25 04:59:47 UTC

RemoteUrl https://github.com/Zaoqu-Liu/liana

RemoteRef main

RemoteSha a14953e154850f6fcf1a73a35a39050724090314

Contents

`assign_lr_weights` *Helper function to assign weights*

Description

Helper function to assign weights

Usage

```
assign_lr_weights(lrs, resource, entity = "ligand")
```

Arguments

`lrs`	ligand_receptor tibble
`resource`	resource
`entity`	name of the entity

`call_cellchat` *Run CellChat with OmniPath function [[DEPRECATED]]*

Description

Run CellChat with OmniPath function [[DEPRECATED]]

Usage

```
call_cellchat(
  sce,
  op_resource,
  .format = TRUE,
  exclude_anns = c(),
  nboot = 100,
  assay = "RNA",
  .seed = 1004,
  .normalize = FALSE,
  .do_parallel = FALSE,
  .raw_use = TRUE,
  expr_prop = 0,
  organism = "human",
  thresh = 1,
  de_thresh = 0.05,
  ...
)
```

Arguments

`sce`	Seurat object as input
`op_resource`	OmniPath Intercell Resource DN
`.format`	bool whether to format output
`exclude_anns`	Annotation criteria to be excluded
`nboot`	number of bootstraps to calculate p-value
`assay`	assay name (RNA by default)
`.seed`	random seed
`.normalize`	# bool whether to normalize non-normalized data with
`.do_parallel`	whether to parallelize or not
`.raw_use`	whether use the raw data or gene expression data projectected to a ppi (should be kept to TRUE)
`expr_prop`	minimum proportion of gene expression per cell type (0 by default), yet perhaps one should consider setting this to an appropriate value between 0 and 1, as an assumptions of these method is that communication is coordinated at the cluster level.

`organism` Obtain CellChatDB for which organism ('mouse' or 'human')

`thresh` p-value threshold (1 by default)

`de_thresh` diff expression of genes p-value

`...` Arguments passed on to `CellChat::subsetCommunication`

`object` CellChat object

`net` Alternative input is a data frame with at least with three columns defining the cell-cell communication network ("source","target","interaction_name")

`slot.name` the slot name of object: slot.name = "net" when extracting the inferred communications at the level of ligands/receptors; slot.name = "netP" when extracting the inferred communications at the level of signaling pathways

`sources.use` a vector giving the index or the name of source cell groups

`targets.use` a vector giving the index or the name of target cell groups.

`signaling` a character vector giving the name of signaling pathways of interest

`pairLR.use` a data frame consisting of one column named either "interaction_name" or "pathway_name", defining the interactions of interest

`datasets` select the inferred cell-cell communications from a particular 'datasets' when inputing a data frame 'net'

`ligand.pvalues,ligand.logFC,ligand.pct.1,ligand.pct.2` set threshold for ligand genes
ligand.pvalues: threshold for pvalues in the differential expression gene analysis (DEG)
ligand.logFC: threshold for logFoldChange in the DEG analysis; When ligand.logFC > 0, keep upgulated genes; otherwise, kepp downregulated genes
ligand.pct.1: threshold for the percent of expressed genes in the defined 'positive' cell group. keep genes with percent greater than ligand.pct.1
ligand.pct.2: threshold for the percent of expressed genes in the cells except for the defined 'positive' cell group

`receptor.pvalues,receptor.logFC,receptor.pct.1,receptor.pct.2` set threshold for receptor genes

Details

CellChat's objects are not lazily documented/exported thus the whole package has to be imported.

Value

A DF of intercellular communication network

`call_connectome` *Function to call connectome with databases from OmniPath [[DEPRECATED]]*

Description

Function to call connectome with databases from OmniPath [[DEPRECATED]]

Usage

```
call_connectome(sce, op_resource = NULL, .format = TRUE, assay, ...)
```

Arguments

sce	Seurat object as input
op_resource	OmniPath Intercell Resource DN
.format	bool whether to format output
assay	assay name
...	dot params passed to connectome

Details

Stats: 1) The 'weight_norm' edge attribute is derived from the normalized expression of the ligand and the receptor in the single-cell data. 2) The 'weight_scale' edge attribute is derived from the z-scores of the ligand and the receptor in each edge, and is of higher value when the ligand and receptor are more specific to a given pair of cell types 3) DEG p-values for L and R

Value

An unfiltered connectome results df

---

call_italk *Run iTALK with OmniPath data [[DEPRECATED]]*

Description

Run iTALK with OmniPath data [[DEPRECATED]]

Usage

```
call_italk(sce, op_resource, assay = "RNA", .format = TRUE, ...)
```

Arguments

sce	Seurat object or SingleCellExperiment as input
op_resource	OmniPath Intercell Resource DN
assay	assay to use from Seurat object
.format	bool: whether to format output
...	Parameters passed to Seurat FindMarkers (ref requires import)

Details

In order to be comparable with the remainder of the methods, we calculate the mean of the ligand and receptor logFC. The original implementation only uses the DE genes above a certain logFC threshold.

Value

An unfiltered iTALK df sorted by relevance

In this case, we use the product of the logFC rather than thresholding, as in the original implementation.

`call_natmi` *Call NATMI Pipeline from R with Resources Querried from OmniPath [[DEPRECATED]]*

Description

Call NATMI Pipeline from R with Resources Querried from OmniPath [[DEPRECATED]]

Usage

```
call_natmi(
  sce,
  op_resource,
  expr_file = "em.csv",
  meta_file = "metadata.csv",
  output_dir = "NATMI_test",
  reso_name = "placeholder",
  assay = "RNA",
  num_cor = 4,
  conda_env = NULL,
  assay.type = "logcounts",
  .format = TRUE,
  .overwrite_data = TRUE,
  .seed = 1004,
  .natmi_path = NULL,
  .delete_input_output = FALSE,
  layer = NULL
)
```

Arguments

`sce`	Seurat or SingleCellExperiment object
`op_resource`	List of OmniPath resources
`expr_file`	expression matrix file name
`meta_file`	annotations (i.e. clusters) file name

output_dir	NATMI output directory
reso_name	name of the resource usually in the format list(name = op_resource)
assay	Seurat assay to be used
num_cor	number of cores to be used
conda_env	name of the conda environment via which NATMI is called
assay.type	logcounts by default, but it's converted back into counts as suggested by the authors
.format	bool whether to format output
.overwrite_data	bool whether Extract and overwrite csv with data from Seurat Object
.seed	random seed
.natmi_path	path of NATMI code and dbs (by default set to liana path)
.delete_input_output	logical whether to delete input and output after run.

Details

This function will save NATMI dbs folder, then it will call the NATMI Python from the NATMI dir and save the output into a specified directory in NATMI's path. It will then load the csvs and format the output to a list of lists.

By default, NATMI's path is set to that of LIANA, but any alternative path can be passed

==========

NATMI Arguments: –interDB INTERDB lrc2p (default) has literature supported ligand-receptor pairs | lrc2a has putative and literature supported ligand-receptor pairs | the user-supplied interaction database can also be used by calling the name of database file without extension –interSpecies INTERSPECIES human (default) | mouse | expandp | expanda –emFile EMFILE the path to the normalised expression matrix file with row names (gene identifiers) and column names (cell-type/single-cell identifiers) –annFile ANNFILE the path to the metafile in which column one has single-cell identifiers and column two has corresponding cluster IDs (see file 'toy.sc.ann.txt' as an example) –species SPECIES human (default) | mouse | rat | zebrafish | fruitfly | chimpanzee | dog | monkey | cattle | chicken | frog | mosquito | nematode | thalecress | rice | riceblastfungus | bakeryeast | neurosporacrassa | fissionyeast | eremotheciumgossypii | kluyveromyceslactis, 21 species are supported –idType IDTYPE symbol (default) | entrez(https://www.ncbi.nlm.nih.gov/gene) | ensembl(https://www.ensembl.org/) | uniprot(https://www.uniprot.org/) | hgnc(https://www.genenames.org/) | mgi(http://www.informatics.jax.org/mgihome/nomen/index.shtml) | custom(gene identifier used in the expression matrix) –coreNum CORENUM the number of CPU cores used, default is one –out OUT the path to save the analysis results (Taken From NATMI's GitHub Page)

Stats: 1) The mean-expression edge weights 2) The specificity-based edge weights * a weight of 1 means both the ligand and receptor are only expressed in one cell type

Note that 'call_natmi' will write the expression matrix to CSV each time its called, unless .overwrite_data is set to FALSE! This can be an extremely time consuming step when working with large datasets

Also, NATMI will sometimes create duplicate files, so please consider saving each run in a new folder. An easy fix would be to simply delete the output, but I am reluctant to automatically delete files via an R script.

Value

DF with NATMI results

`call_sca` *Function to call SingleCellSignalR with databases from OmniPath [[DEPRECATED]]*

Description

Function to call SingleCellSignalR with databases from OmniPath [[DEPRECATED]]

Usage

```
call_sca(
  sce,
  op_resource,
  .format = TRUE,
  assay = "RNA",
  assay.type = "logcounts",
  layer = NULL,
  ...
)
```

Arguments

`sce`	SingleCellExperiment or SeuratObject as input
`op_resource`	OmniPath Intercell Resource DN
`.format`	bool whether to format output
`assay`	Seurat assay data to use
`assay.type`	count slot (logcounts by default)
`...`	arguments passed to 'SCAomni::cell_signaling'

Details

Stats: LRScore = sqrt(LR product)/mean(raw counts) * sqrt(LR product) where expression of l > 0 and r > 0 LRScore = 1 is the highest (~ most likely hit), 0 is the lowest.

Value

An unfiltered SCA tibble

call_squidpy *Call Squidpy Pipeline via reticulate with OmniPath and format results*

Description

Call Squidpy Pipeline via reticulate with OmniPath and format results

Usage

```
call_squidpy(
  sce,
  op_resource,
  seed = 1004,
  conda_env = NULL,
  use_active_env = TRUE,
  assay = "RNA",
  assay.type = "logcounts",
  layer = NULL,
  ...
)
```

Arguments

sce	SingleCellExperiment or Seurat Object as input
op_resource	Tibble or list of OmniPath resources, typically obtained via select_resource
seed	seed passed to squidpy's ligrec function
conda_env	python conda environment to run Squidpy. If NULL, will try: 1. Currently active reticulate environment 2. Environment from RETICULATE_PYTHON_ENV environment variable 3. Fall back to "liana_env" for backward compatibility
use_active_env	logical. If TRUE and a Python environment is already active, use it instead of switching. Default: TRUE
assay	assay name
assay.type	count slot (logcounts by default)
...	kwargs passed to Squidpy; For more information see: https://squidpy.readthedocs.io/en/latest/api/squidpy.gr.ligrec.html

Details

CellPhoneDBv2 algorithm re-implementation in Python.

Note that 'cluster_key' is a parameter passed to the Squidpy function, by default this will be set to the default Ident of the Seurat object.

Value

A list of Squidpy results for each resource

chord_freq *Frequency ChordDiagram*

Description

Frequency ChordDiagram

Usage

```
chord_freq(
  liana_res,
  source_groups = NULL,
  target_groups = NULL,
  cex = 1,
  transparency = 0.4,
  facing = "clockwise",
  adj = c(-0.5, 0.05),
  ...
)
```

Arguments

liana_res	aggregated 'liana_wrap' results from multiple methods, or alternatively results from running 'liana_wrap' with a single method. Should be filtered by some condition (e.g. preferential consesus ranking, specific interactions, etc).
source_groups	names of the source (sender) cell types (NULL = no filter)
target_groups	names of the target cell types (NULL = no filter)
cex	label relative font size
transparency	transparency
facing	axis label rotation (check 'circlize::circos.text' for options)
...	other paramters passed to 'circlize::chordDiagram'
offset	for text.

decomplexify *Helper Function to 'decomplexify' ligands and receptors into individual subunits*

Description

Helper Function to 'decomplexify' ligands and receptors into individual subunits

Usage

```
decomplexify(resource, columns = c("source_genesymbol", "target_genesymbol"))
```

Arguments

resource	a ligrec resource
columns	columns to separate and pivot long (e.g. genesymbol or uniprot), ‘source_genesymbol‘ and ‘target_genesymbol‘ by default

Details

takes any number of columns, and assumes ‘_‘ as sep.

Value

returns a longer tibble with complex subunits on seperate rows

---

decompose_tensor *Wrapper function to run ‘cell2cell_tensor‘ decomposition on a prebuilt tensor.*

Description

Wrapper function to run ‘cell2cell_tensor‘ decomposition on a prebuilt tensor.

Usage

```
decompose_tensor(
  tensor,
  rank = NULL,
  tf_optimization = "robust",
  seed = 1337,
  upper_rank = 25,
  elbow_metric = "error",
  smooth_elbow = FALSE,
  init = "svd",
  svd = "numpy_svd",
  factors_only = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

tensor	Tensor-cell2cell Prebuilt.Tensor class instance
rank	Ranks for the Tensor Factorization (number of factors to deconvolve the original tensor). If NULL, then rank selection is performed using the ‘elbow_rank_selection‘ function.

tf_optimization	indicates whether running the analysis in the ''regular'' or the ''robust'' way. The regular way means that the tensor decomposition is run 10 times per rank evaluated in the elbow analysis, and 1 time in the final decomposition. Additionally, the optimization algorithm has less number of iterations in the regular than the robust case (100 vs 500) and less precision (tolerance of 1e-7 vs 1e-8). The robust case runs the tensor decomposition 20 times per rank evaluated in the elbow analysis, and 100 times in the final decomposition. Here we could use the 'tf_optimization='regular'', which is faster but generates less robust results. We recommend using 'tf_optimization='robust', which takes longer to run (more iteractions and more precise too).
seed	Random seed integer
upper_rank	Upper bound of ranks to explore with the elbow analysis.
init	Initialization method for computing the Tensor Factorization. 'svd', 'random'
factors_only	whether to return only the factors after factorization
verbose	verbosity logical
...	Dictionary containing keyword arguments for the c2c.compute_tensor_factorization function. The function deals with 'random_state' (seed) and 'rank' internally.

Value

an instance of the cell2cell.tensor.BaseTensor class (via reticulate). If build_only is TRUE, then no rank selection or tensor decomposition is returned. Otherwise, returns a tensor with factorization results.

---

filter_nonabundant_celltypes

Filter nun-abundant cell types

Description

Filter nun-abundant cell types

Usage

```
filter_nonabundant_celltypes(
  sce,
  sample_col,
  idents_col,
  min_cells = 10,
  min_samples = 3,
  min_prop = 0.2,
  ctqc = NULL
)
```

Arguments

sce	SingleCellExperiment Object
sample_col	column with sample ids
idents_col	column with cell identity (cell type) ids
min_cells	minimum cells per identity in each sample
min_samples	minimum samples per cell identity
min_prop	minimum proportion of samples in which a cell identity is present (with at least 'min_cells')
ctqc	cell type quality control summary obtained from 'get_abundance_summary'

Value

a filtered SingleCellExperiment Object

FormatConnectome *Helper function to filter and format connectome*

Description

Helper function to filter and format connectome

Usage

```
FormatConnectome(conn)
```

Arguments

conn	connectome object

FormatiTALK *Helper Function to Filter and format iTalk results*

Description

Helper Function to Filter and format iTalk results

Usage

```
FormatiTALK(italk_res, remove.na = TRUE)
```

Arguments

italk_res	iTalk results object
remove.na	bool whether to filter NA

FormatNatmi *Load NATMI results from folder and format appropriately*

Description

Load NATMI results from folder and format appropriately

Usage

```
FormatNatmi(output_path, resource_names, .format = TRUE)
```

Arguments

output_path	NATMI output path
resource_names	results for which resources to load
.format	bool whether to format output

Value

A list of NATMI results per resource loaded from the output directory.

FormatSCA *Helper function to format SingleCellSignalR results*

Description

Helper function to format SingleCellSignalR results

Usage

```
FormatSCA(sca_res, remove.na = TRUE)
```

Arguments

sca_res	Unformatted SCA results
remove.na	bool whether to filter SCA output

`generate_homologs` *Function to generate a homologous OmniPath resource*

Description

Function to generate a homologous OmniPath resource

Usage

```
generate_homologs(
  op_resource,
  target_organism,
  max_homologs = 5,
  .missing_fun = NULL,
  symbols_dict = NULL,
  columns = c("source_genesymbol", "target_genesymbol"),
  verbose = TRUE
)
```

Arguments

`op_resource`	a resource in the format of OmniPath/LIANA
`target_organism`	'ncbi_taxid' or 'name' of the target organism. See 'show_homologene' for available organisms via OmnipathR's 'HomoloGene'
`max_homologs`	Determines the max number of homologs to be translated. Certain genes will have multiple homolog matches, with some having also certain isoforms considered. To exclude cases in which the number of matched homologs is too high, one can adjust the homologs parameter. Setting this to '1' would mean that one-to-many homolog matches are discarded
`.missing_fun`	approach to handle missing interactions. By default set to 'NULL' which would mean that any interactions without a homology match will be filtered. This can be set to e.g. 'str_to_title' when working with murine symbols. Then if a gene has no matched homolog, instead of discarding it, the '.missing_fun' will be used to format the name from human. Hence, increasing the number of matches, but likely introducing some mismatches.
`symbols_dict`	'NULL' by default, then 'get_homologene_dict' is called to generate a dictionary from OmniPathR's homologene resource. Alternatively, one can pass their own symbols_dictionary.
`verbose`	logical for verbosity
`source`	name of the source (ligand) column
`target`	name of the target (receptor) column

Value

a converted ligand-receptor resource

generate_lr_geneset *Generate a geneset resource for each LR*

Description

Generate a geneset resource for each LR

Usage

```
generate_lr_geneset(sce, resource, lr_sep = "^")
```

Arguments

sce	SingleCellExperiment object with liana_tensor_c2c computed
resource	resource with 'source', 'target', 'weight' columns

Value

a tibble in decoupleR format

generate_orthologs *Deprecated call to generate_homologs*

Description

Deprecated call to generate_homologs

Usage

```
generate_orthologs(...)
```

Arguments

...	Arguments passed on to generate_orthologs

get_c2c_factors *Returns tensor cell2cell results*

Description

Returns tensor cell2cell results

Usage

```
get_c2c_factors(sce, group_col = NULL, sample_col)
```

Arguments

sce	SingleCellExperiment with factors output from tensor-cell2cell
group_col	context descriptor - to be obtained from ‘colData(sce)‘
sample_col	context/sample names - obtained from ‘colData(sce)‘

get_connectome *Function to obtain connectome-like weights*

Description

Function to obtain connectome-like weights

Usage

```
get_connectome(lr_res, ...)
```

Arguments

lr_res	ligand-receptor DE results and other stats between clusters
...	Arguments passed on to liana_call
	method name of the method to be called

Value

Returns a tibble with specificity weights (‘weight_sc‘) as calculated by Connectome

get_curated_omni *Function to Generate the Curated (Default) LIANA resource*

Description

Function to Generate the Curated (Default) LIANA resource

Usage

```
get_curated_omni(
 curated_resources = c("CellPhoneDB", "CellChatDB", "ICELLNET", "connectomeDB2020",
    "CellTalkDB")
)
```

Arguments

`curated_resources`
: the curated resources from which we wish to obtain interactions. By default, it includes interactions curated in the context of CCC from CellPhoneDB, CellChat, ICELLNET, connectomeDB, CellTalkDB, and SignaLink.

Details

Here, we define the curated resources as those that are defined as manually or expert curated in the context of cell-cell communication. Albeit, "Guide2Pharma", "HPMR", and "Kirouac2010" are also such resources the remainder of the resources used to generate Omnipath, use those as sources. Hence, we assume that the second round of manual curation done in subsequent, more recently published resources would already contain the high quality interactions of the aforementioned 3. We also omit Cellinker, as it results in a large mount of ambigous interactions, but one could consider adding it to the list of curated resources.

Value

a curated OmniPath resource formatted for LIANA

get_logfc *Function to obtain logFC weights*

Description

Function to obtain logFC weights

Usage

```
get_logfc(lr_res, ...)
```

Arguments

lr_res	ligand-receptor DE results and other stats between clusters
...	Arguments passed on to liana_call
	method name of the method to be called

Value

Returns a tibble with a logFC metric ('logfc_comb'). 'logfc_comb' is calculated as the product of the (1 vs the rest) log2FC for each ligand and receptor gene

---

get_lr_resources *Helper function that returns the name of each intercell resource in OmniPath*

Description

Helper function that returns the name of each intercell resource in OmniPath

Usage

```
get_lr_resources()
```

Value

A list of strings for each intercell resource in OmniPath

---

get_natmi *Function to obtain NATMI-like weights*

Description

Function to obtain NATMI-like weights

Usage

```
get_natmi(lr_res, ...)
```

Arguments

lr_res	ligand-receptor DE results and other stats between clusters
...	Arguments passed on to liana_call
	method name of the method to be called

Value

Returns a tibble with specificity weights ('edge_specificity') as calculated by NATMI

get_partners *Retrieves intercellular communication partners (ligands or receptors) from one ligand-receptor resource.*

Description

Retrieves intercellular communication partners (ligands or receptors) from one ligand-receptor resource.

Usage

```
get_partners(side, resource, ...)
```

Arguments

side	'ligand' (trans), 'receptor' (rec) or 'both' (both short or long notation can be used)
resource	Name of current resource (taken from get_lr_resources)
...	Inherit dot params from import_omnipath_intercell

get_permutations *Helper Function to generate shuffled means*

Description

Helper Function to generate shuffled means

Usage

```
get_permutations(
  lr_res,
  sce,
  nperms = 1000,
  seed = 1234,
  parallelize = FALSE,
  workers = 4,
  assay.type = "logcounts",
  verbose = TRUE
)
```

Arguments

lr_res	liana_pipe results
sce	SingleCellExperiment Object
nperms	number of permutations
seed	number used to set random seed
parallelize	logical whether to parallelize
workers	Number of workers to be used in parallelization
assay.type	- the type of data to be used to calculate the means (logcounts by default), available options are: "counts" and "logcounts"
verbose	logical for verbosity

Details

This function could be made generalizable to any set of genes, depending on the set (currently lr_res genes) that is used to filter - i.e. it could be replaced with e.g. genes from TF regulons

Value

Returns a list of shuffled gene means by cluster

---

get_sca *Function to obtain SingleCellSignalR-like scores*

Description

Function to obtain SingleCellSignalR-like scores

Usage

```
get_sca(lr_res, ...)
```

Arguments

lr_res	ligand-receptor DE results and other stats between clusters
...	Arguments passed on to liana_call method name of the method to be called

Value

Returns a tibble with specificity weights ('LRscore') as calculated by SingleCellSignalR

heat_freq *Communication Frequency heatmap plot*

Description

Communication Frequency heatmap plot

Usage

```
heat_freq(liana_res, ...)
```

Arguments

liana_res	aggregated liana results (preferably truncated to some threshold)
...	Arguments passed on to liana_heatmap
	mat Diagonal celltype-celltype matrix to be plotted. In theory, any metric deemed meaningful between cell pairs can be plotted.
	font_size base font_size - other fontsizes are relative to this one
	grid_text logical whether to display grid text or not
	name name of the heatmap. By default the heatmap name is used as the title of the heatmap legend.
	row_title Row tittle
	column_title Column tittle

Details

This plot was inspired by CellPhoneDB and also CellChat's heatmap design. It makes the assumption that the number of interactions inferred between cell types is informative of the communication events occurring in the system as a whole. This is a rather strong assumption limited by the arbitrarily filtered interactions Thus, I suggest that one limits any conclusions, unless supported by complimentary information, such as biological prior knowledge.

liana_aggregate *Function to Aggregate CCC Method Results*

Description

Function to Aggregate CCC Method Results

Usage

```
liana_aggregate(
  liana_res,
  aggregate_how = NULL,
  resource = NULL,
  set_cap = "max",
  cap = NULL,
  get_ranks = TRUE,
  get_agrank = TRUE,
  .score_mode = .score_specs,
  verbose = TRUE,
  join_cols = NULL,
  ...
)
```

Arguments

liana_res	LIANA results
aggregate_how	way to aggregate, by default (NULL) will aggregate all passed methods with the approach specified in 'liana:::.score_specs'. Alternative options are 'magnitude' and 'specificity'.
resource	If methods are ran with multiple resources, the name of the resource of interest needs to be provided *Note* if a name is not provided, the first results based on the first resource in the list will be returned
set_cap	Function used to set ranked cap (i.e. the value that is assigned to interactions with NA for scores); By default, this is set to "max", which is the maximum number of interactions obtained by the methods; Some methods return all possible ligand-receptor combinations for each possible source and target cell pair - i.e. the known universe of all possible interactions (based on the CCC resource)
cap	A cap can for all methods can also be manually set, then the top X interactions, based on the 'specificity' scores for each method will be returned and the ranking will be carried out solely on them
get_ranks	boolean, whether to return consensus ranks for methods
get_agrank	boolean, whether to return aggregate rank using the 'RobustRankAggreg' package.
.score_mode	defines the way that the methods would be aggragate. By default, we use the score of each method which reflects specificity (if available), if not e.g. the case of SCA we use it's sole scoring function. This aggregation is by default done on the basis of the list returns by '.score_mode'. Alternatively, one could pass '.score_housekeep' to obtain an aggragate of the housekeeping interactions of each method.
join_cols	columns by which different method results will be joined. NULL by default, and automatically will handle the columns depending on the methods used.
...	Additional parameters (currently unused, reserved for future extensions)

Details

set_cap is the name of the name of a function that is to be executed on a vector representing the number of rows in the results for each method, by default this is set to max, but any other function that works with vectors could be passed - e.g. min, mean, etc.

This function also decomplexifies any complex present in the CellChat results which returns complexes by default

Value

Tibble with the interaction results and ranking for each method

Examples

```
liana_path <- system.file(package = "liana")
# load testdata
testdata <- readRDS(file.path(liana_path, "testdata", "input", "testdata.rds"))
# run liana
liana_res <- liana_wrap(testdata, method = c("sca", "natmi"))
# aggregate results from multiple methods
liana_res <- liana_aggregate(liana_res)
```

---

liana_bysample *Wrapper around 'liana_wrap' to run liana for each sample.*

Description

Wrapper around 'liana_wrap' to run liana for each sample.

Usage

```
liana_bysample(
  sce,
  idents_col,
  sample_col,
  verbose = TRUE,
  inplace = TRUE,
  aggregate_how = NULL,
  ...
)
```

Arguments

idents_col	name of the cluster column
sample_col	name of the sample/context column
verbose	verbosity logical
inplace	logical (TRUE by default) if liana results are to be saved to the SingleCellExperiment object ('sce@metadata$liana_res')

aggregate_how if running multiple methods (default), then one cal also choose to aggregate the CCC results by sample.

... Arguments passed on to liana_wrap

sce 'SingleCellExperiment' object or 'SeuratObject'

method method(s) to be run via liana

resource resource(s) to be used by the methods ('Consensus' by default), Use 'all' to run all 'human' resources in one go), or 'custom' to run liana_wrap with an appropriately formatted custom resource, passed via 'exernal_resource'

external_resource external resource in OmniPath tibble format

min_cells minimum cell per cell identity to be considered for analysis

return_all whether to return all possible interactions. Any interaction with 'expr_prop' below the specific threshold will be assigned to the *worst* possible score in those that pass the threshold. For example, p-values from CellPhoneDB will be assigned to max(pvalue) - likely 1, and lr_means will be assigned to min(lr_means). Note that this applies only to the internal methods of liana.

supp_columns any supplementary/additional columns which are to be returned by liana. Possibilities include: c("ligand.expr", "receptor.expr" "ligand.stat", "receptor.stat", "ligand.pval", "receptor.pval", "ligand.FDR", "receptor.FDR", etc)

assay assay to be used by Seurat, by default set to 'NULL' and will use the DefaultAssay.

.simplify if methods are run with only 1 resource, return a list of tibbles for each method (default), rather than a list of lists with method-resource combinations

base Default to NULL (i.e. log2-transformation is assumed for SCE, and log-tranformation for Seurat). This is a requred step for the calculation of the logFC method - ensures that any other preprocessing of the counts is preserved. One could also pass 'NaN' if they wish to use the counts stored in the counts assay/slot, or any other number according to the base that was used for log-tranformation.

cell.adj cell adjacency tibble/dataframe /w weights by which we will 'multiply' the relevant columns. Any cell pairs with a weights of 0 will be filtered out. Note that if working with LIANA's default methods, we suggest weights >= 0 & =< 1. This ensure that all methods' score will be meaningfully weighed without changing the interpretation of their scores, thus allow one to filter SCA, rank NATMI, etc.

Details

takes a Seurat/SCE object and runs LIANA by sample/condition. The key by which the samples are separated is build from the 'condition_col' and 'sample_col', separated by the 'key_sep'.

Value

If inplace is true returns a sce object with 'liana_res' in 'sce@metadata', else it returns a named list of tibble with liana results per sample.

liana_call *Wrapper Function to obtain scores via liana_pipe*

Description

Wrapper Function to obtain scores via liana_pipe

Usage

```
liana_call(method, lr_res, ...)
```

Arguments

method	name of the method to be called
lr_res	ligand-receptor DE results and other stats between clusters
...	Arguments passed on to liana_pipe
	sce SingleCellExperiment Object
	op_resource resource tibble obtained via select_resource
	assay assay to be used ("RNA" by default)
	assay.type - the type of data to be used to calculate the means (logcounts by default), available options are: "counts" and "logcounts"
	verbose logical for verbosity
	cell.adj cell adjacency tibble/dataframe /w weights by which we will 'multiply' the relevant columns. Any cell pairs with a weights of 0 will be filtered out. Note that if working with LIANA's default methods, we suggest weights >= 0 & =< 1. This ensure that all methods' score will be meaningfully weighed without changing the interpretation of their scores, thus allow one to filter SCA, rank NATMI, etc.
	test.type String specifying the type of pairwise test to perform - a t-test with "t", a Wilcoxon rank sum test with "wilcox", or a binomial test with "binom".
	pval.type A string specifying how p-values are to be combined across pairwise comparisons for a given group/cluster.
	base base for conversion from log-tranformed ~CPM back to ~CPM.

Value

lr_res modified to be method-specific

liana_defaults *Function to pass Default Arguments for each method*

Description

Function to pass Default Arguments for each method

Usage

```
liana_defaults(
  assay = "RNA",
  assay.type = "logcounts",
  expr_prop = 0.1,
  seed = 1004,
  complex_policy = "mean0",
  parallelize = FALSE,
  workers = 8,
  permutation.params = NULL,
  liana_pipe.params = NULL,
  liana_call.params = NULL,
  cellphonedb.params = NULL,
  natmi.params = NULL,
  sca.params = NULL,
  connectome.params = NULL,
  cytotalk.params = NULL,
  logfc.params = NULL,
  cellchat.params = NULL,
  squidpy.params = NULL,
  call_sca.params = NULL,
  call_natmi.params = NULL,
  call_connectome.params = NULL,
  call_italk.params = NULL,
  ...
)
```

Arguments

assay	Assay name passed to ‘call_italk‘, ‘call_sca‘, ‘call_cellchat‘, and ‘call_connectome‘
assay.type	- the type of data to be used to calculate the means (logcounts by default), available options are: "counts" and "logcounts"
expr_prop	minimum proportion of gene expression per cell type (0.1 by default). Note that when working with complexes, the minimum subunit proportion will be used for filtering.
seed	random seed integer
complex_policy	policy how to account for the presence of complexes.

`parallelize` whether to parallelize cellphonedb-like

`workers` number of workers to be called

`permutation.params`
list of parameters passed to permutation methods

`liana_pipe.params`
list of Parameters passed to NATMI `liana_pipe`

`liana_call.params`
list of Parameters passed to NATMI `liana_call`

`cellphonedb.params`
list of Parameters passed to liana's internal cellphonedb implementation (cellphonedb_score)

`natmi.params` list of Parameters passed to liana's internal edge_specificity implementation (natmi_score)

`sca.params` list of Parameters passed to liana's internal LRScore implementation (sca_score)

`connectome.params`
list of Parameters passed to liana's internal connectome's weight_sc implementation (connectome_score)

`cytotalk.params`
list of Parameters passed to liana's internal crosstalk scores implementation (cytotalk_score)

`logfc.params` list of Parameters passed to liana's internal logFC implementation (logfc_score)

`cellchat.params`
list of Parameters passed to CellChat `call_cellchat`

`squidpy.params` list of Parameters passed to Squidpy `call_squidpy`

`call_sca.params`
list of Parameters passed to SingleCellSignalR `call_sca`

`call_natmi.params`
list of Parameters passed to NATMI `call_natmi`

`call_connectome.params`
list of Parameters passed to Connectome `call_connectome`

`call_italk.params`
list of Parameters passed to iTALK `call_italk`

Details

The default parameters for each method can also be overwritten by manually passing a list of parameters for the appropraite method `liana_wrap`

Further, each 'get_*' method will by default obtain the default params passed via `liana_pipe` and `liana_call`. This is done so that most steps required for the calculation of these methods are undertaken only once.

NB! LIANA's internal methods are made consistent. There is no reason to pass specific parameters to any of them. Thus, it is best that one sticks to the non-nested parameters of this function (i.e. excluding '.params'), unless a very specific reason requires any of LIANA's internal parameters to be changed.

Value

A list of the default parameters for each method

Examples

```
liana_path <- system.file(package = "liana")
# load testdata
testdata <- readRDS(file.path(liana_path, "testdata", "input", "testdata.rds"))
# get a `named` list with all default parameters passed to liana.
def_params <- liana_defaults()
# any of these can then be overwritten and passed to `...` in `liana_wrap`
# with the `.params` suffix to the parameter name type. For example,
liana_res <- liana_wrap(testdata,
  permutation.params = list(nperms = 2),
  liana_pipe.params = list(test.type = "wilcox")
)
```

---

liana_dotplot *Liana dotplot interactions by source and target cells*

Description

Liana dotplot interactions by source and target cells

Usage

```
liana_dotplot(
  liana_res,
  source_groups = NULL,
  target_groups = NULL,
  ntop = NULL,
  specificity = "natmi.edge_specificity",
  magnitude = "sca.LRscore",
  y.label = "Interactions (Ligand -> Receptor)",
  size.label = "Interaction\nSpecificity",
  colour.label = "Expression\nMagnitude",
  show_complex = TRUE,
  size_range = c(2, 10),
  invert_specificity = FALSE,
  invert_magnitude = FALSE,
  invert_function = function(x) -log10(x + 1e-10)
)
```

Arguments

liana_res	aggregated ‘liana_wrap‘ results from multiple methods, or alternatively results from running ‘liana_wrap‘ with a single method. Should be filtered by some condition (e.g. preferential consesus ranking, specific interactions, etc).

source_groups	names of the source (sender) cell types (NULL = no filter)
target_groups	names of the target cell types (NULL = no filter)
ntop	number of interactions to return. Note that this assumes that the tibble is sorted in descending order of interaction importance!
specificity	column to represent the dot-size of the interaction (by default 'natmi.edge_specificity')
magnitude	column to represent interactions expression magnitude (by default 'sca.LRscore')
y.label	y label name
size.label	size (~specificty) label name
colour.label	colour (~magnitude) label name
show_complex	logical whether to show complexes (default - TRUE) or only the subunit with minimum expression.

Details

Here, we refer to 'specificity' as how specific this interaction is to a cell type pair regards to the rest of the cell type pairs (e.g. CellPhoneDB's p-values, NATMI's specificity edges, Connectome's scaled weights, etc)

'magnitude' on the other hand is a direct measure of the expression alone, by default we use SingleCellSignalR's dataset indepent LRscore (bound between 0 and 1). Yet, one could also use CellChat's probabilities or CellPhoneDB's means, etc.

Value

a ggplot2 object

---

liana_heatmap *Communication by cell type Heatmap*

Description

Communication by cell type Heatmap

Usage

```
liana_heatmap(
  mat,
  font_size = 12,
  grid_text = FALSE,
  name = "Frequency",
  pallette = c("white", "violetred2"),
  row_title = "Sender (Cell types)",
  column_title = "Receiver (Cell types)",
  ...
)
```

Arguments

mat	Diagonal celltype-celltype matrix to be plotted. In theory, any metric deemed meaningful between cell pairs can be plotted.
font_size	base font_size - other fontsizes are relative to this one
grid_text	logical whether to display grid text or not
name	name of the heatmap. By default the heatmap name is used as the title of the heatmap legend.
row_title	Row tittle
column_title	Column tittle
...	parameters passed to 'ComplexHeatmap::Heatmap'

Details

Heatmap function inspired by CellPhoneDBv3 and CellChat's designs on communication heatmaps.

---

liana_message *LIANA message/warning helper function to allow for verbosity*

Description

LIANA message/warning helper function to allow for verbosity

Usage

```
liana_message(..., output = "message", verbose = TRUE)
```

Arguments

...	zero or more objects which can be coerced to character (and which are pasted together with no separator) or a single condition object.
output	type of output - message, warning, or stop
verbose	logical for verbosity

`liana_pipe` *Liana Pipe which runs DE analysis and merges needed information for LR inference*

Description

Liana Pipe which runs DE analysis and merges needed information for LR inference

Usage

```
liana_pipe(
  sce,
  op_resource,
  test.type = "wilcox",
  pval.type = "all",
  assay = "RNA",
  assay.type = "logcounts",
  verbose = TRUE,
  base,
  cell.adj = NULL
)
```

Arguments

`sce`	SingleCellExperiment Object
`op_resource`	resource tibble obtained via `select_resource`
`test.type`	String specifying the type of pairwise test to perform - a t-test with `"t"`, a Wilcoxon rank sum test with `"wilcox"`, or a binomial test with `"binom"`.
`pval.type`	A string specifying how p-values are to be combined across pairwise comparisons for a given group/cluster.
`assay`	assay to be used ("RNA" by default)
`assay.type`	- the type of data to be used to calculate the means (logcounts by default), available options are: "counts" and "logcounts"
`verbose`	logical for verbosity
`base`	base for conversion from log-tranformed ~CPM back to ~CPM.
`cell.adj`	cell adjacency tibble/dataframe /w weights by which we will 'multiply' the relevant columns. Any cell pairs with a weights of 0 will be filtered out. Note that if working with LIANA's default methods, we suggest weights >= 0 & =< 1. This ensure that all methods' score will be meaningfully weighed without changing the interpretation of their scores, thus allow one to filter SCA, rank NATMI, etc.

Value

Returns a tibble with information required for LR calculations downstream

liana_prep — *Function to handle different types of object as input and do basic quality checks*

Description

Function to handle different types of object as input and do basic quality checks

Usage

```
liana_prep(sce, ...)
```

Arguments

sce	SingleCellExperiment or Seurat object
...	dot dot dot bucket - not passed to anything, handles issues with passing non-existing arguments

liana_scores — *Function to obtain different scoring schemes*

Description

Function to obtain different scoring schemes

Usage

```
liana_scores(score_object, lr_res, ...)
```

Arguments

score_object	score_object specific to the test obtained from score_specs
lr_res	ligand-receptor DE results and other stats between clusters
...	dot params passed to '*_score' functions

Value

lr_res modified to be method-specific

liana_tensor_c2c *Wrapper function to run 'cell2cell_tensor' with LIANA output.*

Description

Wrapper function to run 'cell2cell_tensor' with LIANA output.

Usage

```
liana_tensor_c2c(
  sce = NULL,
  context_df_dict = NULL,
  score_col = "LRscore",
  how = "inner",
  lr_fill = NaN,
  cell_fill = NaN,
  lr_sep = "^",
  context_order = NULL,
  sort_elements = TRUE,
  device = NULL,
  rank = NULL,
  seed = 1337,
  upper_rank = 25,
  runs = 3,
  init = "svd",
  build_only = FALSE,
  factors_only = TRUE,
  conda_env = NULL,
  use_available = FALSE,
  verbose = TRUE,
  inplace = TRUE,
  sender_col = "source",
  receiver_col = "target",
  ligand_col = "ligand.complex",
  receptor_col = "receptor.complex",
  ...
)
```

Arguments

sce	SingleCellExperiment with 'context_df_dict' - i.e. liana results per context (sample) stored at 'sce@metadata$liana_res'
context_df_dict	Dictionary (named list) containing a dataframe for each context. The dataframe must contain columns containing sender (source) cells, receiver (target) cells, ligands, receptors, and communication scores, separately. Keys are context

	names and values are dataframes. NULL by default. If not NULL will be used instead of 'sce@metadata$liana_res'.
score_col	Name of the column containing the communication scores in all context dataframes.
how	Approach to consider cell types and genes present across multiple contexts. - 'inner' : Considers only cell types and LR pairs that are present in all contexts (intersection). - 'outer' : Considers all cell types and LR pairs that are present across contexts (union). - 'outer_lr' : Considers only cell types that are present in all contexts (intersection), while all LR pairs that are present across contexts (union). - 'outer_cells' : Considers only LR pairs that are present in all contexts (intersection), while all cell types that are present across contexts (union).
lr_fill	Value to fill communication scores when a ligand-receptor pair is not present across all contexts. (NaN by default)
cell_fill	Value to fill communication scores when a cell is not present across all ligand-receptor pairs or all contexts. (NaN by default)
lr_sep	Separation character to join ligands and receptors into a LR pair name. ('^' by Default)
context_order	List used to sort the contexts when building the tensor. Elements must be all elements in 'names(context_df_dict)'. (NULL by default)
sort_elements	Whether alphabetically sorting elements in the InteractionTensor. The Context Dimension is not sorted if a 'context_order' list is provided. (TRUE by default).
device	Device to use when backend is pytorch. Options are: ['cpu', 'cuda:0', None]. NULL by Default
rank	Ranks for the Tensor Factorization (number of factors to deconvolve the original tensor). If NULL, then rank selection is performed using the 'elbow_rank_selection' function.
seed	Random seed integer
upper_rank	Upper bound of ranks to explore with the elbow analysis.
runs	Number of tensor factorization performed for a given rank. Each factorization varies in the seed of initialization. Consider increasing the number of runs, in order to obtain a more robust rank estimate.
init	Initialization method for computing the Tensor Factorization. 'svd', 'random'
build_only	Whether to only return a tensor instance, without rank selection and no factorization.
factors_only	whether to return only the factors after factorization
conda_env	name of existing conda environment
use_available	whether to use c2c if available in current env. False by default.
verbose	verbosity logical
inplace	logical (TRUE by default) if liana results are to be saved to the SingleCellExperiment object ('sce@metadata$liana_res')
sender_col	Name of the column containing the sender cells in all context dataframes.
receiver_col	Name of the column containing the receiver cells in all context dataframes.
ligand_col	Name of the column containing the ligands in all context dataframes.

receptor_col	Name of the column containing the receptors in all context dataframes.
...	Dictionary containing keyword arguments for the c2c.compute_tensor_factorization function. The function deals with 'random_state' (seed) and 'rank' internally.

Details

This function servers as a one-liner wrapper to the tensor factorisation method described in tensor_cell2cell. We refer the user to the publication and tensor_cell2cell tutorial page made by the authors. Logically, one should cite cell2cell's paper if their method was used via LIANA.

Value

an instance of the cell2cell.tensor.BaseTensor class (via reticulate). If build_only is TRUE, then no rank selection or tensor decomposition is returned. Otherwise, returns a tensor with factorization results.

---

liana_wrap *LIANA wrapper function*

Description

LIANA wrapper function

Usage

```
liana_wrap(
  sce,
  method = c("natmi", "connectome", "logfc", "sca", "cellphonedb"),
  resource = c("Consensus"),
  idents_col = NULL,
  external_resource,
  min_cells = 5,
  return_all = FALSE,
  supp_columns = NULL,
  verbose = TRUE,
  assay = NULL,
  .simplify = TRUE,
  cell.adj = NULL,
  base = NULL,
  layer = NULL,
  ...
)
```

Arguments

sce	'SingleCellExperiment' object or 'SeuratObject'
method	method(s) to be run via liana
resource	resource(s) to be used by the methods ('Consensus' by default), Use 'all' to run all 'human' resources in one go), or 'custom' to run liana_wrap with an appropriately formatted custom resource, passed via 'exernal_resource'
idents_col	the cell identities/labels to be used. By default set to NULL, and will used the active Idents or colLabels from the seurat_object or SCE, respectively.
external_resource	external resource in OmniPath tibble format
min_cells	minimum cell per cell identity to be considered for analysis
return_all	whether to return all possible interactions. Any interaction with 'expr_prop' below the specific threshold will be assigned to the *worst* possible score in those that pass the threshold. For example, p-values from CellPhoneDB will be assigned to max(pvalue) - likely 1, and lr_means will be assigned to min(lr_means). Note that this applies only to the internal methods of liana.
supp_columns	any supplementary/additional columns which are to be returned by liana. Possibilities include: c("ligand.expr", "receptor.expr" "ligand.stat", "receptor.stat", "ligand.pval", "receptor.pval", "ligand.FDR", "receptor.FDR", etc)
verbose	logical whether to output messages and warnings ('TRUE' by default)
assay	assay to be used by Seurat, by default set to 'NULL' and will use the DefaultAssay.
.simplify	if methods are run with only 1 resource, return a list of tibbles for each method (default), rather than a list of lists with method-resource combinations
cell.adj	cell adjacency tibble/dataframe /w weights by which we will 'multiply' the relevant columns. Any cell pairs with a weights of 0 will be filtered out. Note that if working with LIANA's default methods, we suggest weights >= 0 & =< 1. This ensure that all methods' score will be meaningfully weighed without changing the interpretation of their scores, thus allow one to filter SCA, rank NATMI, etc.
base	Default to NULL (i.e. log2-transformation is assumed for SCE, and log-tranformation for Seurat). This is a requred step for the calculation of the logFC method - ensures that any other preprocessing of the counts is preserved. One could also pass 'NaN' if they wish to use the counts stored in the counts assay/slot, or any other number according to the base that was used for log-tranformation.
...	Arguments passed on to liana_defaults expr_prop minimum proportion of gene expression per cell type (0.1 by default). Note that when working with complexes, the minimum subunit proportion will be used for filtering. complex_policy policy how to account for the presence of complexes. seed random seed integer parallelize whether to parallelize cellphonedb-like workers number of workers to be called liana_pipe.params list of Parameters passed to NATMI liana_pipe

`liana_call.params` list of Parameters passed to NATMI `liana_call`

`cellchat.params` list of Parameters passed to CellChat `call_cellchat`

`squidpy.params` list of Parameters passed to Squidpy `call_squidpy`

`call_connectome.params` list of Parameters passed to Connectome `call_connectome`

`call_italk.params` list of Parameters passed to iTALK `call_italk`

`call_natmi.params` list of Parameters passed to NATMI `call_natmi`

`call_sca.params` list of Parameters passed to SingleCellSignalR `call_sca`

`cellphonedb.params` list of Parameters passed to liana's internal cellphonedb implementation (cellphonedb_score)

`natmi.params` list of Parameters passed to liana's internal edge_specificity implementation (natmi_score)

`sca.params` list of Parameters passed to liana's internal LRScore implementation (sca_score)

`connectome.params` list of Parameters passed to liana's internal connectome's weight_sc implementation (connectome_score)

`cytotalk.params` list of Parameters passed to liana's internal crosstalk scores implementation (cytotalk_score)

`logfc.params` list of Parameters passed to liana's internal logFC implementation (logfc_score)

`permutation.params` list of parameters passed to permutation methods

`assay.type` - the type of data to be used to calculate the means (logcounts by default), available options are: "counts" and "logcounts"

Details

LIANA wrapper method that can be used to call each method with a given set of intercellular resources from the OmniPath universe. Please see 'liana_defaults()' for more information about the default parameters passed used by 'liana_wrap', if you wish to modify them.

Value

A list of method-resource results - i.e. provided resources are run with each method If only one resource is selected, a single tibble (with results for that resource) will be returned for each of the selected methods

Examples

```
liana_path <- system.file(package = "liana")
# load testdata
testdata <- readRDS(file.path(liana_path, "testdata", "input", "testdata.rds"))
# run only 2 methods from liana
liana_res <- liana_wrap(testdata,
  method = c("cellphonedb", "sca"),
  resource = "Consensus", # default resource
  # example run with *only* 2 permutations for cpdb
  permutation.params = list(nperms = 2)
)
```

min0 *Helper Function which returns the value closest to 0*

Description

Helper Function which returns the value closest to 0

Usage

```
min0(vec)
```

Arguments

vec	numeric vector

Value

value closest to 0

minmax *Helper min-max normalization function*

Description

Helper min-max normalization function

Usage

```
minmax(x, ...)
```

Arguments

x	numeric vector
...	Arguments passed on to base::max

Value

normalized vector scaled to [0, 1]

plot_c2c_cells *Plot the product of loadings between the source and target loadings within a factor*

Description

Plot the product of loadings between the source and target loadings within a factor

Usage

```
plot_c2c_cells(sce, factor_of_int, ...)
```

Arguments

factor_of_int	factor of interest e.g. Factor.8
...	Arguments passed on to ComplexHeatmap::Heatmap

matrix A matrix. Either numeric or character. If it is a simple vector, it will be converted to a one-column matrix.

col A vector of colors if the color mapping is discrete or a color mapping function if the matrix is continuous numbers (should be generated by colorRamp2). If the matrix is continuous, the value can also be a vector of colors so that colors can be interpolated. Pass to ColorMapping. For more details and examples, please refer to https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#colors .

name Name of the heatmap. By default the heatmap name is used as the title of the heatmap legend.

na_col Color for NA values.

rect_gp Graphic parameters for drawing rectangles (for heatmap body). The value should be specified by gpar and fill parameter is ignored.

color_space The color space in which colors are interpolated. Only used if matrix is numeric and col is a vector of colors. Pass to colorRamp2.

border Whether draw border. The value can be logical or a string of color.

border_gp Graphic parameters for the borders. If you want to set different parameters for different heatmap slices, please consider to use decorate_heatmap_body.

cell_fun Self-defined function to add graphics on each cell. Seven parameters will be passed into this function: j, i, x, y, width, height, fill which are column index, row index in matrix, coordinate of the cell, the width and height of the cell and the filled color. x, y, width and height are all unit objects.

layer_fun Similar as cell_fun, but is vectorized. Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#customize-the-heatmap-body .

jitter Random shifts added to the matrix. The value can be logical or a single numeric value. It it is TRUE, random values from uniform distribution between 0 and 1e-10 are generated. If it is a numeric value, the range for the

uniform distribution is (0, jitter). It is mainly to solve the problem of "Error: node stack overflow" when there are too many identical rows/columns for plotting the dendrograms. ADD: From version 2.5.6, the error of node stack overflow has been fixed, now this argument is ignored.

row_title Title on the row.

row_title_side Will the title be put on the left or right of the heatmap?

row_title_gp Graphic parameters for row title.

row_title_rot Rotation of row title.

column_title Title on the column.

column_title_side Will the title be put on the top or bottom of the heatmap?

column_title_gp Graphic parameters for column title.

column_title_rot Rotation of column titles.

cluster_rows If the value is a logical, it controls whether to make cluster on rows. The value can also be a hclust or a dendrogram which already contains clustering. Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#clustering .

cluster_row_slices If rows are split into slices, whether perform clustering on the slice means?

clustering_distance_rows It can be a pre-defined character which is in ("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "spearman", "kendall"). It can also be a function. If the function has one argument, the input argument should be a matrix and the returned value should be a dist object. If the function has two arguments, the input arguments are two vectors and the function calculates distance between these two vectors.

clustering_method_rows Method to perform hierarchical clustering, pass to hclust.

row_dend_side Should the row dendrogram be put on the left or right of the heatmap?

row_dend_width Width of the row dendrogram, should be a unit object.

show_row_dend Whether show row dendrogram?

row_dend_gp Graphic parameters for the dendrogram segments. If users already provide a dendrogram object with edges rendered, this argument will be ignored.

row_dend_reorder Apply reordering on row dendrograms. The value can be a logical value or a vector which contains weight which is used to reorder rows. The reordering is applied by reorder.dendrogram.

cluster_columns Whether make cluster on columns? Same settings as cluster_rows.

cluster_column_slices If columns are split into slices, whether perform clustering on the slice means?

clustering_distance_columns Same setting as clustering_distance_rows.

clustering_method_columns Method to perform hierarchical clustering, pass to hclust.

column_dend_side Should the column dendrogram be put on the top or bottom of the heatmap?

column_dend_height height of the column cluster, should be a unit object.

show_column_dend Whether show column dendrogram?

column_dend_gp Graphic parameters for dendrogram segments. Same settings as row_dend_gp.

column_dend_reorder Apply reordering on column dendrograms. Same settings as row_dend_reorder.

row_order Order of rows. Manually setting row order turns off clustering.

column_order Order of column.

row_labels Optional row labels which are put as row names in the heatmap.

row_names_side Should the row names be put on the left or right of the heatmap?

show_row_names Whether show row names.

row_names_max_width Maximum width of row names viewport.

row_names_gp Graphic parameters for row names.

row_names_rot Rotation of row names.

row_names_centered Should row names put centered?

column_labels Optional column labels which are put as column names in the heatmap.

column_names_side Should the column names be put on the top or bottom of the heatmap?

column_names_max_height Maximum height of column names viewport.

show_column_names Whether show column names.

column_names_gp Graphic parameters for drawing text.

column_names_rot Rotation of column names.

column_names_centered Should column names put centered?

top_annotation A HeatmapAnnotation object.

bottom_annotation A HeatmapAnnotation object.

left_annotation It should be specified by rowAnnotation.

right_annotation it should be specified by rowAnnotation.

km Apply k-means clustering on rows. If the value is larger than 1, the heatmap will be split by rows according to the k-means clustering. For each row slice, hierarchical clustering is still applied with parameters above.

split A vector or a data frame by which the rows are split. But if cluster_rows is a clustering object, split can be a single number indicating to split the dendrogram by cutree.

row_km Same as km.

row_km_repeats Number of k-means runs to get a consensus k-means clustering. Note if row_km_repeats is set to more than one, the final number of groups might be smaller than row_km, but this might means the original row_km is not a good choice.

row_split Same as split.

column_km K-means clustering on columns.

column_km_repeats Number of k-means runs to get a consensus k-means clustering. Similar as row_km_repeats.

column_split Split on columns. For heatmap splitting, please refer to https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-split .

gap Gap between row slices if the heatmap is split by rows. The value should be a unit object.

row_gap Same as gap.

column_gap Gap between column slices.

show_parent_dend_line When heatmap is split, whether to add a dashed line to mark parent dendrogram and children dendrograms?

width Width of the heatmap body.

height Height of the heatmap body.

heatmap_width Width of the whole heatmap (including heatmap components)

heatmap_height Height of the whole heatmap (including heatmap components). Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#size-of-the-heatmap .

show_heatmap_legend Whether show heatmap legend?

heatmap_legend_param A list contains parameters for the heatmap legends. See color_mapping_legend,ColorMapping-method for all available parameters.

use_raster Whether render the heatmap body as a raster image. It helps to reduce file size when the matrix is huge. If number of rows or columns is more than 2000, it is by default turned on. Note if cell_fun is set, use_raster is enforced to be FALSE.

raster_device Graphic device which is used to generate the raster image.

raster_quality A value larger than 1.

raster_device_param A list of further parameters for the selected graphic device. For raster image support, please check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-as-raster-image .

raster_resize_mat Whether resize the matrix to let the dimension of the matrix the same as the dimension of the raster image? The value can be logical. If it is TRUE, mean is used to summarize the sub matrix which corresponds to a single pixel. The value can also be a summary function, e.g. max.

raster_by_magick Whether to use image_resize to scale the image.

raster_magick_filter Pass to filter argument of image_resize. A character scalar and all possible values are in filter_types. The default is "Lanczos".

post_fun A function which will be executed after the heatmap list is drawn.

factors factors as formatted by ‘format_c2c_factors‘

plot_c2c_overview *Function to plot an Overview of tensor-c2c results*

Description

Function to plot an Overview of tensor-c2c results

Usage

```
plot_c2c_overview(sce, group_col, sample_col)
```

Arguments

sce	SingleCellExperiment with factors output from tensor-cell2cell
group_col	context descriptor - to be obtained from ‘colData(sce)‘
sample_col	context/sample names - obtained from ‘colData(sce)‘

plot_context_boxplot *Generate boxplots with significance*

Description

Generate boxplots with significance

Usage

```
plot_context_boxplot(sce, sample_col, group_col, test = "t.test", ...)
```

Arguments

sce	SingleCellExperiment with factors output from tensor-cell2cell
sample_col	context/sample names - obtained from ‘colData(sce)‘
group_col	context descriptor - to be obtained from ‘colData(sce)‘
...	arguments passed to the test used.

`plot_context_heat` *Plot a Heatmap of context loadings*

Description

Plot a Heatmap of context loadings

Usage

```
plot_context_heat(sce, sample_col, group_col, ...)
```

Arguments

`sce`	SingleCellExperiment with factors output from tensor-cell2cell
`sample_col`	context/sample names - obtained from 'colData(sce)'
`group_col`	context descriptor - to be obtained from 'colData(sce)'
`...`	Arguments passed on to `ComplexHeatmap::Heatmap`

`matrix` A matrix. Either numeric or character. If it is a simple vector, it will be converted to a one-column matrix.

`col` A vector of colors if the color mapping is discrete or a color mapping function if the matrix is continuous numbers (should be generated by `colorRamp2`). If the matrix is continuous, the value can also be a vector of colors so that colors can be interpolated. Pass to `ColorMapping`. For more details and examples, please refer to https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#colors .

`name` Name of the heatmap. By default the heatmap name is used as the title of the heatmap legend.

`na_col` Color for `NA` values.

`rect_gp` Graphic parameters for drawing rectangles (for heatmap body). The value should be specified by `gpar` and `fill` parameter is ignored.

`color_space` The color space in which colors are interpolated. Only used if `matrix` is numeric and `col` is a vector of colors. Pass to `colorRamp2`.

`border` Whether draw border. The value can be logical or a string of color.

`border_gp` Graphic parameters for the borders. If you want to set different parameters for different heatmap slices, please consider to use `decorate_heatmap_body`.

`cell_fun` Self-defined function to add graphics on each cell. Seven parameters will be passed into this function: `j`, `i`, `x`, `y`, `width`, `height`, `fill` which are column index, row index in `matrix`, coordinate of the cell, the width and height of the cell and the filled color. `x`, `y`, `width` and `height` are all `unit` objects.

`layer_fun` Similar as `cell_fun`, but is vectorized. Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#customize-the-heatmap-body .

`jitter` Random shifts added to the matrix. The value can be logical or a single numeric value. It it is `TRUE`, random values from uniform distribution between 0 and 1e-10 are generated. If it is a numeric value, the range for the uniform distribution is (0, `jitter`). It is mainly to solve the problem of "Error: node stack overflow" when there are too many identical rows/columns for plotting the dendrograms. ADD: From version 2.5.6, the error of node stack overflow has been fixed, now this argument is ignored.

`row_title` Title on the row.

`row_title_side` Will the title be put on the left or right of the heatmap?

`row_title_gp` Graphic parameters for row title.

`row_title_rot` Rotation of row title.

`column_title` Title on the column.

`column_title_side` Will the title be put on the top or bottom of the heatmap?

`column_title_gp` Graphic parameters for column title.

`column_title_rot` Rotation of column titles.

`cluster_rows` If the value is a logical, it controls whether to make cluster on rows. The value can also be a `hclust` or a `dendrogram` which already contains clustering. Check `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#clustering` .

`cluster_row_slices` If rows are split into slices, whether perform clustering on the slice means?

`clustering_distance_rows` It can be a pre-defined character which is in ("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "spearman", "kendall"). It can also be a function. If the function has one argument, the input argument should be a matrix and the returned value should be a `dist` object. If the function has two arguments, the input arguments are two vectors and the function calculates distance between these two vectors.

`clustering_method_rows` Method to perform hierarchical clustering, pass to `hclust`.

`row_dend_side` Should the row dendrogram be put on the left or right of the heatmap?

`row_dend_width` Width of the row dendrogram, should be a `unit` object.

`show_row_dend` Whether show row dendrogram?

`row_dend_gp` Graphic parameters for the dendrogram segments. If users already provide a `dendrogram` object with edges rendered, this argument will be ignored.

`row_dend_reorder` Apply reordering on row dendrograms. The value can be a logical value or a vector which contains weight which is used to reorder rows. The reordering is applied by `reorder.dendrogram`.

`cluster_columns` Whether make cluster on columns? Same settings as `cluster_rows`.

`cluster_column_slices` If columns are split into slices, whether perform clustering on the slice means?

`clustering_distance_columns` Same setting as `clustering_distance_rows`.

`clustering_method_columns` Method to perform hierarchical clustering, pass to `hclust`.

column_dend_side Should the column dendrogram be put on the top or bottom of the heatmap?

column_dend_height height of the column cluster, should be a unit object.

show_column_dend Whether show column dendrogram?

column_dend_gp Graphic parameters for dendrogram segments. Same settings as row_dend_gp.

column_dend_reorder Apply reordering on column dendrograms. Same settings as row_dend_reorder.

row_order Order of rows. Manually setting row order turns off clustering.

column_order Order of column.

row_labels Optional row labels which are put as row names in the heatmap.

row_names_side Should the row names be put on the left or right of the heatmap?

show_row_names Whether show row names.

row_names_max_width Maximum width of row names viewport.

row_names_gp Graphic parameters for row names.

row_names_rot Rotation of row names.

row_names_centered Should row names put centered?

column_labels Optional column labels which are put as column names in the heatmap.

column_names_side Should the column names be put on the top or bottom of the heatmap?

column_names_max_height Maximum height of column names viewport.

show_column_names Whether show column names.

column_names_gp Graphic parameters for drawing text.

column_names_rot Rotation of column names.

column_names_centered Should column names put centered?

top_annotation A HeatmapAnnotation object.

bottom_annotation A HeatmapAnnotation object.

left_annotation It should be specified by rowAnnotation.

right_annotation it should be specified by rowAnnotation.

km Apply k-means clustering on rows. If the value is larger than 1, the heatmap will be split by rows according to the k-means clustering. For each row slice, hierarchical clustering is still applied with parameters above.

split A vector or a data frame by which the rows are split. But if cluster_rows is a clustering object, split can be a single number indicating to split the dendrogram by cutree.

row_km Same as km.

row_km_repeats Number of k-means runs to get a consensus k-means clustering. Note if row_km_repeats is set to more than one, the final number of groups might be smaller than row_km, but this might means the original row_km is not a good choice.

row_split Same as split.

column_km K-means clustering on columns.

column_km_repeats Number of k-means runs to get a consensus k-means clustering. Similar as row_km_repeats.

column_split Split on columns. For heatmap splitting, please refer to https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-split .

gap Gap between row slices if the heatmap is split by rows. The value should be a unit object.

row_gap Same as gap.

column_gap Gap between column slices.

show_parent_dend_line When heatmap is split, whether to add a dashed line to mark parent dendrogram and children dendrograms?

width Width of the heatmap body.

height Height of the heatmap body.

heatmap_width Width of the whole heatmap (including heatmap components)

heatmap_height Height of the whole heatmap (including heatmap components). Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#size-of-the-heatmap .

show_heatmap_legend Whether show heatmap legend?

heatmap_legend_param A list contains parameters for the heatmap legends. See color_mapping_legend,ColorMapping-method for all available parameters.

use_raster Whether render the heatmap body as a raster image. It helps to reduce file size when the matrix is huge. If number of rows or columns is more than 2000, it is by default turned on. Note if cell_fun is set, use_raster is enforced to be FALSE.

raster_device Graphic device which is used to generate the raster image.

raster_quality A value larger than 1.

raster_device_param A list of further parameters for the selected graphic device. For raster image support, please check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-as-raster-image .

raster_resize_mat Whether resize the matrix to let the dimension of the matrix the same as the dimension of the raster image? The value can be logical. If it is TRUE, mean is used to summarize the sub matrix which corresponds to a single pixel. The value can also be a summary function, e.g. max.

raster_by_magick Whether to use image_resize to scale the image.

raster_magick_filter Pass to filter argument of image_resize. A character scalar and all possible values are in filter_types. The default is "Lanczos".

post_fun A function which will be executed after the heatmap list is drawn.

Value

a ComplexHeatmap object

plot_lr_heatmap *Function to plot a UMAP of context loadings*

Description

Function to plot a UMAP of context loadings

Usage

```
plot_lr_heatmap(sce, lr_sep = "^", n = 5, ...)
```

Arguments

sce	SingleCellExperiment with factors output from tensor-cell2cell
n	Number of rows to return for top_n(), fraction of rows to return for top_frac(). If n is positive, selects the top rows. If negative, selects the bottom rows. If x is grouped, this is the number (or fraction) of rows per group. Will include more rows if there are ties.
...	Arguments passed on to ComplexHeatmap::Heatmap matrix A matrix. Either numeric or character. If it is a simple vector, it will be converted to a one-column matrix. col A vector of colors if the color mapping is discrete or a color mapping function if the matrix is continuous numbers (should be generated by colorRamp2). If the matrix is continuous, the value can also be a vector of colors so that colors can be interpolated. Pass to ColorMapping. For more details and examples, please refer to https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#colors . name Name of the heatmap. By default the heatmap name is used as the title of the heatmap legend. na_col Color for NA values. rect_gp Graphic parameters for drawing rectangles (for heatmap body). The value should be specified by gpar and fill parameter is ignored. color_space The color space in which colors are interpolated. Only used if matrix is numeric and col is a vector of colors. Pass to colorRamp2. border Whether draw border. The value can be logical or a string of color. border_gp Graphic parameters for the borders. If you want to set different parameters for different heatmap slices, please consider to use decorate_heatmap_body. cell_fun Self-defined function to add graphics on each cell. Seven parameters will be passed into this function: j, i, x, y, width, height, fill which are column index, row index in matrix, coordinate of the cell, the width and height of the cell and the filled color. x, y, width and height are all unit objects. layer_fun Similar as cell_fun, but is vectorized. Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#customize-the-heatmap-body .

`jitter` Random shifts added to the matrix. The value can be logical or a single numeric value. It it is `TRUE`, random values from uniform distribution between 0 and 1e-10 are generated. If it is a numeric value, the range for the uniform distribution is (0, `jitter`). It is mainly to solve the problem of "Error: node stack overflow" when there are too many identical rows/columns for plotting the dendrograms. ADD: From version 2.5.6, the error of node stack overflow has been fixed, now this argument is ignored.

`row_title` Title on the row.

`row_title_side` Will the title be put on the left or right of the heatmap?

`row_title_gp` Graphic parameters for row title.

`row_title_rot` Rotation of row title.

`column_title` Title on the column.

`column_title_side` Will the title be put on the top or bottom of the heatmap?

`column_title_gp` Graphic parameters for column title.

`column_title_rot` Rotation of column titles.

`cluster_rows` If the value is a logical, it controls whether to make cluster on rows. The value can also be a `hclust` or a `dendrogram` which already contains clustering. Check `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#clustering` .

`cluster_row_slices` If rows are split into slices, whether perform clustering on the slice means?

`clustering_distance_rows` It can be a pre-defined character which is in ("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "spearman", "kendall"). It can also be a function. If the function has one argument, the input argument should be a matrix and the returned value should be a `dist` object. If the function has two arguments, the input arguments are two vectors and the function calculates distance between these two vectors.

`clustering_method_rows` Method to perform hierarchical clustering, pass to `hclust`.

`row_dend_side` Should the row dendrogram be put on the left or right of the heatmap?

`row_dend_width` Width of the row dendrogram, should be a `unit` object.

`show_row_dend` Whether show row dendrogram?

`row_dend_gp` Graphic parameters for the dendrogram segments. If users already provide a `dendrogram` object with edges rendered, this argument will be ignored.

`row_dend_reorder` Apply reordering on row dendrograms. The value can be a logical value or a vector which contains weight which is used to reorder rows. The reordering is applied by `reorder.dendrogram`.

`cluster_columns` Whether make cluster on columns? Same settings as `cluster_rows`.

`cluster_column_slices` If columns are split into slices, whether perform clustering on the slice means?

`clustering_distance_columns` Same setting as `clustering_distance_rows`.

`clustering_method_columns` Method to perform hierarchical clustering, pass to `hclust`.

column_dend_side Should the column dendrogram be put on the top or bottom of the heatmap?

column_dend_height height of the column cluster, should be a unit object.

show_column_dend Whether show column dendrogram?

column_dend_gp Graphic parameters for dendrogram segments. Same settings as row_dend_gp.

column_dend_reorder Apply reordering on column dendrograms. Same settings as row_dend_reorder.

row_order Order of rows. Manually setting row order turns off clustering.

column_order Order of column.

row_labels Optional row labels which are put as row names in the heatmap.

row_names_side Should the row names be put on the left or right of the heatmap?

show_row_names Whether show row names.

row_names_max_width Maximum width of row names viewport.

row_names_gp Graphic parameters for row names.

row_names_rot Rotation of row names.

row_names_centered Should row names put centered?

column_labels Optional column labels which are put as column names in the heatmap.

column_names_side Should the column names be put on the top or bottom of the heatmap?

column_names_max_height Maximum height of column names viewport.

show_column_names Whether show column names.

column_names_gp Graphic parameters for drawing text.

column_names_rot Rotation of column names.

column_names_centered Should column names put centered?

top_annotation A HeatmapAnnotation object.

bottom_annotation A HeatmapAnnotation object.

left_annotation It should be specified by rowAnnotation.

right_annotation it should be specified by rowAnnotation.

km Apply k-means clustering on rows. If the value is larger than 1, the heatmap will be split by rows according to the k-means clustering. For each row slice, hierarchical clustering is still applied with parameters above.

split A vector or a data frame by which the rows are split. But if cluster_rows is a clustering object, split can be a single number indicating to split the dendrogram by cutree.

row_km Same as km.

row_km_repeats Number of k-means runs to get a consensus k-means clustering. Note if row_km_repeats is set to more than one, the final number of groups might be smaller than row_km, but this might means the original row_km is not a good choice.

row_split Same as split.

column_km K-means clustering on columns.

column_km_repeats Number of k-means runs to get a consensus k-means clustering. Similar as row_km_repeats.

column_split Split on columns. For heatmap splitting, please refer to https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-split .

gap Gap between row slices if the heatmap is split by rows. The value should be a unit object.

row_gap Same as gap.

column_gap Gap between column slices.

show_parent_dend_line When heatmap is split, whether to add a dashed line to mark parent dendrogram and children dendrograms?

width Width of the heatmap body.

height Height of the heatmap body.

heatmap_width Width of the whole heatmap (including heatmap components)

heatmap_height Height of the whole heatmap (including heatmap components). Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#size-of-the-heatmap .

show_heatmap_legend Whether show heatmap legend?

heatmap_legend_param A list contains parameters for the heatmap legends. See color_mapping_legend,ColorMapping-method for all available parameters.

use_raster Whether render the heatmap body as a raster image. It helps to reduce file size when the matrix is huge. If number of rows or columns is more than 2000, it is by default turned on. Note if cell_fun is set, use_raster is enforced to be FALSE.

raster_device Graphic device which is used to generate the raster image.

raster_quality A value larger than 1.

raster_device_param A list of further parameters for the selected graphic device. For raster image support, please check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-as-raster-image .

raster_resize_mat Whether resize the matrix to let the dimension of the matrix the same as the dimension of the raster image? The value can be logical. If it is TRUE, mean is used to summarize the sub matrix which corresponds to a single pixel. The value can also be a summary function, e.g. max.

raster_by_magick Whether to use image_resize to scale the image.

raster_magick_filter Pass to filter argument of image_resize. A character scalar and all possible values are in filter_types. The default is "Lanczos".

post_fun A function which will be executed after the heatmap list is drawn.

Value

a ComplexHeatmap object

`preprocess_scores` *Helper function to deal with tensor sparsity and liana's scores as in Python*

Description

Helper function to deal with tensor sparsity and liana's scores as in Python

Usage

```
preprocess_scores(
  context_df_dict,
  score_col = "magnitude_rank",
  sender_col = "source",
  receiver_col = "target",
  ligand_col = "ligand.complex",
  receptor_col = "receptor.complex",
  outer_fraction = 0,
  invert = TRUE,
  invert_fun = function(x) 1 - x,
  non_negative = TRUE,
  non_negative_fill = 0,
  lr_sep = "^",
  verbose = TRUE
)
```

Arguments

`context_df_dict`	Dictionary (named list) containing a dataframe for each context. The dataframe must contain columns containing sender (source) cells, receiver (target) cells, ligands, receptors, and communication scores, separately. Keys are context names and values are dataframes. NULL by default. If not NULL will be used instead of 'sce@metadata$liana_res'.
`score_col`	Name of the column containing the communication scores in all context dataframes.
`sender_col`	Name of the column containing the sender cells in all context dataframes.
`receiver_col`	Name of the column containing the receiver cells in all context dataframes.
`ligand_col`	Name of the column containing the ligands in all context dataframes.
`receptor_col`	Name of the column containing the receptors in all context dataframes.
`outer_fraction`	controls the elements to include in the union scenario of the 'how' options. Only elements that are present at least in this fraction of samples/contexts will be included. When this value is 0, considers all elements across the samples. When this value is 1, it acts as using 'how='inner''
`invert`	boolean wheter to invert the score (TRUE by defeault)
`invert_fun`	function used to invert scores

non_negative	whether to set negative scores to 0
non_negative_fill	the value to be used to fill negative values
lr_sep	ligand-receptor separator; '^' by default.
verbose	verbosity logical

`rank_aggregate` *Aggregate CCC Method results and by both magnitude and specificity ranks*

Description

Aggregate CCC Method results and by both magnitude and specificity ranks

Usage

```
rank_aggregate(liana_res, ...)
```

Arguments

liana_res	LIANA results
...	Arguments passed on to liana_aggregate

aggregate_how way to aggregate, by default (NULL) will aggregate all passed methods with the approach specified in 'liana:::.score_specs'. Alternative options are 'magnitude' and 'specificity'.

set_cap Function used to set ranked cap (i.e. the value that is assigned to interactions with NA for scores); By default, this is set to "max", which is the maximum number of interactions obtained by the methods; Some methods return all possible ligand-receptor combinations for each possible source and target cell pair - i.e. the known universe of all possible interactions (based on the CCC resource)

resource If methods are ran with multiple resources, the name of the resource of interest needs to be provided *Note* if a name is not provided, the first results based on the first resource in the list will be returned

cap A cap can for all methods can also be manually set, then the top X interactions, based on the 'specificity' scores for each method will be returned and the ranking will be carried out solely on them

get_ranks boolean, whether to return consensus ranks for methods

get_agrank boolean, whether to return aggregate rank using the 'RobustRankAggreg' package.

.score_mode defines the way that the methods would be aggragate. By default, we use the score of each method which reflects specificity (if available), if not e.g. the case of SCA we use it's sole scoring function. This aggregation is by default done on the basis of the list returns by '.score_mode'. Alternatively, one could pass '.score_housekeep' to obtain an aggragate of the housekeeping interactions of each method.

join_cols columns by which different method results will be joined. NULL by default, and automatically will handle the columns depending on the methods used.

Examples

```
liana_path <- system.file(package = "liana")
# load testdata
testdata <- readRDS(file.path(liana_path, "testdata", "input", "testdata.rds"))
# run liana
liana_res <- liana_wrap(testdata, method = c("sca", "natmi"))
# aggregate results from multiple methods
liana_res <- rank_aggregate(liana_res)
```

---

rank_method *Helper function to rank each method*

Description

Helper function to rank each method

Usage

```
rank_method(liana_res, method_name, mode = "specificity")
```

Arguments

liana_res	liana_results for a single method
method_name	name of the method
mode	ranking to be carried out. Accepted modes are ‘specificity‘ and ‘magnitude‘. The first is meant to reflect the specificity of interactions across all cell types, while the latter typically reflects how highly expressed is a given interaction.

Details

this function makes use of liana’s ‘liana:::.score_specs‘ and ‘liana:::.score_housekeep‘ functions.

select_resource *Helper Function to Handle resource choices*

Description

Helper Function to Handle resource choices

Usage

```
select_resource(resource)
```

Arguments

resource	names of the resources. Passing 'all' will return all human resources (i.e. all resources, except MouseConsensus)

Details

This function simply reads omni_resources.rds and returns the resources. Any of the resources can also be obtained via the same file. or the 'compile_ligrec' function, which querries and assmelbes the resources via 'OmniPathR'.

'Default' - The Default (inbuilt) resource for each of the methods; if using the 'call_*' functions, the default resource is used by passing *NULL* to the resource parameter. 'Reshuffled' - a reshuffled (randomized control) version of ConnectomeDB

show_homologene *Helper function to show available organisms via OmnipathR's homologene resource*

Description

Helper function to show available organisms via OmnipathR's homologene resource

Usage

```
show_homologene()
```

show_methods *Helper Function to return the methods in LIANA*

Description

Helper Function to return the methods in LIANA

Usage

```
show_methods()
```

Details

methods starting with 'call_*' were re-implemented in liana and albeit their original pipelines (and packages are still supported), we recommend using the liana re-implementations for efficiency

show_resources *Helper Function to return the Resources in LIANA*

Description

Helper Function to return the Resources in LIANA

Usage

```
show_resources()
```

Index