# Package: multinichenetr (via r-universe)

May 25, 2026

**Type** Package

**Title** MultiNicheNet: a flexible framework for differential cell-cell communication analysis from multi-sample multi-condition single-cell transcriptomics data

**Version** 2.1.0

**Description** This package allows you the investigate differences in intercellular communication between multiple conditions of interest. It is a flexible framework that enables multi-criteria prioritization of cell-cell communication patterns from scRNAseq datasets with complex experimental designs. These datasets can contain multiple samples (e.g. patients) over different groups of interest (e.g. disease subtypes). With MultiNicheNet, you can now better analyze the differences in cell-cell signaling between the different groups of interest.

**License** GPL-3 + file LICENSE

**Encoding** UTF-8

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

**BugReports** https://github.com/Zaoqu-Liu/multinichenetr/issues

**Depends** R (>= 3.5.0)

**LazyData** yes

**LazyDataCompression** bzip2

**Imports** circlize, patchwork, ggplot2, tibble, tidyr, purrr, ComplexHeatmap, stringr, generics, dplyr, grid, muscat, limma, SummarizedExperiment, S4Vectors, magrittr, scater, nichenetr, RColorBrewer, ggpubr, parallel, locfdr, edgeR, sva, SingleCellExperiment, ggbeeswarm, Hmisc, igraph, tidygraph, ggraph, scran, UpSetR, factoextra, viridis, foreach, doParallel

**Suggests** knitr, testthat, covr, tidyverse, BiocStyle, rmarkdown

**VignetteBuilder** knitr

**Remotes** github::Zaoqu-Liu/nichenetr

**Additional_repositories** https://zaoqu-liu.r-universe.dev

**RoxygenNote** 7.3.1

**Config/pak/sysreqs**
libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev libglu1-mesa-dev libgmp3-dev make libgsl0-dev libharfbuzz-dev jags libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libmpfr-dev libopenmpi-dev libssl-dev perl libproj-dev python3 libx11-dev zlib1g-dev

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

**Date/Publication** 2026-01-25 04:57:32 UTC

**RemoteUrl** https://github.com/Zaoqu-Liu/multinichenetr

**RemoteRef** main

**RemoteSha** 8f0a5748b8cd2eafbb1ce1dcbf988e4659847f1b

# Contents

---

`add_empirical_pval_fdr`

*Add empirical p-values and adjusted p-values to the DE output table.*

---

## Description

`add_empirical_pval_fdr` Add empirical p-values and adjusted p-values to the DE output of Muscat. Credits to Jeroen Gillis (cf satuRn package)

## Usage

```
add_empirical_pval_fdr(de_output_tidy, plot = FALSE)
```

## Arguments

| | |
|---|---|
| de_output_tidy | Data frame of DE results, containing at least the following columns: cluster_id, contrast, p_val, logFC. |
| plot | TRUE or FALSE (default): should we plot the z-score distribution? |

## Value

de_output_tidy dataframe with two columns added: p_emp and p_adj_emp.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)
de_output_tidy = muscat::resDS(celltype_de$sce, celltype_de$de_output, bind = "row", cpm = FALSE, frq = FALSE) %>%
de_output_tidy = add_empirical_pval_fdr(de_output_tidy)

## End(Not run)
```

---

# add_extra_criterion *add_extra_criterion*

---

## Description

add_extra_criterion Update the aggregated prioritization score based on one or more new prioritization criteria. Examples of useful criteria: proteomics data for scoring ligands/receptors for DE at protein level, spatial co-localization of cell types and/or ligand-receptor pairs.

## Usage

```
add_extra_criterion(prioritization_tables, new_criteria_tbl, regular_criteria_tbl, scenario = "regula
```

## Arguments

| | |
|---|---|
| prioritization_tables | output of 'generate_prioritization_tables' |
| new_criteria_tbl | tibble with 3 columns: criterion, weight, regularization_factor. See example code and vignette for usage. |
| regular_criteria_tbl | tibble with 3 columns: criterion, weight, regularization_factor. See example code and vignette for usage. |
| scenario | Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group. |

## Value

prioritization_tables with updated aggregated prioritization score based on the new criteria (same output as 'generate_prioritization_tables')

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")
```

```
celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

new_criteria_data = readRDS("data/additional_data_modality.rds")
prioritization_tables$group_prioritization_tbl = prioritization_tables$group_prioritization_tbl %>% inner_join(n
regular_criteria_tbl = tibble(criterion = c("scaled_lfc_ligand","scaled_p_val_ligand_adapted","scaled_lfc_recept
new_criteria_tbl = tibble(criterion = c("new_criterion1","new_criterion2"), weight = c(1,1), regularization_facto
prioritization_tables = add_extra_criterion(prioritization_tables, new_criteria_tbl, regular_criteria_tbl, scena

## End(Not run)
```

## alias_to_symbol_SCE *Convert aliases to official gene symbols in a SingleCellExperiment Object*

### Description

alias_to_symbol_SCEConvert aliases to official gene symbols in a SingleCellExperiment Object. Makes use of ‘nichenetr::convert_alias_to_symbols‘

### Usage

```
alias_to_symbol_SCE(sce, organism)
```

### Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| organism | Is sce data from "mouse" or "human" |

### Value

SingleCellExperiment Object

### Examples

```
## Not run:
sce = sce %>% alias_to_symbol_SCE("human")

## End(Not run)
```

## combine_sender_receiver_de *combine_sender_receiver_de*

### Description

combine_sender_receiver_de Combine Muscat differential expression output for senders and receivers by linkgin ligands to receptors based on the prior knowledge ligand-receptor network.

### Usage

```
combine_sender_receiver_de(sender_de, receiver_de, senders_oi, receivers_oi, lr_network)
```

## Arguments

| | |
|---|---|
| sender_de | Differential expression analysis output for the sender cell types. 'de_output_tidy' slot of the output of 'perform_muscat_de_analysis'. |
| receiver_de | Differential expression analysis output for the receiver cell types. 'de_output_tidy' slot of the output of 'perform_muscat_de_analysis'. |
| senders_oi | Default NULL: all celltypes will be considered as senders. If you want to select specific senders_oi: you can add this here as character vector. |
| receivers_oi | Default NULL: all celltypes will be considered as receivers. If you want to select specific receivers_oi: you can add this here as character vector. |
| lr_network | Prior knowledge Ligand-Receptor network (columns: ligand, receptor) |

## Value

Data frame combining Muscat DE output for sender and receiver linked to each other through joining by the ligand-receptor network.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)
sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de$de_output_tidy,
 receiver_de = celltype_de$de_output_tidy,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

## End(Not run)
```

combine_sender_receiver_info_ic
*combine_sender_receiver_info_ic*

## Description

combine_sender_receiver_info_ic Link the ligand-expression information of the Sender cell type to the receptor-expression information of the Receiver cell type. Linking via prior knowledge ligand-receptor network.

## Usage

```
combine_sender_receiver_info_ic(sender_info, receiver_info, senders_oi, receivers_oi, lr_network)
```

## Arguments

| | |
|---|---|
| sender_info | Output of ‘process_info_to_ic‘ with ‘ic_type = "sender"‘ |
| receiver_info | Output of ‘process_info_to_ic‘ with ‘ic_type = "receiver"‘ |
| senders_oi | Character vector indicating the names of the sender cell types of interest |
| receivers_oi | Character vector indicating the names of the receiver cell types of interest |
| lr_network | Prior knowledge Ligand-Receptor network (columns: ligand, receptor) |

## Value

List with data frames containing ligand-receptor sender-receiver combined expression information (see output of ‘get_avg_frac_exprs_abund‘ and ‘process_info_to_ic‘)

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g
receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf


## End(Not run)
```

compare_normal_emp_pvals *compare_normal_emp_pvals*

## Description

compare_normal_emp_pvals Compare nr and rank of DE genes between normal p-values and empirical p-values

## Usage

```
compare_normal_emp_pvals(DE_info, DE_info_emp, adj_pval = FALSE)
```

## Arguments

| | |
|---|---|
| DE_info | Output of 'get_DE_info' |
| DE_info_emp | Output of 'get_empirical_pvals' |
| adj_pval | Should the adjusted p-values be compared (TRUE) or the non-adjusted ones (FALSE)? Defautl: FALSE |

## Value

a list òf plots for each celltype-contrast pair: an upset plot and line plot are drawn.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
covariates = NA
contrasts_oi = c("'High-Low','Low-High'")
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
DE_info = get_DE_info(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   covariates = covariates,
   contrasts = contrasts_oi)
DE_info_emp = get_empirical_pvals(DE_info$celltype_de$de_output_tidy)
comparison_plots = compare_normal_emp_pvals(DE_info, DE_info_emp)

## End(Not run)
```

fix_frq_df *fix_frq_df*

## Description

fix_frq_df Fix muscat-feature/bug in fraction calculation: in case a there are no cells of a cell type in a sample, that expression fraction will be NA / NaN. Change these NA/NaN to 0.

## Usage

```
fix_frq_df(sce, frq_celltype_samples)
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| frq_celltype_samples | Sample-average data frame output of 'get_muscat_exprs_frac' |

## Value

Fixed data frame with fraction of cells expressing a gene.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
frq_df = get_muscat_exprs_frac(sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = group_id) %>% .$f
if(nrow(frq_df %>% dplyr::filter(is.na(fraction_sample))) > 0 | nrow(frq_df %>% dplyr::filter(is.nan(fraction_sam
  frq_df = fix_frq_df(sce, frq_df)
  }

## End(Not run)
```

geneinfo_alias_human *Gene annotation information: version 2 - january 2022 - suited for alias conversion*

## Description

A data.frame/tibble describing HGNC human gene symbols, their entrez ids and potential aliases.

### Usage

```
geneinfo_alias_human
```

### Format

A data frame/tibble

**symbol** human gene symbol

**entrez** human gene entrez

**alias** human gene alias

---

## geneinfo_alias_mouse *Gene annotation information: version 2 - january 2022 - suited for alias conversion*

---

### Description

A data.frame/tibble describing MGI mouse gene symbols, their entrez ids and potential aliases.

### Usage

```
geneinfo_alias_mouse
```

### Format

A data frame/tibble

**symbol** mouse gene symbol

**entrez** mouse gene entrez

**alias** mouse gene alias

---

## generate_prioritization_tables *generate_prioritization_tables*

---

### Description

`generate_prioritization_tables` Perform the MultiNicheNet prioritization of cell-cell interactions. Combine the following prioritization criteria in a single aggregated prioritization score: differential expression of ligand and receptor, cell-type-condition-specificity of expression of ligand and receptor, NicheNet ligand activity, fraction of samples in a group that express a senderLigand-receiverReceptor pair.

### Usage

```
generate_prioritization_tables(sender_receiver_info, sender_receiver_de, ligand_activities_targets_D
```

## Arguments

`sender_receiver_info`
: Output of 'combine_sender_receiver_info_ic'

`sender_receiver_de`
: Output of 'combine_sender_receiver_de'

`ligand_activities_targets_DEgenes`
: Output of 'get_ligand_activities_targets_DEgenes'

`contrast_tbl`
: Data frame providing names for each of the contrasts in contrasts_oi in the 'contrast' column, and the corresponding group of interest in the 'group' column. Entries in the 'group' column should thus be present in the group_id column in the metadata. Example for 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")': 'contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))'

`sender_receiver_tbl`
: Data frame with all sender-receiver cell type combinations (columns: sender and receiver)

`grouping_tbl`
: Data frame showing the groups of each sample (and batches per sample if applicable) (columns: sample and group; and if applicable all batches of interest)

`scenario`
: Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

`fraction_cutoff`
: Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample.

`abundance_data_receiver`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

`abundance_data_sender`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

`ligand_activity_down`
: For prioritization based on ligand activity: consider the max of up- and downregulation ('TRUE') or consider only upregulated activity ('FALSE', default from version 2 on).

## Value

List containing multiple data frames of prioritized senderLigand-receiverReceptor interactions (with sample- and group-based expression information), ligand activities and ligand-target links.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")

celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
```

```
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

## End(Not run)
```

---

`generate_prioritization_tables_condition_specific_celltypes_receiver` *generate_prioritization_tables_condition_specific_celltypes_receiver*

---

## Description

`generate_prioritization_tables_condition_specific_celltypes_receiver` Perform the MultiNicheNet prioritization of cell-cell interactions. Focus on including condition-specific cell types as receiver cells. This implies no DE information will be used for prioritization of receptors, nor ligand activities for ligands Combine the following prioritization criteria in a single aggregated prioritization score: differential expression of ligand and receptor, cell-type-condition-specificity of expression of ligand and receptor, NicheNet ligand activity, fraction of samples in a group that express a senderLigand-receiverReceptor pair.

## Usage

```
generate_prioritization_tables_condition_specific_celltypes_receiver(sender_receiver_info, sender_re
```

## Arguments

| | |
|---|---|
| `sender_receiver_info` | Output of ‘combine_sender_receiver_info_ic‘ |
| `sender_receiver_de` | Output of ‘combine_sender_receiver_de‘ |
| `ligand_activities_targets_DEgenes` | Output of ‘get_ligand_activities_targets_DEgenes‘ |
| `contrast_tbl` | Data frame providing names for each of the contrasts in contrasts_oi in the ’contrast’ column, and the corresponding group of interest in the ’group’ column. Entries in the ’group’ column should thus be present in the group_id column in the metadata. Example for ‘contrasts_oi = c("’A-(B+C+D)/3’, ’B-(A+C+D)/3’")‘: ‘contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))‘ |

`sender_receiver_tbl`
: Data frame with all sender-receiver cell type combinations (columns: sender and receiver)

`grouping_tbl`
: Data frame showing the groups of each sample (and batches per sample if applicable) (columns: sample and group; and if applicable all batches of interest)

`scenario`
: Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

`fraction_cutoff`
: Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample.

`abundance_data_receiver`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

`abundance_data_sender`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

`ligand_activity_down`
: For prioritization based on ligand activity: consider the max of up- and downregulation ('TRUE') or consider only upregulated activity ('FALSE', default from version 2 on).

## Value

List containing multiple data frames of prioritized senderLigand-receiverReceptor interactions (with sample- and group-based expression information), ligand activities and ligand-target links.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
```

```
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")

celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables_condition_specific_celltypes_receiver(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

## End(Not run)
```

generate_prioritization_tables_condition_specific_celltypes_sender
*generate_prioritization_tables_condition_specific_celltypes_sender*

## Description

generate_prioritization_tables_condition_specific_celltypes_sender Perform the MultiNicheNet prioritization of cell-cell interactions. Focus on including condition-specific cell types as sender cells. This implies no DE information will be used for prioritization of ligands. Combine the following prioritization criteria in a single aggregated prioritization score: differential expression of ligand and receptor, cell-type-condition-specificity of expression of ligand and receptor, NicheNet ligand activity, fraction of samples in a group that express a senderLigand-receiverReceptor pair.

## Usage

```
generate_prioritization_tables_condition_specific_celltypes_sender(sender_receiver_info, sender_rece
```

## Arguments

| | |
|---|---|
| sender_receiver_info | Output of 'combine_sender_receiver_info_ic' |
| sender_receiver_de | Output of 'combine_sender_receiver_de' |
| ligand_activities_targets_DEgenes | Output of 'get_ligand_activities_targets_DEgenes' |
| contrast_tbl | Data frame providing names for each of the contrasts in contrasts_oi in the 'contrast' column, and the corresponding group of interest in the 'group' column. Entries in the 'group' column should thus be present in the group_id column in the metadata. Example for 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")': 'contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))' |
| sender_receiver_tbl | Data frame with all sender-receiver cell type combinations (columns: sender and receiver) |
| grouping_tbl | Data frame showing the groups of each sample (and batches per sample if applicable) (columns: sample and group; and if applicable all batches of interest) |
| scenario | Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that |

will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

`fraction_cutoff`
: Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample.

`abundance_data_receiver`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

`abundance_data_sender`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

`ligand_activity_down`
: For prioritization based on ligand activity: consider the max of up- and downregulation ('TRUE') or consider only upregulated activity ('FALSE', default from version 2 on).

## Value

List containing multiple data frames of prioritized senderLigand-receiverReceptor interactions (with sample- and group-based expression information), ligand activities and ligand-target links.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")

celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf
```

```
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables_condition_specific_celltypes_sender(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

## End(Not run)
```

---

`generate_prioritization_tables_sampleAgnostic_multifactorial`
*generate_prioritization_tables_sampleAgnostic_multifactorial*

---

## Description

`generate_prioritization_tables_sampleAgnostic_multifactorial` Perform the MultiNicheNet prioritization of cell-cell interactions – only for analyses that are sample-agnostic/cell-level

and require multifactorial analyses. Combine the following prioritization criteria in a single aggregated prioritization score: differential expression of ligand and receptor, cell-type-condition-specificity of expression of ligand and receptor, NicheNet ligand activity, fraction of samples in a group that express a senderLigand-receiverReceptor pair.

## Usage

```
generate_prioritization_tables_sampleAgnostic_multifactorial(sender_receiver_info, sender_receiver_c
```

## Arguments

`sender_receiver_info`
: Output of 'combine_sender_receiver_info_ic'

`sender_receiver_de`
: Output of 'combine_sender_receiver_de'

`ligand_activities_targets_DEgenes`
: Output of 'get_ligand_activities_targets_DEgenes'

`contrast_tbl`
: Data frame providing names for each of the contrasts in contrasts_oi in the 'contrast' column, and the corresponding group of interest in the 'group' column. Entries in the 'group' column should thus be present in the group_id column in the metadata. Example for 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")': 'contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))'

`sender_receiver_tbl`
: Data frame with all sender-receiver cell type combinations (columns: sender and receiver)

`grouping_tbl`
: Data frame showing the groups of each sample (and batches per sample if applicable) (columns: sample and group; and if applicable all batches of interest)

`scenario`
: Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

`fraction_cutoff`
: Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample.

`abundance_data_receiver`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

`abundance_data_sender`
: Data frame with number of cells per cell type - sample combination; output of 'process_info_to_ic'

ligand_activity_down
: For prioritization based on ligand activity: consider the max of up- and downregulation ('TRUE') or consider only upregulated activity ('FALSE', default from version 2 on).

## Value

List containing multiple data frames of prioritized senderLigand-receiverReceptor interactions (with sample- and group-based expression information), ligand activities and ligand-target links.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")

celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)
```

```
ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables_sampleAgnostic_multifactorial(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

## End(Not run)
```

---

`get_abundance_info` *get_abundance_info*

---

## Description

`get_abundance_info` Visualize cell type abundances.

## Usage

```
get_abundance_info(sce, sample_id, group_id, celltype_id, min_cells = 10, senders_oi, receivers_oi, ba
```

## Arguments

| | |
|---|---|
| `sce` | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| `sample_id` | Name of the meta data column that indicates from which sample/patient a cell comes from |
| `group_id` | Name of the meta data column that indicates from which group/condition a cell comes from |
| `celltype_id` | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| `min_cells` | Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'. |

| | |
|---|---|
| senders_oi | Default NULL: all celltypes will be considered as senders. If you want to select specific senders_oi: you can add this here as character vector. |
| receivers_oi | Default NULL: all celltypes will be considered as receivers. If you want to select specific receivers_oi: you can add this here as character vector. |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |

## Value

List containing cell type abundance plots and abundance_data data frame.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
abundance_celltype_info = get_abundance_info(sce = sce, sample_id = sample_id, group_id = group_id, celltype_id =

## End(Not run)
```

---

# get_avg_pb_exprs *get_avg_pb_exprs*

---

## Description

get_avg_pb_exprs Calculate the average and normalized pseudobulk expression of each gene per sample and per group.

## Usage

```
get_avg_pb_exprs(sce, sample_id, celltype_id, group_id, batches = NA, min_cells = 10)
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |

| | |
|---|---|
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |
| min_cells | Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'. |

## Value

List containing data frames with average and normalized pseudobulk of expression per sample and per group.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
celltype_info = get_avg_pb_exprs(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = group_id)

## End(Not run)
```

---

get_DE_info *get_DE_info*

---

## Description

get_DE_info Perform differential expression analysis via Muscat - Pseudobulking approach. Also visualize the p-value distribution. Under the hood, the following function is used: 'perform_muscat_de_analysis'.

## Usage

```
get_DE_info(sce, sample_id, group_id, celltype_id, batches, covariates, contrasts_oi, expressed_df, mi
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |

`celltype_id` Name of the column in the meta data of sce that indicates the cell type of a cell.

`batches` NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector.

`covariates` NA if no covariates should be corrected for. If there should be corrected for covariates uring DE analysis, this argument should be the name(s) of the columns in the meta data that indicate the covariate(s). Can both be categorical and continuous. Pseudobulk expression values will not be corrected for the first element of this vector.

`contrasts_oi` String indicating the contrasts of interest (= which groups/conditions will be compared) for the differential expression and MultiNicheNet analysis. We will demonstrate here a few examples to indicate how to write this. Check the limma package manuals for more information about defining design matrices and contrasts for differential expression analysis.
If wanting to compare group A vs B: 'contrasts_oi = c("'A-B'")'
If wanting to compare group A vs B & B vs A: 'contrasts_oi = c("'A-B','B-A'")'
If wanting to compare group A vs B & A vs C & A vs D: 'contrasts_oi = c("'A-B','A-C', 'A-D'")'
If wanting to compare group A vs B and C: 'contrasts_oi = c("'A-(B+C)/2'")'
If wanting to compare group A vs B, C and D: 'contrasts_oi = c("'A-(B+C+D)/3'")'
If wanting to compare group A vs B, C and D & B vs A,C,D: 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")'
Note that the groups A, B, ... should be present in the meta data column 'group_id'.

`expressed_df` tibble with three columns: gene, celltype, expressed; this data frame indicates which genes can be considered as expressed in each cell type.

`min_cells` Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'.

`assay_oi_pb` Indicates which information of the assay of interest should be used (counts, scaled data,...). Default: "counts". See 'muscat::aggregateData'.

`fun_oi_pb` Indicates way of doing the pseudobulking. Default: "sum". See 'muscat::aggregateData'.

`de_method_oi` Indicates the DE method that will be used after pseudobulking. Default: "edgeR". See 'muscat::pbDS'.

`findMarkers` Indicate whether we should also calculate DE results with the classic scran::findMarkers approach. Default (recommended): FALSE. if TRUE: both pseudobulk-based and cell-level based DE results will be generated.

`contrast_tbl` see explanation in multi_nichenet_analysis function – here: only required to give as input if findMarkers = TRUE.

## Value

List with output of the differential expression analysis in 1) default format('muscat::pbDS()'), and 2) in a tidy table format ('muscat::resDS()') (both in the 'celltype_de' slot); Histogram plot of the p-values is also returned.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
covariates = NA
contrasts_oi = c("'High-Low','Low-High'")
frq_list = get_frac_exprs(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = group_id)
DE_info = get_DE_info(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   covariates = covariates,
   contrasts = contrasts_oi,
   expressed_df = frq_list$expressed_df)

## End(Not run)
```

---

`get_DE_info_sampleAgnostic`

*get_DE_info_sampleAgnostic*

---

## Description

`get_DE_info_sampleAgnostic` Perform differential expression analysis via scran::findMarkers approach. Also visualize the p-value distribution.

## Usage

```
get_DE_info_sampleAgnostic(sce, group_id, celltype_id, contrasts_oi, expressed_df, min_cells = 10, con
```

## Arguments

| | |
|---|---|
| `sce` | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| `group_id` | Name of the meta data column that indicates from which group/condition a cell comes from |
| `celltype_id` | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| `contrasts_oi` | String indicating the contrasts of interest (= which groups/conditions will be compared) for the differential expression and MultiNicheNet analysis. We will demonstrate here a few examples to indicate how to write this. Check the limma package manuals for more information about defining design matrices and contrasts for differential expression analysis. |

If wanting to compare group A vs B: 'contrasts_oi = c("'A-B'")'
If wanting to compare group A vs B & B vs A: 'contrasts_oi = c("'A-B','B-A'")'
If wanting to compare group A vs B & A vs C & A vs D: 'contrasts_oi = c("'A-B','A-C', 'A-D'")'
If wanting to compare group A vs B and C: 'contrasts_oi = c("'A-(B+C)/2'")'
If wanting to compare group A vs B, C and D: 'contrasts_oi = c("'A-(B+C+D)/3'")'
If wanting to compare group A vs B, C and D & B vs A,C,D: 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")'
Note that the groups A, B, ... should be present in the meta data column 'group_id'.

`expressed_df` tibble with three columns: gene, celltype, expressed; this data frame indicates which genes can be considered as expressed in each cell type.

`min_cells` Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'.

`contrast_tbl` see explanation in multi_nichenet_analysis function – here: only required to give as input if findMarkers = TRUE.

## Value

List with output of the differential expression analysis in 1) default format('muscat::pbDS()'), and 2) in a tidy table format ('muscat::resDS()') (both in the 'celltype_de' slot); Histogram plot of the p-values is also returned.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
covariates = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
frq_list = get_frac_exprs_sampleAgnostic(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g
DE_info = get_DE_info_sampleAgnostic(
   sce = sce,
   celltype_id = celltype_id,
   group_id = group_id,
   contrasts = contrasts_oi,
   expressed_df = frq_list$expressed_df,
   contrast_tbl = contrast_tbl)

## End(Not run)
```

get_empirical_pvals *get_empirical_pvals*

## Description

get_empirical_pvals Calculate empirical p-values based on a DE output. Show p-value distribution histograms. Under the hood, the following functions are used: ‘add_empirical_pval_fdr‘ and ‘get_FDR_empirical_plots_all‘

## Usage

```
get_empirical_pvals(de_output_tidy)
```

## Arguments

| | |
|---|---|
| de_output_tidy | Differential expression analysis output for the sender cell types. ‘de_output_tidy‘ slot of the output of ‘perform_muscat_de_analysis‘. |

## Value

‘de_output_tidy‘, but now 2 columns added with the empirical pvalues (normal and adjusted for multiple testing); Histogram plot of the empirical p-values is also returned.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
DE_info = get_DE_info(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)
DE_info_emp = get_empirical_pvals(DE_info$celltype_de$de_output_tidy)

## End(Not run)
```

`get_FDR_empirical` *Add empirical p-values and adjusted p-values to a subset of the DE output table.*

## Description

`get_FDR_empirical` Add empirical p-values and adjusted p-values to a subset of the DE output table. This is the function that works under the hood of 'add_empirical_pval_fdr'. Credits to Jeroen Gillis (cf satuRn package)

## Usage

```
get_FDR_empirical(de_output_tidy, cluster_id_oi, contrast_oi, plot = FALSE)
```

## Arguments

| | |
|---|---|
| `de_output_tidy` | Data frame of DE results, containing at least the following columns: cluster_id, contrast, p_val, logFC. |
| `cluster_id_oi` | Indicate which celltype DE results should be filtered for. |
| `contrast_oi` | Indicate which contrast DE results should be filtered for. |
| `plot` | TRUE or FALSE (default): should we plot the z-score distribution? |

## Value

de_output_tidy dataframe (for the celltype and contrast of interest) with two columns added: p_emp and p_adj_emp.

`get_FDR_empirical_plots` *Get diagnostic plots of the empirical null.*

## Description

`get_FDR_empirical_plots` Get diagnostic plots of the empirical null.. This is the function that works under the hood of 'get_FDR_empirical_plots_all'. Credits to Jeroen Gillis (cf satuRn package)

## Usage

```
get_FDR_empirical_plots(de_output_tidy, cluster_id_oi, contrast_oi)
```

### Arguments

| | |
|---|---|
| de_output_tidy | Data frame of DE results, containing at least the following columns: cluster_id, contrast, p_val, logFC. |
| cluster_id_oi | Indicate which celltype DE results should be filtered for. |
| contrast_oi | Indicate which contrast DE results should be filtered for. |

### Value

plot object

---

## get_FDR_empirical_plots_all

*Get diagnostic plots of the empirical null.*

---

### Description

get_FDR_empirical_plots_all Get diagnostic plots of the empirical null. Credits to Jeroen Gillis (cf satuRn package)

### Usage

```
get_FDR_empirical_plots_all(de_output_tidy)
```

### Arguments

| | |
|---|---|
| de_output_tidy | Data frame of DE results, containing at least the following columns: cluster_id, contrast, p_val, logFC. |

### Value

list of plots

---

## get_frac_exprs

*get_frac_exprs*

---

### Description

get_frac_exprs Calculate the average fraction of expression of each gene per sample and per group.

### Usage

```
get_frac_exprs(sce, sample_id, celltype_id, group_id, batches = NA, min_cells = 10, fraction_cutoff = 0
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |
| min_cells | Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'. |
| fraction_cutoff | Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample. |
| min_sample_prop | Parameter to define the minimal required nr of samples in which a gene should be expressed in more than 'fraction_cutoff' of cells in that sample (per cell type). This nr of samples is calculated as the 'min_sample_prop' fraction of the nr of samples of the smallest group (after considering samples with n_cells >= 'min_cells'. Default: 'min_sample_prop = 0.50'. Examples: if there are 8 samples in the smallest group, there should be min_sample_prop*8 (= 4 in this example) samples with sufficient fraction of expressing cells. |

## Value

List containing data frames with the fraction of expression per sample and per group.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
frac_info = get_frac_exprs(sce = sce, sample_id = sample_id, celltype_id = celltype_id, group_id = group_id)

## End(Not run)
```

get_frac_exprs_sampleAgnostic
*get_frac_exprs_sampleAgnostic*

## Description

get_frac_exprs_sampleAgnostic Calculate the average fraction of expression of each gene per group. All cells from all samples will be pooled per group/condition.

## Usage

```
get_frac_exprs_sampleAgnostic(sce, sample_id, celltype_id, group_id, batches = NA, min_cells = 10, frac
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |
| min_cells | Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'. |
| fraction_cutoff | Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample. |
| min_sample_prop | Default, and only recommended value = 1. Hereby, the gene should be expressed in at least one group/condition. |

## Value

List containing data frames with the fraction of expression per sample and per group.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
frac_info = get_frac_exprs_sampleAgnostic(sce = sce, sample_id = sample_id, celltype_id = celltype_id, group_id =

## End(Not run)
```

---

`get_ligand_activities_targets_DEgenes`
*get_ligand_activities_targets_DEgenes*

---

## Description

`get_ligand_activities_targets_DEgenes` Predict NicheNet ligand activities and ligand-target links for the receiver cell types. Uses 'nichenetr::predict_ligand_activities()' and 'nichenetr::get_weighted_ligand_target_link under the hood.

## Usage

```
get_ligand_activities_targets_DEgenes(receiver_de, receivers_oi, ligand_target_matrix, logFC_thresho
```

## Arguments

| | |
|---|---|
| `receiver_de` | Differential expression analysis output for the receiver cell types. 'de_output_tidy' slot of the output of 'perform_muscat_de_analysis'. |
| `receivers_oi` | Default NULL: all celltypes will be considered as receivers. If you want to select specific receivers_oi: you can add this here as character vector. |
| `ligand_target_matrix` | Prior knowledge model of ligand-target regulatory potential (matrix with ligands in columns and targets in rows). See https://github.com/saeyslab/nichenetr. |
| `logFC_threshold` | For defining the gene set of interest for NicheNet ligand activity: what is the minimum logFC a gene should have to belong to this gene set? Default: 0.25/ |
| `p_val_threshold` | For defining the gene set of interest for NicheNet ligand activity: what is the maximam p-value a gene should have to belong to this gene set? Default: 0.05. |
| `p_val_adj` | For defining the gene set of interest for NicheNet ligand activity: should we look at the p-value corrected for multiple testing? Default: FALSE. |
| `top_n_target` | For defining NicheNet ligand-target links: which top N predicted target genes. See 'nichenetr::get_weighted_ligand_target_links()'. |
| `verbose` | Indicate which different steps of the pipeline are running or not. Default: FALSE. |
| `n.cores` | The number of cores used for parallel computation of the ligand activities per receiver cell type. Default: 1 - no parallel computation. |

## Value

List with two data frames: one data frame containing the ligand activities and ligand-target links, one containing the DE gene information.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)
receiver_de = celltype_de$de_output_tidy
ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = receiver_de,
   receivers_oi = receivers_oi,
   ligand_target_matrix = ligand_target_matrix)

## End(Not run)
```

---

# get_ligand_activities_targets_DEgenes_beta

*get_ligand_activities_targets_DEgenes_beta*

---

## Description

`get_ligand_activities_targets_DEgenes_beta` BETA-version with new parallelization functionality – Predict NicheNet ligand activities and ligand-target links for the receiver cell types. Uses ‘nichenetr::predict_ligand_activities()‘ and ‘nichenetr::get_weighted_ligand_target_links()‘ under the hood.

## Usage

```
get_ligand_activities_targets_DEgenes_beta(receiver_de, receivers_oi, ligand_target_matrix, logFC_th
```

## Arguments

| | |
|---|---|
| receiver_de | Differential expression analysis output for the receiver cell types. 'de_output_tidy' slot of the output of 'perform_muscat_de_analysis'. |
| receivers_oi | Default NULL: all celltypes will be considered as receivers. If you want to select specific receivers_oi: you can add this here as character vector. |
| ligand_target_matrix | Prior knowledge model of ligand-target regulatory potential (matrix with ligands in columns and targets in rows). See https://github.com/saeyslab/nichenetr. |
| logFC_threshold | For defining the gene set of interest for NicheNet ligand activity: what is the minimum logFC a gene should have to belong to this gene set? Default: 0.25/ |
| p_val_threshold | For defining the gene set of interest for NicheNet ligand activity: what is the maximam p-value a gene should have to belong to this gene set? Default: 0.05. |
| p_val_adj | For defining the gene set of interest for NicheNet ligand activity: should we look at the p-value corrected for multiple testing? Default: FALSE. |
| top_n_target | For defining NicheNet ligand-target links: which top N predicted target genes. See 'nichenetr::get_weighted_ligand_target_links()'. |
| verbose | Indicate which different steps of the pipeline are running or not. Default: FALSE. |
| n.cores | The number of cores used for parallel computation of the ligand activities per receiver cell type. Default: 1 - no parallel computation. |

## Value

List with two data frames: one data frame containing the ligand activities and ligand-target links, one containing the DE gene information.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)
```

```
receiver_de = celltype_de$de_output_tidy
ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes_beta(
   receiver_de = receiver_de,
   receivers_oi = receivers_oi,
   ligand_target_matrix = ligand_target_matrix,
   n.cores = 2)

## End(Not run)
```

---

`get_muscat_exprs_avg` *get_muscat_exprs_avg*

---

## Description

`get_muscat_exprs_avg` Calculate sample- and group-average of gene expression per cell type.

## Usage

```
get_muscat_exprs_avg(sce, sample_id, celltype_id, group_id)
```

## Arguments

| | |
|---|---|
| `sce` | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| `sample_id` | Name of the meta data column that indicates from which sample/patient a cell comes from |
| `celltype_id` | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| `group_id` | Name of the meta data column that indicates from which group/condition a cell comes from |

## Value

Data frame with average gene expression per sample and per group.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
muscat_exprs_avg = get_muscat_exprs_avg(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = gr

## End(Not run)
```

get_muscat_exprs_frac *get_muscat_exprs_frac*

## Description

get_muscat_exprs_frac Calculate sample- and group-average of fraction of cells in a cell type expressing a gene.

## Usage

```
get_muscat_exprs_frac(sce, sample_id, celltype_id, group_id)
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |

## Value

List with two dataframes: one with fraction of cells in a cell type expressing a gene, averaged per sample; and one averaged per group.

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
muscat_exprs_frac = get_muscat_exprs_frac(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id =

## End(Not run)
```

get_pseudobulk_logCPM_exprs *get_pseudobulk_logCPM_exprs*

## Description

get_pseudobulk_logCPM_exprs Calculate the 'library-size' normalized pseudbulk counts per sample for each gene - returned values are similar to logCPM.

## Usage

```
get_pseudobulk_logCPM_exprs(sce, sample_id, celltype_id, group_id, batches = NA, assay_oi_pb = "counts
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |
| assay_oi_pb | Indicates which information of the assay of interest should be used (counts, scaled data,...). Default: "counts". See 'muscat::aggregateData'. |
| fun_oi_pb | Indicates way of doing the pseudobulking. Default: "sum". See 'muscat::aggregateData'. |

## Value

Data frame with logCPM-like values of the library-size corrected pseudobulked counts ('pb_sample') per gene per sample. pb_sample = log2( ((pb_raw/effective_library_size) * 1000000) + 1). effective_library_size = lib.size * norm.factors (through edgeR::calcNormFactors).

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
pseudobulk_logCPM_exprs = get_pseudobulk_logCPM_exprs(sce = sce, sample_id = sample_id, celltype_id =  celltype_id
```

```
## End(Not run)
```

---

`get_top_n_lr_pairs` *get_top_n_lr_pairs*

---

## Description

`get_top_n_lr_pairs` Get top n ligand-receptor pairs based on MultiNicheNet prioritization. Top pairs can be filtered by group, senders, receivers en based on top_n or cutoff based on the prioritization score.

## Usage

```
get_top_n_lr_pairs(prioritization_tables, top_n, groups_oi = NULL, senders_oi = NULL, receivers_oi = NU
```

## Arguments

| | |
|---|---|
| `prioritization_tables` | output of 'generate_prioritization_tables' |
| `top_n` | Indicates how many top ligand-receptor pairs need to be returned |
| `groups_oi` | character vector indicating the groups for which top pairs need to be returned. Default: NULL: all groups are considered. |
| `senders_oi` | character vector indicating the senders for which top pairs need to be returned. Default: NULL: all senders are considered. |
| `receivers_oi` | character vector indicating the receivers for which top pairs need to be returned. Default: NULL: all receivers are considered. |
| `rank_per_group` | Should top_n be given per group (TRUE, default) or over all groups (FALSE) |

## Value

Tibble that shows the top-ranked ligand-receptor pairs for the groups and cell types of interest

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
```

```
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")

celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

top50_tbl = get_top_n_lr_pairs(prioritization_tables, 50)
```

```
## End(Not run)
```

---

`infer_intercellular_regulatory_network` *infer_intercellular_regulatory_network*

---

## Description

`infer_intercellular_regulatory_network` Infer a network showing the gene regulatory links between ligands from sender cell types to their induced ligands/receptors in receiver cell types. Links are only drawn if the ligand/receptor in the receiver is a potential downstream target of the ligand (based on prior knowledge, and optionally with sufficient correlation in expression across the different samples).

## Usage

```
infer_intercellular_regulatory_network(lr_target_df, prioritized_tbl_oi)
```

## Arguments

| | |
|---|---|
| `lr_target_df` | tibble with columns: group, sender, receiver, ligand, receptor, id, target, direction_regulation |
| `prioritized_tbl_oi` | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |

## Value

list containing 3 elements: links, nodes, prioritized_lr_interactions. Links is a tibble that can be used to create a network with igraph, together with the node tibble. prioritized_lr_interactions is the subset of the input prioritized_tbl_oi, focusing on interaction elements present in this network, and hereby further prioritizing.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
```

```
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
lr_target_prior_cor_filtered = output$lr_target_prior_cor %>% filter(scaled_prior_score > 0.50 & (pearson > 0.66 |
 prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% distinct(id, ligand, receptor, se
 prioritized_tbl_oi = prioritized_tbl_oi %>% filter(id %in% lr_target_prior_cor_filtered$id)
 prioritized_tbl_oi = prioritized_tbl_oi %>% group_by(ligand, sender, group) %>% top_n(2, prioritization_score)
 lr_target_df = lr_target_prior_cor_filtered  %>% distinct(group, sender, receiver, ligand, receptor, id, target, c
 network = infer_intercellular_regulatory_network(lr_target_df, prioritized_tbl_oi)

## End(Not run)
```

---

`ligand_target_matrix_test` *Ligand-Target Matrix: subset NicheNet 2.0.*

---

## Description

Matrix with ligand-target regulatory potential scores

## Usage

```
ligand_target_matrix_test
```

## Format

A matrix

lr_target_prior_cor_inference *lr_target_prior_cor_inference*

## Description

lr_target_prior_cor_inference Calculate the pearson and spearman expression correlation between ligand-receptor pair pseudobulk expression products and DE gene expression product. Add the NicheNet ligand-target regulatory potential score as well as prior knowledge support for the LR–>Target link.

## Usage

```
lr_target_prior_cor_inference(receivers_oi, abundance_expression_info, celltype_de, grouping_tbl, pri
```

## Arguments

| | |
|---|---|
| receivers_oi | Character vector with the names of the receiver cell types of interest |
| abundance_expression_info | Output of 'get_abundance_expression_info_separate' or 'get_abundance_expression_info' |
| celltype_de | Output of 'perform_muscat_de_analysis' |
| grouping_tbl | Data frame showing the groups of each sample (and batches per sample if applicable) (columns: sample and group; and if applicable all batches of interest) |
| prioritization_tables | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |
| ligand_target_matrix | Prior knowledge model of ligand-target regulatory potential (matrix with ligands in columns and targets in rows). See https://github.com/saeyslab/nichenetr. |
| logFC_threshold | For defining the gene set of interest for NicheNet ligand activity: what is the minimum logFC a gene should have to belong to this gene set? Default: 0.25/ |
| p_val_threshold | For defining the gene set of interest for NicheNet ligand activity: what is the maximam p-value a gene should have to belong to this gene set? Default: 0.05. |
| p_val_adj | For defining the gene set of interest for NicheNet ligand activity: should we look at the p-value corrected for multiple testing? Default: FALSE. |
| top_n_LR | top nr of LR pairs for which correlation with target genes will be calculated. Is 2500 by default. If you want to calculate correlation for all LR pairs, set this argument to NA. |

## Value

Tibble with expression correlation and prior knowledge support measures for ligand-receptor to target gene links

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")

celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

abundance_expression_info = list(abund_plot_sample = abund_plot, abund_plot_group = abund_plot_boxplot, abundance

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)
```

```
sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

receivers_oi = prioritization_tables$group_prioritization_tbl$receiver %>% unique()
lr_target_prior_cor = lr_target_prior_cor_inference(receivers_oi, abundance_expression_info, celltype_de, groupi


## End(Not run)
```

---

`make_circos_group_comparison` *make_circos_group_comparison*

---

## Description

`make_circos_group_comparison` Make a circos plot with top prioritized ligand-receptor interactions for each group of interest. In each circos, all the possible LR pairs will be shown, but arrows will only be drawn between the ones belonging to the group of interest.

## Usage

```
make_circos_group_comparison(prioritized_tbl_oi, colors_sender, colors_receiver)
```

## Arguments

| | |
|---|---|
| `prioritized_tbl_oi` | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |
| `colors_sender` | Named vector of colors associated to each sender cell type. Vector = color, names = sender names. If sender and receiver cell types are the same, recommended that this vector is the same as 'colors_receiver'. |

colors_receiver
: Named vector of colors associated to each receiver cell type. Vector = color, names = sender names. Vector = color, names = sender names. If sender and receiver cell types are the same, recommended that this vector is the same as 'colors_receiver'.

## Value

a list with a circos plot for each group of interest, and a legend showing the color corresponding to each sender/receiver cell type.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )

## End(Not run)
```

---

# make_circos_one_group *make_circos_one_group*

---

## Description

make_circos_one_group Make a circos plot with top prioritized ligand-receptor interactions for one group of interest.

## Usage

```
make_circos_one_group(prioritized_tbl_oi, colors_sender, colors_receiver)
```

## Arguments

| | |
|---|---|
| prioritized_tbl_oi | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |
| colors_sender | Named vector of colors associated to each sender cell type. Vector = color, names = sender names. If sender and receiver cell types are the same, recommended that this vector is the same as 'colors_receiver'. |
| colors_receiver | Named vector of colors associated to each receiver cell type. Vector = color, names = sender names. Vector = color, names = sender names. If sender and receiver cell types are the same, recommended that this vector is the same as 'colors_receiver'. |

## Value

a list with a circos plot for one group of interest, and a legend showing the color corresponding to each sender/receiver cell type.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )


## End(Not run)
```

make_DEgene_dotplot_pseudobulk
*make_DEgene_dotplot_pseudobulk*

## Description

make_DEgene_dotplot_pseudobulk Visualize the scaled pseudobulk expression of DE genes per sample, and compare the different groups. Genes in rows, samples in columns

## Usage

```
make_DEgene_dotplot_pseudobulk(genes_oi, celltype_info, prioritization_tables, celltype_oi, grouping_
```

## Arguments

| | |
|---|---|
| genes_oi | Character vector with names of genes to visualize |
| celltype_info | 'celltype_info' or 'receiver_info' slot of the output of the 'multi_nichenet_analysis' function |
| prioritization_tables | 'prioritization_tables' slot of the output of the 'generate_prioritization_tables' or 'multi_nichenet_analysis' function |
| celltype_oi | Character vector with names of celltype of interest |
| grouping_tbl | 'grouping_tbl' slot of the output of the 'multi_nichenet_analysis' function |
| groups_oi | Which groups to show? Default: NULL – will show all groups. |

## Value

Gene expression dotplot list: pseudobulk version and single-cell version

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
```

```
    batches = batches,
    lr_network = lr_network,
    ligand_target_matrix = ligand_target_matrix,
    contrasts_oi = contrasts_oi,
    contrast_tbl = contrast_tbl

    )
group_oi = "High"
receiver_oi = "Malignant"
targets_oi = output$ligand_activities_targets_DEgenes$de_genes_df %>% inner_join(contrast_tbl) %>% filter(group =
p_target = make_DEgene_dotplot_pseudobulk(genes_oi = targets_oi, celltype_info = output$celltype_info, prioritiza

## End(Not run)
```

---

`make_DEgene_dotplot_pseudobulk_batch`
*make_DEgene_dotplot_pseudobulk_batch*

---

## Description

`make_DEgene_dotplot_pseudobulk_batch` Visualize the scaled pseudobulk expression of DE genes per sample, and compare the different groups. Genes in rows, samples in columns

## Usage

```
make_DEgene_dotplot_pseudobulk_batch(genes_oi, celltype_info, prioritization_tables, celltype_oi, bat
```

## Arguments

| | |
|---|---|
| `genes_oi` | Character vector with names of genes to visualize |
| `celltype_info` | 'celltype_info' or 'receiver_info' slot of the output of the 'multi_nichenet_analysis' function |
| `prioritization_tables` | 'prioritization_tables' slot of the output of the 'generate_prioritization_tables' or 'multi_nichenet_analysis' function |
| `celltype_oi` | Character vector with names of celltype of interest |
| `batch_oi` | Name of the batch that needs to be visualized for each sample |
| `grouping_tbl` | 'grouping_tbl' slot of the output of the 'multi_nichenet_analysis' function |
| `groups_oi` | Which groups to show? Default: NULL – will show all groups. |

## Value

Gene expression dotplot list: pseudobulk version and single-cell version

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = "batch"
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
group_oi = "High"
receiver_oi = "Malignant"
targets_oi = output$ligand_activities_targets_DEgenes$de_genes_df %>% inner_join(contrast_tbl) %>% filter(group =
p_target = make_DEgene_dotplot_pseudobulk_batch(genes_oi = targets_oi, celltype_info = output$celltype_info, prio

## End(Not run)
```

---

`make_DEgene_dotplot_pseudobulk_reversed`
*make_DEgene_dotplot_pseudobulk_reversed*

---

## Description

`make_DEgene_dotplot_pseudobulk_reversed` Visualize the scaled pseudobulk expression of DE genes per sample, and compare the different groups. Genes and sample positions are reversed compared to 'make_DEgene_dotplot_pseudobulk': genes in columns, samples in rows.

## Usage

```
make_DEgene_dotplot_pseudobulk_reversed(genes_oi, celltype_info, prioritization_tables, celltype_oi,
```

## Arguments

| | |
|---|---|
| genes_oi | Character vector with names of genes to visualize |
| celltype_info | 'celltype_info' or 'receiver_info' slot of the output of the 'multi_nichenet_analysis' function |
| prioritization_tables | 'prioritization_tables' slot of the output of the 'generate_prioritization_tables' or 'multi_nichenet_analysis' function |
| celltype_oi | Character vector with names of celltype of interest |
| grouping_tbl | 'grouping_tbl' slot of the output of the 'multi_nichenet_analysis' function |
| groups_oi | Which groups to show? Default: NULL – will show all groups. |
| target_regulation_df | NULL or a data frame with the columns gene and direction_regulation: indicates whether genes should be divided in up- and downregulated. Default: NULL – no division. |

## Value

Gene expression dotplot list: pseudobulk version and single-cell version

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
group_oi = "High"
receiver_oi = "Malignant"
targets_oi = output$ligand_activities_targets_DEgenes$de_genes_df %>% inner_join(contrast_tbl) %>% filter(group =
p_target = make_DEgene_dotplot_pseudobulk_reversed(genes_oi = targets_oi, celltype_info = output$celltype_info, p
```

```
## End(Not run)
```

---

`make_ggraph_ligand_target_links` *make_ggraph_ligand_target_links*

---

## Description

`make_ggraph_ligand_target_links` Make a network showing the gene regulatory links between ligands from sender cell types to their induced ligands/receptors in receiver cell types. Lins are only drawn if the ligand/receptor in the receiver is a potential downstream target of the ligand based on prior knowledge and sufficient correlation in expression across the different samples.

## Usage

```
make_ggraph_ligand_target_links(lr_target_prior_cor_filtered, prioritized_tbl_oi, colors)
```

## Arguments

| | |
|---|---|
| `lr_target_prior_cor_filtered` | Data frame filtered from 'lr_target_prior_cor' (= output of 'multi_nichenet_analysis' or 'lr_target_prior_cor_inference'). Filter should be done to keep onl LR–>Target links that are both supported by prior knowledge and correlation in terms of expression. |
| `prioritized_tbl_oi` | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |
| `colors` | Named vector of colors associated to each sender cell type. Vector = color, names = sender names. |

## Value

ggplot object with plot of LR–>Target links, the graph object itself, and the prioritized LR links

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
```

```
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
lr_target_prior_cor_filtered = output$lr_target_prior_cor %>% filter(scaled_prior_score > 0.50 & (pearson > 0.66 |
 prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% distinct(id, ligand, receptor, se
 prioritized_tbl_oi = prioritized_tbl_oi %>% filter(id %in% lr_target_prior_cor_filtered$id)
 prioritized_tbl_oi = prioritized_tbl_oi %>% group_by(ligand, sender, group) %>% top_n(2, prioritization_score)
graph_plot = make_ggraph_ligand_target_links(lr_target_prior_cor_filtered = lr_target_prior_cor_filtered, priori
graph_plot$plot

## End(Not run)
```

---

make_ggraph_signaling_path *make_ggraph_signaling_path*

---

## Description

make_ggraph_signaling_path Visualize the Ligand-Receptor to target signaling paths

## Usage

```
make_ggraph_signaling_path(signaling_graph_list, colors, ligands_all, receptors_all, targets_all)
```

## Arguments

| | |
|---|---|
| signaling_graph_list | Output of 'nichenetr::get_ligand_signaling_path_with_receptor' |
| colors | Named vector of colors associated to each node type: Example: colors <- c("ligand" = "indianred2", "receptor" = "orange", "target" = "steelblue2", "mediator" = "grey25"). |
| ligands_all | Name of the ligand(s) |
| receptors_all | Name of the receptor(s) |
| targets_all | Name of the target(s) |

## Value

ggraph and tidygraph objec of signaling paths between predefined LR–>Target links

## Examples

```
## Not run:
library(dplyr)
weighted_networks = readRDS(url("https://zenodo.org/record/3260758/files/weighted_networks.rds"))
ligand_tf_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_tf_matrix.rds"))
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
sig_network = readRDS(url("https://zenodo.org/record/3260758/files/signaling_network.rds"))
gr_network = readRDS(url("https://zenodo.org/record/3260758/files/gr_network.rds"))
ligands_all = "COL1A1" # this can be a list of multiple ligands if required
receptors_all = "ITGB1"
targets_all = c("S100A1","SERPINE1")
active_signaling_network = nichenetr::get_ligand_signaling_path_with_receptor(ligand_tf_matrix = ligand_tf_matri
data_source_network = nichenetr::infer_supporting_datasources(signaling_graph_list = active_signaling_network,lr
active_signaling_network_min_max = active_signaling_network
active_signaling_network_min_max$sig = active_signaling_network_min_max$sig %>% mutate(weight = ((weight-min(wei
active_signaling_network_min_max$gr = active_signaling_network_min_max$gr %>% mutate(weight = ((weight-min(weigh
colors = c("ligand" = "indianred2", "receptor" = "orange", "target" = "steelblue2", "mediator" = "grey75")
ggraph_signaling_path = make_ggraph_signaling_path(active_signaling_network_min_max, colors)#' colors = c("ligan


## End(Not run)
```

---

`make_ligand_activity_plots` *make_ligand_activity_plots*

---

## Description

`make_ligand_activity_plots` Visualize the ligand activities (normal and scaled) of each group-receiver combination

## Usage

```
make_ligand_activity_plots(prioritization_tables, ligands_oi, contrast_tbl, widths = NULL)
```

## Arguments

| | |
|---|---|
| `prioritization_tables` | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |
| `ligands_oi` | Character vector of ligands for which the activities should be visualized |
| `contrast_tbl` | Table to link the contrast definitions to the group ids. |
| `widths` | Vector of 2 elements: Width of the scaled ligand activity panel, width of the ligand activity panel. Default NULL: automatically defined based number of group-receiver combinations. If manual change: example format: c(3,2) |

## Value

Heatmap of ligand activities (normal and scaled) of each group-receiver combination

## Examples

```
## Not run:

library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
ligands_oi = output$prioritization_tables$ligand_activities_target_de_tbl %>% inner_join(contrast_tbl) %>% group
plot_oi = make_ligand_activity_plots(output$prioritization_tables, ligands_oi, contrast_tbl)
plot_oi

## End(Not run)
```

---

# make_ligand_activity_target_plot

*make_ligand_activity_target_plot*

---

## Description

make_ligand_activity_target_plot Summary plot showing the activity of prioritized ligands acting on a receiver cell type of interest, together with the predicted target genes and their sample-by-sample expression

## Usage

```
make_ligand_activity_target_plot(group_oi, receiver_oi, prioritized_tbl_oi, prioritization_tables, li
```

## Arguments

| | |
|---|---|
| group_oi | Character vector: name of the group of interest |
| receiver_oi | Character vector of receiver cell type of interest |
| prioritized_tbl_oi | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |
| prioritization_tables | 'prioritization_tables' slot of the output of the 'generate_prioritization_tables' or 'multi_nichenet_analysis' function |
| ligand_activities_targets_DEgenes | Sublist in the output of 'multi_nichenet_analysis' |
| contrast_tbl | Table to link the contrast definitions to the group ids. |
| grouping_tbl | 'grouping_tbl' slot of the output of the 'multi_nichenet_analysis' function |
| receiver_info | 'celltype_info' or 'receiver_info' slot of the output of the 'multi_nichenet_analysis' function |
| ligand_target_matrix | Prior knowledge model of ligand-target regulatory potential (matrix with ligands in columns and targets in rows). See https://github.com/saeyslab/nichenetr. |
| groups_oi | Which groups to show? Default: NULL – will show all groups. |
| plot_legend | if TRUE (default): show legend on the same figure, if FALSE (recommended): show legend in separate figure |
| heights | Vector of 2 elements: height of the ligand-activity-target panel, height of the target expression panel. Default NULL: automatically defined based on nr of ligands and samples. If manual change: example format: c(1,1) |
| widths | Vector of 3 elements: Width of the scaled ligand activity panel, width of the ligand activity panel, width of the ligand-target heatmap panel. Default NULL: automatically defined based on nr of target genes and group-receiver combinations. If manual change: example format: c(1,1,10) |

## Value

Summary plot showing the activity of prioritized ligands acting on a receiver cell type of interest, together with the predicted target genes and their sample-by-sample expression

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
```

```
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
group_oi = "High"
receiver_oi = "Malignant"
prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% filter(fraction_expressing_ligand
combined_plot = make_ligand_activity_target_plot(group_oi, receiver_oi, prioritized_tbl_oi, output$prioritizatio


## End(Not run)
```

---

`make_ligand_receptor_violin_plot` *make_ligand_receptor_violin_plot*

---

## Description

`make_ligand_receptor_violin_plot` Plot combining a violin plot of of the ligand of interest in the sender cell type of interest, and a violin plot of the receptor of interest in the receiver cell type of interest.

## Usage

```
make_ligand_receptor_violin_plot(sce, ligand_oi, receptor_oi, sender_oi, receiver_oi, group_oi, group
```

## Arguments

| | |
|---|---|
| `sce` | SingleCellExperiment object |
| `ligand_oi` | Character vector of name of the ligand of interest |
| `receptor_oi` | Character vector of name of the receptor of interest |
| `sender_oi` | Character vector with the names of the sender cell type of interest |
| `receiver_oi` | Character vector with the names of the receiver cell type of interest |
| `group_oi` | Character vector of name of the group of interest |

group_id Name of the meta data column that indicates from which group/condition a cell comes from

sample_id Name of the colData(sce) column in which the id of the sample is defined

celltype_id Name of the meta data column that indicates the cell type of a cell

batch_oi Name of a batch of interest based on which the visualization will be split. Default: NA: no batch.

background_groups
Default NULL: all groups in the group_id metadata column will be chosen. But user can fill in a character vector with the names of all gruops of interest.

## Value

Plot combining a violin plot of of the ligand of interest in the sender cell type of interest, and a violin plot of the receptor of interest in the receiver cell type of interest.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl
     )
ligand_oi = "DLL1"
receptor_oi = "NOTCH3"
group_oi = "High"
sender_oi = "Malignant"
receiver_oi = "myofibroblast"
p_violin = make_ligand_receptor_violin_plot(sce = sce, ligand_oi = ligand_oi, receptor_oi = receptor_oi, group_oi


## End(Not run)
```

make_lite_output *make_lite_output*

## Description

make_lite_output Reduce the size of the MultiNicheNet output object (for memory efficiency), by only keeping expression information for present ligands, receptors, and genes DE in at least one probed condition.

## Usage

```
make_lite_output(multinichenet_output, top_n_LR = 2500)
```

## Arguments

| | |
|---|---|
| multinichenet_output | Output of a MultiNicheNet analysis (result of 'multi_nichenet_analysis()'). |
| top_n_LR | top nr of LR pairs for which correlation with target genes will be calculated. Is 2500 by default. If you want to calculate correlation for all LR pairs, set this argument to NA. |

## Value

multinichenet output list (= result of 'multi_nichenet_analysis()'), but now filtered such that expression information is only returned for present ligands, receptors, and genes DE in at least one probed condition.

## Examples

```
## Not run:
library(dplyr)
multinichenet_output = multinichenet_output %>% make_lite_output()


## End(Not run)
```

---

make_lite_output_condition_specific
*make_lite_output_condition_specific*

## Description

make_lite_output_condition_specific Reduce the size of the MultiNicheNet output object (for memory efficiency), by only keeping expression information for present ligands, receptors, and genes DE in at least one probed condition. This function is specific for an object that contains the prioritization tables of the condition-specific cell type workflow as well.

### Usage

```
make_lite_output_condition_specific(multinichenet_output, top_n_LR = 2500)
```

### Arguments

| | |
|---|---|
| multinichenet_output | Output of a MultiNicheNet analysis (result of 'multi_nichenet_analysis()'). |
| top_n_LR | top nr of LR pairs for which correlation with target genes will be calculated. Is 2500 by default. If you want to calculate correlation for all LR pairs, set this argument to NA. |

### Value

multinichenet output list (= result of 'multi_nichenet_analysis()'), but now filtered such that expression information is only returned for present ligands, receptors, and genes DE in at least one probed condition.

### Examples

```
## Not run:
library(dplyr)
multinichenet_output = multinichenet_output %>% make_lite_output_condition_specific()


## End(Not run)
```

---

## make_lr_target_correlation_plot
*make_lr_target_correlation_plot*

---

### Description

make_lr_target_correlation_plot Plot Ligand-Receptor expression, Ligand-Receptor–>Target gene links that are both supported by prior knowledge and have correlation in expression, and Target expression

### Usage

```
make_lr_target_correlation_plot(prioritization_tables, prioritized_tbl_oi, lr_target_prior_cor_filte
```

### Arguments

| | |
|---|---|
| prioritization_tables | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |

prioritized_tbl_oi
: Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples).

lr_target_prior_cor_filtered
: Data frame filtered from 'lr_target_prior_cor' (= output of 'multi_nichenet_analysis' or 'lr_target_prior_cor_inference'). Filter should be done to keep onl LR–>Target links that are both supported by prior knowledge and correlation in terms of expression.

grouping_tbl
: 'grouping_tbl' slot of the output of the 'multi_nichenet_analysis' function

receiver_info
: 'celltype_info' or 'receiver_info' slot of the output of the 'multi_nichenet_analysis' function

receiver_oi
: Character vector with the names of the receiver cell type of interest

plot_legend
: if TRUE (default): show legend on the same figure, if FALSE (recommended): show legend in separate figure

heights
: Vector of 2 elements: height of the ligand-activity-target panel, height of the target expression panel. Default NULL: automatically defined based on nr of ligands and samples. If manual change: example format: c(1,1)

widths
: Vector of 3 elements: Width of the scaled ligand activity panel, width of the ligand activity panel, width of the ligand-target heatmap panel. Default NULL: automatically defined based on nr of target genes and group-receiver combinations. If manual change: example format: c(1,1,10)

## Value

ggplot object with a combined plot of LR expression vs target expression

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
```

```
    contrasts_oi = contrasts_oi,
    contrast_tbl = contrast_tbl
    )
group_oi = "High"
receiver_oi = "Malignant"
prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% distinct(id, ligand, receptor, ser
lr_target_prior_cor_filtered = output$lr_target_prior_cor %>% filter(scaled_prior_score > 0.50 & (pearson > 0.66 |
prioritized_tbl_oi = prioritized_tbl_oi %>% filter(id %in% lr_target_prior_cor_filtered$id)
prioritized_tbl_oi = prioritized_tbl_oi %>% group_by(ligand, sender, group) %>% top_n(2, prioritization_score)
lr_target_correlation_plot = make_lr_target_correlation_plot(output$prioritization_tables, prioritized_tbl_oi, l

## End(Not run)
```

---

`make_lr_target_prior_cor_heatmap`
*make_lr_target_prior_cor_heatmap*

---

## Description

`make_lr_target_prior_cor_heatmap` Plot Ligand-Receptor–>Target gene links that are both supported by prior knowledge and have correlation in expression

## Usage

```
make_lr_target_prior_cor_heatmap(lr_target_prior_cor_filtered, add_grid = TRUE)
```

## Arguments

| | |
|---|---|
| `lr_target_prior_cor_filtered` | Data frame filtered from 'lr_target_prior_cor' (= output of 'multi_nichenet_analysis' or 'lr_target_prior_cor_inference'). Filter should be done to keep onl LR–>Target links that are both supported by prior knowledge and correlation in terms of expression. |
| `add_grid` | add a ggplot-facet grid to easier link LR pairs to target genes. Default: TRUE. |

## Value

ggplot object with plot of LR–>Target links

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
```

```
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl
     )
lr_target_prior_cor_filtered = output$lr_target_prior_cor %>% filter(scaled_prior_score > 0.50 & (pearson > 0.66 |
lr_target_prior_cor_heatmap = make_lr_target_prior_cor_heatmap(lr_target_prior_cor_filtered)

## End(Not run)
```

---

`make_lr_target_scatter_plot` *make_lr_target_scatter_plot*

---

## Description

`make_lr_target_scatter_plot` Plot Ligand-Receptor pseudobulk expression product values vs pseudobulk expression of correlated target genes supported by prior information.

## Usage

```
make_lr_target_scatter_plot(prioritization_tables, ligand_oi, receptor_oi, sender_oi, receiver_oi, re
```

## Arguments

| | |
|---|---|
| `prioritization_tables` | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |
| `ligand_oi` | Character vector of name of the ligand of interest |
| `receptor_oi` | Character vector of name of the receptor of interest |
| `sender_oi` | Character vector with the names of the sender cell type of interest |
| `receiver_oi` | Character vector with the names of the receiver cell type of interest |
| `receiver_info` | 'celltype_info' or 'receiver_info' slot of the output of the 'multi_nichenet_analysis' function |
| `grouping_tbl` | 'grouping_tbl' slot of the output of the 'multi_nichenet_analysis' function |

lr_target_prior_cor_filtered
: Data frame filtered from 'lr_target_prior_cor' (= output of 'multi_nichenet_analysis' or 'lr_target_prior_cor_inference'). Filter should be done to keep onl LR–>Target links that are both supported by prior knowledge and correlation in terms of expression.

## Value

ggplot object with plot of LR expression vs target expression

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl
     )
ligand_oi ="IL24"
receptor_oi = "IL22RA1"
sender_oi = "CAF"
receiver_oi ="Malignant"
lr_target_prior_cor_filtered = output$lr_target_prior_cor %>% filter(scaled_prior_score > 0.50 & (pearson > 0.66 |
lr_target_scatter_plot = make_lr_target_scatter_plot(output$prioritization_tables, ligand_oi, receptor_oi, sende

## End(Not run)
```

---

make_sample_lr_prod_activity_batch_plots
: *make_sample_lr_prod_activity_batch_plots*

---

## Description

make_sample_lr_prod_activity_batch_plots Visualize the scaled product of Ligand-Receptor (pseudobulk) expression per sample, and compare the different groups. In addition, show the NicheNet ligand activities in each receiver-celltype combination. On top of this summary plot, a heatmap indicates the batch value for each displayed sample.

## Usage

```
make_sample_lr_prod_activity_batch_plots(prioritization_tables, prioritized_tbl_oi, grouping_tbl, ba
```

## Arguments

| | |
|---|---|
| prioritization_tables | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |
| prioritized_tbl_oi | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |
| grouping_tbl | Data frame linking the sample_id, group_id and batch_oi |
| batch_oi | Name of the batch that needs to be visualized for each sample |
| widths | Vector of 4 elements: Width of the LR exprs product panel, width of the scaled ligand activity panel, width of the ligand activity panel & width of the cell-type specificity panel. Default NULL: automatically defined based on nr of samples vs nr of group-receiver combinations. If manual change: example format: c(6,2,2,1) |
| heights | Vector of 2 elements: Height of the batch panel and height of the ligand-receptor prod+activity panel. Default NULL: automatically defined based on the nr of Ligand-Receptor pairs. If manual change: example format: c(1,5) |

## Value

Ligand-Receptor Expression Product & Ligand Activities Dotplot/Heatmap, complemented with a heatmap indicating the batch of interest

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = "batch"
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
```

```
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
group_oi = "High"
batch_oi = "batch"
prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% filter(fraction_expressing_ligand
plot_oi = make_sample_lr_prod_activity_batch_plots(output$prioritization_tables, prioritized_tbl_oi, output$grou
plot_oi

## End(Not run)
```

---

`make_sample_lr_prod_activity_plots` *make_sample_lr_prod_activity_plots*

---

## Description

`make_sample_lr_prod_activity_plots` Visualize the scaled product of Ligand-Receptor (pseudobulk) expression per sample, and compare the different groups. In addition, show the NicheNet ligand activities in each receiver-celltype combination.

## Usage

```
make_sample_lr_prod_activity_plots(prioritization_tables, prioritized_tbl_oi, widths = NULL)
```

## Arguments

| | |
|---|---|
| `prioritization_tables` | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |
| `prioritized_tbl_oi` | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |
| `widths` | Vector of 4 elements: Width of the LR exprs product panel, width of the scaled ligand activity panel, width of the ligand activity panel & width of the cell-type specificity panel. Default NULL: automatically defined based on nr of samples vs nr of group-receiver combinations. If manual change: example format: c(6,2,2,1) |

## Value

Ligand-Receptor Expression Product & Ligand Activities Dotplot/Heatmap

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
group_oi = "High"
prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% filter(fraction_expressing_ligand
plot_oi = make_sample_lr_prod_activity_plots(output$prioritization_tables, prioritized_tbl_oi)
plot_oi

## End(Not run)
```

---

# make_sample_lr_prod_activity_plots_Omnipath

*make_sample_lr_prod_activity_plots_Omnipath*

---

## Description

make_sample_lr_prod_activity_plots_Omnipath Visualize the scaled product of Ligand-Receptor (pseudobulk) expression per sample, and compare the different groups. In addition, show the NicheNet ligand activities in each receiver-celltype combination, AND the Omnipath curation scores of the LR pairs.

## Usage

```
make_sample_lr_prod_activity_plots_Omnipath(prioritization_tables, prioritized_tbl_oi, widths = NULL)
```

## Arguments

| | |
|---|---|
| prioritization_tables | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |
| prioritized_tbl_oi | Should be the same as in 'make_sample_lr_prod_activity_plots', except, there should be Omnipath LR quality metrics included: curation_effort, n_resources, n_references. See example here, and vignettes, on how to do this. |
| widths | Vector of 4 elements: Width of the LR exprs product panel, width of the scaled ligand activity panel, width of the cell-type specificity panel & width of the Omnipath panel. Default NULL: automatically defined based on nr of samples vs nr of group-receiver combinations. If manual change: example format: c(6,2,2,1) |

## Value

Ligand-Receptor Expression Product & Ligand Activities Dotplot/Heatmap

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
lr_network_all = readRDS("../../../NicheNet_V2/networks/data/ligand_receptor/lr_network_human_allInfo_30112033.
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl

     )
group_oi = "High"
prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% filter(fraction_expressing_ligand
plot_oi = make_sample_lr_prod_activity_plots_Omnipath(output$prioritization_tables, prioritized_tbl_oi %>% inner
plot_oi

## End(Not run)
```

make_sample_lr_prod_plots *make_sample_lr_prod_plots*

## Description

make_sample_lr_prod_plots Visualize the scaled product of Ligand-Receptor (pseudobulk) expression per sample, and compare the different groups

## Usage

```
make_sample_lr_prod_plots(prioritization_tables, prioritized_tbl_oi)
```

## Arguments

| | |
|---|---|
| prioritization_tables | Output of 'generate_prioritization_tables' or sublist in the output of 'multi_nichenet_analysis' |
| prioritized_tbl_oi | Subset of 'prioritization_tables$group_prioritization_tbl': the ligand-receptor interactions shown in this subset will be visualized: recommended to consider the top n LR interactions of a group of interest, based on the prioritization_score (eg n = 50; see vignettes for examples). |

## Value

Ligand-Receptor Expression Product Dotplot/Heatmap

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
```

```
    contrast_tbl = contrast_tbl

    )
group_oi = "High"
prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% filter(fraction_expressing_ligand
plot_oi = make_sample_lr_prod_plots(output$prioritization_tables, prioritized_tbl_oi)
plot_oi

## End(Not run)
```

---

`make_target_violin_plot` *make_target_violin_plot*

---

## Description

`make_target_violin_plot` Violin plot of a target gene of interest: per sample, and samples are grouped per group

## Usage

```
make_target_violin_plot(sce, target_oi, receiver_oi, group_oi, group_id, sample_id, celltype_id, batch
```

## Arguments

| | |
|---|---|
| `sce` | SingleCellExperiment object |
| `target_oi` | Name of the gene of interest |
| `receiver_oi` | Character vector with the names of the receiver cell type of interest |
| `group_oi` | Character vector of name of the group of interest |
| `group_id` | Name of the meta data column that indicates from which group/condition a cell comes from |
| `sample_id` | Name of the colData(sce) column in which the id of the sample is defined |
| `celltype_id` | Name of the meta data column that indicates the cell type of a cell |
| `batch_oi` | Name of a batch of interest based on which the visualization will be split. Default: NA: no batch. |
| `background_groups` | Default NULL: all groups in the group_id metadata column will be chosen. But user can fill in a character vector with the names of all gruops of interest. |

## Value

ggplot object: Violin plot of a target gene of interest: per sample, and samples are grouped per group

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl
     )
receiver_oi = "Malignant"
group_oi = "High"
target_oi = "RAB31"
p_violin_target = make_target_violin_plot(sce = sce, target_oi = target_oi, receiver_oi = receiver_oi, group_oi = g

## End(Not run)
```

---

# makenames_SCE *make.names of all genes in a SingleCellExperiment Object*

---

## Description

makenames_SCE make.names of all genes in a SingleCellExperiment Object. Avoids missing of genes that are sometimes in original symbol, and sometimes in make.names() format

## Usage

```
makenames_SCE(sce)
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |

## Value

SingleCellExperiment Object

## Examples

```
## Not run:
sce = sce %>% makenames_SCE()

## End(Not run)
```

---

multi_nichenet_analysis
*multi_nichenet_analysis*

---

## Description

multi_nichenet_analysis Perform a MultiNicheNet analysis in an all-vs-all setting.

## Usage

```
multi_nichenet_analysis(
sce, celltype_id, sample_id,group_id, batches, covariates, lr_network,ligand_target_matrix,contrasts_
assay_oi_pb ="counts",fun_oi_pb = "sum",de_method_oi = "edgeR",min_cells = 10,logFC_threshold = 0.50,p
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |
| covariates | NA if no covariates should be corrected for. If there should be corrected for covariates uring DE analysis, this argument should be the name(s) of the columns in the meta data that indicate the covariate(s). Can both be categorical and continuous. Pseudobulk expression values will not be corrected for the first element of this vector. |
| lr_network | Prior knowledge Ligand-Receptor network (columns: ligand, receptor) |
| ligand_target_matrix | Prior knowledge model of ligand-target regulatory potential (matrix with ligands in columns and targets in rows). See https://github.com/saeyslab/nichenetr. |

`contrasts_oi` String indicating the contrasts of interest (= which groups/conditions will be compared) for the differential expression and MultiNicheNet analysis. We will demonstrate here a few examples to indicate how to write this. Check the limma package manuals for more information about defining design matrices and contrasts for differential expression analysis.
If wanting to compare group A vs B: 'contrasts_oi = c("'A-B'")'
If wanting to compare group A vs B & B vs A: 'contrasts_oi = c("'A-B','B-A'")'
If wanting to compare group A vs B & A vs C & A vs D: 'contrasts_oi = c("'A-B','A-C', 'A-D'")'
If wanting to compare group A vs B and C: 'contrasts_oi = c("'A-(B+C)/2'")'
If wanting to compare group A vs B, C and D: 'contrasts_oi = c("'A-(B+C+D)/3'")'
If wanting to compare group A vs B, C and D & B vs A,C,D: 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")'
Note that the groups A, B, ... should be present in the meta data column 'group_id'.

`contrast_tbl` Data frame providing names for each of the contrasts in contrasts_oi in the 'contrast' column, and the corresponding group of interest in the 'group' column. Entries in the 'group' column should thus be present in the group_id column in the metadata. Example for 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")': 'contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))'

`senders_oi` Default NULL: all celltypes will be considered as senders. If you want to select specific senders_oi: you can add this here as character vector.

`receivers_oi` Default NULL: all celltypes will be considered as receivers. If you want to select specific receivers_oi: you can add this here as character vector.

`fraction_cutoff`
Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample.

`min_sample_prop`
Parameter to define the minimal required nr of samples in which a gene should be expressed in more than 'fraction_cutoff' of cells in that sample (per cell type). This nr of samples is calculated as the 'min_sample_prop' fraction of the nr of samples of the smallest group (after considering samples with n_cells >= 'min_cells'. Default: 'min_sample_prop = 0.50'. Examples: if there are 8 samples in the smallest group, there should be min_sample_prop*8 (= 4 in this example) samples with sufficient fraction of expressing cells.

`scenario` Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

`ligand_activity_down`
: Default: FALSE, downregulatory ligand activity is not considered for prioritization. TRUE: both up- and downregulatory activity are considered for prioritization.

`assay_oi_pb`
: Indicates which information of the assay of interest should be used (counts, scaled data,...). Default: "counts". See 'muscat::aggregateData'.

`fun_oi_pb`
: Indicates way of doing the pseudobulking. Default: "sum". See 'muscat::aggregateData'.

`de_method_oi`
: Indicates the DE method that will be used after pseudobulking. Default: "edgeR". See 'muscat::pbDS'.

`min_cells`
: Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'.

`logFC_threshold`
: For defining the gene set of interest for NicheNet ligand activity: what is the minimum logFC a gene should have to belong to this gene set? Default: 0.25/

`p_val_threshold`
: For defining the gene set of interest for NicheNet ligand activity: what is the maximam p-value a gene should have to belong to this gene set? Default: 0.05.

`p_val_adj`
: For defining the gene set of interest for NicheNet ligand activity: should we look at the p-value corrected for multiple testing? Default: FALSE.

`empirical_pval`
: For defining the gene set of interest for NicheNet ligand activity - and for ranking DE ligands and receptors: should we use the normal p-values, or the p-values that are corrected by the empirical null procedure. The latter could be beneficial if p-value distribution histograms indicate potential problems in the model definition (eg not all relevant batches are selected, etc). Default: TRUE.

`top_n_target`
: For defining NicheNet ligand-target links: which top N predicted target genes. See 'nichenetr::get_weighted_ligand_target_links()'.

`verbose`
: Indicate which different steps of the pipeline are running or not. Default: FALSE.

`n.cores`
: The number of cores used for parallel computation of the ligand activities per receiver cell type. Default: 1 - no parallel computation.

`return_lr_prod_matrix`
: Indicate whether to calculate a senderLigand-receiverReceptor matrix, which could be used for unsupervised analysis of the cell-cell communication. Default FALSE. Setting to FALSE might be beneficial to avoid memory issues.

`findMarkers`
: Indicate whether we should also calculate DE results with the classic scran::findMarkers approach. Default (recommended): FALSE. if TRUE: both pseudobulk-based and cell-level based DE results will be generated.

`top_n_LR`
: top nr of LR pairs for which correlation with target genes will be calculated. Is 2500 by default. If you want to calculate correlation for all expressed LR pairs, set this argument to NA.

## Value

List containing information and output of the MultiNicheNet analysis. celltype_info: contains average expression value and fraction of each cell type - sample combination, celltype_de: contains

output of the differential expression analysis, sender_receiver_info: links the expression information of the ligand in the sender cell types to the expression of the receptor in the receiver cell types, sender_receiver_de: links the differential information of the ligand in the sender cell types to the expression of the receptor in the receiver cell types ligand_activities_targets_DEgenes: contains the output of the NicheNet ligand activity analysis, and the NicheNet ligand-target inference prioritization_tables: contains the tables with the final prioritization scores lr_prod_mat: matrix of the ligand-receptor expression product of the expressed senderLigand-receiverReceptor pairs, grouping_tbl: data frame showing the group per sample lr_target_prior_cor: data frame showing the expression correlation between ligand-receptor pairs and DE genes + NicheNet regulatory potential scores indicating the amount of prior knowledge supporting a LR-target regulatory link

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
covariates = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     covariates = covariates,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl)

## End(Not run)
```

---

# multi_nichenet_analysis_sampleAgnostic

*multi_nichenet_analysis_sampleAgnostic*

---

## Description

`multi_nichenet_analysis_sampleAgnostic` Perform a sample-agnostic MultiNicheNet analysis in an all-vs-all setting. This means that all samples per group will be pooled.

## Usage

```
multi_nichenet_analysis_sampleAgnostic(
sce, celltype_id, sample_id,group_id, batches, covariates, lr_network,ligand_target_matrix,contrasts_
assay_oi_pb ="counts",fun_oi_pb = "sum",de_method_oi = "edgeR",min_cells = 10,logFC_threshold = 0.50,p
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |
| covariates | NA if no covariates should be corrected for. If there should be corrected for covariates uring DE analysis, this argument should be the name(s) of the columns in the meta data that indicate the covariate(s). Can both be categorical and continuous. Pseudobulk expression values will not be corrected for the first element of this vector. |
| lr_network | Prior knowledge Ligand-Receptor network (columns: ligand, receptor) |
| ligand_target_matrix | Prior knowledge model of ligand-target regulatory potential (matrix with ligands in columns and targets in rows). See https://github.com/saeyslab/nichenetr. |
| contrasts_oi | String indicating the contrasts of interest (= which groups/conditions will be compared) for the differential expression and MultiNicheNet analysis. We will demonstrate here a few examples to indicate how to write this. Check the limma package manuals for more information about defining design matrices and contrasts for differential expression analysis.<br>If wanting to compare group A vs B: 'contrasts_oi = c("'A-B'")'<br>If wanting to compare group A vs B & B vs A: 'contrasts_oi = c("'A-B','B-A'")'<br>If wanting to compare group A vs B & A vs C & A vs D: 'contrasts_oi = c("'A-B','A-C', 'A-D'")'<br>If wanting to compare group A vs B and C: 'contrasts_oi = c("'A-(B+C)/2'")'<br>If wanting to compare group A vs B, C and D: 'contrasts_oi = c("'A-(B+C+D)/3'")'<br>If wanting to compare group A vs B, C and D & B vs A,C,D: 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")'<br>Note that the groups A, B, ... should be present in the meta data column 'group_id'. |
| contrast_tbl | Data frame providing names for each of the contrasts in contrasts_oi in the 'contrast' column, and the corresponding group of interest in the 'group' column. Entries in the 'group' column should thus be present in the group_id |

column in the metadata. Example for 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")': 'contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))'

`senders_oi` Default NULL: all celltypes will be considered as senders. If you want to select specific senders_oi: you can add this here as character vector.

`receivers_oi` Default NULL: all celltypes will be considered as receivers. If you want to select specific receivers_oi: you can add this here as character vector.

`fraction_cutoff`
Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample.

`min_sample_prop`
Parameter to define the minimal required nr of samples in which a gene should be expressed in more than 'fraction_cutoff' of cells in that sample (per cell type). This nr of samples is calculated as the 'min_sample_prop' fraction of the nr of samples of the smallest group (after considering samples with n_cells >= 'min_cells'. Default: 'min_sample_prop = 0.50'. Examples: if there are 8 samples in the smallest group, there should be min_sample_prop*8 (= 4 in this example) samples with sufficient fraction of expressing cells.

`scenario` Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

`ligand_activity_down`
Default: FALSE, downregulatory ligand activity is not considered for prioritization. TRUE: both up- and downregulatory activity are considered for prioritization.

`assay_oi_pb` Indicates which information of the assay of interest should be used (counts, scaled data,...). Default: "counts". See 'muscat::aggregateData'.

`fun_oi_pb` Indicates way of doing the pseudobulking. Default: "sum". See 'muscat::aggregateData'.

`de_method_oi` Indicates the DE method that will be used after pseudobulking. Default: "edgeR". See 'muscat::pbDS'.

`min_cells` Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'.

`logFC_threshold`
For defining the gene set of interest for NicheNet ligand activity: what is the minimum logFC a gene should have to belong to this gene set? Default: 0.25/

`p_val_threshold`
For defining the gene set of interest for NicheNet ligand activity: what is the maximam p-value a gene should have to belong to this gene set? Default: 0.05.

p_val_adj For defining the gene set of interest for NicheNet ligand activity: should we look at the p-value corrected for multiple testing? Default: FALSE.

empirical_pval For defining the gene set of interest for NicheNet ligand activity - and for ranking DE ligands and receptors: should we use the normal p-values, or the p-values that are corrected by the empirical null procedure. The latter could be beneficial if p-value distribution histograms indicate potential problems in the model definition (eg not all relevant batches are selected, etc). Default: TRUE.

top_n_target For defining NicheNet ligand-target links: which top N predicted target genes. See 'nichenetr::get_weighted_ligand_target_links()'.

verbose Indicate which different steps of the pipeline are running or not. Default: FALSE.

n.cores The number of cores used for parallel computation of the ligand activities per receiver cell type. Default: 1 - no parallel computation.

return_lr_prod_matrix
Indicate whether to calculate a senderLigand-receiverReceptor matrix, which could be used for unsupervised analysis of the cell-cell communication. Default FALSE. Setting to FALSE might be beneficial to avoid memory issues.

top_n_LR top nr of LR pairs for which correlation with target genes will be calculated. Is 2500 by default. If you want to calculate correlation for all expressed LR pairs, set this argument to NA.

## Value

List containing information and output of the MultiNicheNet analysis. celltype_info: contains average expression value and fraction of each cell type - sample combination, celltype_de: contains output of the differential expression analysis, sender_receiver_info: links the expression information of the ligand in the sender cell types to the expression of the receptor in the receiver cell types, sender_receiver_de: links the differential information of the ligand in the sender cell types to the expression of the receptor in the receiver cell types ligand_activities_targets_DEgenes: contains the output of the NicheNet ligand activity analysis, and the NicheNet ligand-target inference prioritization_tables: contains the tables with the final prioritization scores lr_prod_mat: matrix of the ligand-receptor expression product of the expressed senderLigand-receiverReceptor pairs, grouping_tbl: data frame showing the group per sample lr_target_prior_cor: data frame showing the expression correlation between ligand-receptor pairs and DE genes + NicheNet regulatory potential scores indicating the amount of prior knowledge supporting a LR-target regulatory link

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
covariates = NA
contrasts_oi = c("'High-Low','Low-High'")
```

```
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis_sampleAgnostic(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     covariates = covariates,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl)

## End(Not run)
```

---

`p.adjust_empirical` *Get empirical p-values and adjusted p-values.*

---

## Description

`p.adjust_empirical` Get emperical p-values and adjusted p-values. Credits to Jeroen Gillis (cf satuRn package)

## Usage

```
p.adjust_empirical(pvalues, tvalues, plot = FALSE, celltype = NULL, contrast = NULL)
```

## Arguments

| | |
|---|---|
| `pvalues` | Vector of original p-values |
| `tvalues` | Vector of original t-values: in case of DE analysis: log fold changes |
| `plot` | TRUE or FALSE (default): should we plot the z-score distribution? |
| `celltype` | NULL, or name of the cell type of interest - this will be added to the plot title if plot = TRUE |
| `contrast` | NULL, or name of the contrast of interest - thhis will be added to the plot title if plot = TRUE |

## Value

List with empirical p-values and adjusted p-values + ggplot output and estimated delta and sigma.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)
de_output_tidy = muscat::resDS(celltype_de$sce, celltype_de$de_output, bind = "row", cpm = FALSE, frq = FALSE) %>%
emp_res = p.adjust_empirical(de_output_tidy %>% pull(p_val), de_output_tidy  %>% pull(p_val), plot = T)

## End(Not run)
```

---

`perform_muscat_de_analysis`

*perform_muscat_de_analysis*

---

## Description

`perform_muscat_de_analysis` Perform differential expression analysis via Muscat - Pseudobulking approach - FOR ONE CELLTYPE.

## Usage

```
perform_muscat_de_analysis(sce, sample_id, celltype_id, group_id, batches, covariates, contrasts, expr
```

## Arguments

| | |
|---|---|
| `sce` | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| `sample_id` | Name of the meta data column that indicates from which sample/patient a cell comes from |
| `celltype_id` | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| `group_id` | Name of the meta data column that indicates from which group/condition a cell comes from |

batches — NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector.

covariates — NA if no covariates should be corrected for. If there should be corrected for covariates uring DE analysis, this argument should be the name(s) of the columns in the meta data that indicate the covariate(s). Can both be categorical and continuous. Pseudobulk expression values will not be corrected for the first element of this vector.

contrasts — String indicating the contrasts of interest (= which groups/conditions will be compared) for the differential expression and MultiNicheNet analysis. We will demonstrate here a few examples to indicate how to write this. Check the limma package manuals for more information about defining design matrices and contrasts for differential expression analysis. If wanting to compare group A vs B: 'contrasts_oi = c("'A-B'")' If wanting to compare group A vs B & B vs A: 'contrasts_oi = c("'A-B','B-A'")' If wanting to compare group A vs B & A vs C & A vs D: 'contrasts_oi = c("'A-B','A-C', 'A-D'")' If wanting to compare group A vs B and C: 'contrasts_oi = c("'A-(B+C)/2'")' If wanting to compare group A vs B, C and D: 'contrasts_oi = c("'A-(B+C+D)/3'")' If wanting to compare group A vs B, C and D & B vs A,C,D: 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")' Note that the groups A, B, ... should be present in the meta data column 'group_id'.

expressed_df — tibble with three columns: gene, celltype, expressed; this data frame indicates which genes can be considered as expressed in each cell type.

assay_oi_pb — Indicates which information of the assay of interest should be used (counts, scaled data,...). Default: "counts". See 'muscat::aggregateData'.

fun_oi_pb — Indicates way of doing the pseudobulking. Default: "sum". See 'muscat::aggregateData'.

de_method_oi — Indicates the DE method that will be used after pseudobulking. Default: "edgeR". See 'muscat::pbDS'.

min_cells — Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'.

## Value

List with output of the differential expression analysis in 1) default format('muscat::pbDS()'), and 2) in a tidy table format ('muscat::resDS()').

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "CAF"
```

```
batches = NA
covariates = NA
contrasts_oi = c("'High-Low','Low-High'")
frq_list = get_frac_exprs(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = group_id)
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   covariates = covariates,
   contrasts = contrasts_oi,
   expressed_df = frq_list$expressed_df)

## End(Not run)
```

---

`prioritize_condition_specific_receiver` *prioritize_condition_specific_receiver*

---

## Description

`prioritize_condition_specific_receiver` Perform the MultiNicheNet prioritization of cell-cell interactions. Focus on including condition-specific cell types as receiver cells. This implies no DE information will be used for prioritization of receptors, nor ligand activities for ligands

## Usage

```
prioritize_condition_specific_receiver(abundance_info, abundance_expression_info, condition_specific
```

## Arguments

| | |
|---|---|
| `abundance_info` | Output from 'make_abundance_plots' |
| `abundance_expression_info` | Output from 'get_abundance_expression_info' |
| `condition_specific_celltypes` | Character vector of condition-specific cell types |
| `grouping_tbl` | Data frame showing the groups of each sample (and batches per sample if applicable) (columns: sample and group; and if applicable all batches of interest) |
| `fraction_cutoff` | Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample. |
| `contrast_tbl` | Data frame providing names for each of the contrasts in contrasts_oi in the 'contrast' column, and the corresponding group of interest in the 'group' column. Entries in the 'group' column should thus be present in the group_id |

column in the metadata. Example for 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")': 'contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))'

`sender_receiver_de`
: Output of 'combine_sender_receiver_de'

`lr_network`
: Prior knowledge Ligand-Receptor network (columns: ligand, receptor)

`ligand_activities_targets_DEgenes`
: Output of 'get_ligand_activities_targets_DEgenes'

`scenario`
: Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

`ligand_activity_down`
: For prioritization based on ligand activity: consider the max of up- and downregulation ('TRUE') or consider only upregulated activity ('FALSE', default from version 2 on).

## Value

List containing multiple data frames of prioritized senderLigand-receiverReceptor interactions (with sample- and group-based expression information), ligand activities and ligand-target links.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")
```

```
celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

condition_specific_celltypes = "CAFs"
abundance_info = make_abundance_plots(sce = sce, sample_id = sample_id, group_id = group_id, celltype_id =  celltyp
prioritization_tables_with_condition_specific_celltype_receiver = prioritize_condition_specific_receiver(
abundance_info = abundance_info,
abundance_expression_info = abundance_expression_info,
condition_specific_celltypes = condition_specific_celltypes,
grouping_tbl = grouping_tbl,
fraction_cutoff = fraction_cutoff,
```

```
contrast_tbl = contrast_tbl,
sender_receiver_de = sender_receiver_de,
lr_network = lr_network,
ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
scenario = "regular",
ligand_activity_down = FALSE
)



## End(Not run)
```

---

`prioritize_condition_specific_sender` *prioritize_condition_specific_sender*

---

## Description

`prioritize_condition_specific_sender` Perform the MultiNicheNet prioritization of cell-cell interactions. Focus on including condition-specific cell types as sender cells. This implies no DE information will be used for prioritization of ligands.

## Usage

```
prioritize_condition_specific_sender(abundance_info, abundance_expression_info, condition_specific_c
```

## Arguments

| | |
|---|---|
| `abundance_info` | Output from 'make_abundance_plots' |
| `abundance_expression_info` | Output from 'get_abundance_expression_info' |
| `condition_specific_celltypes` | Character vector of condition-specific cell types |
| `grouping_tbl` | Data frame showing the groups of each sample (and batches per sample if applicable) (columns: sample and group; and if applicable all batches of interest) |
| `fraction_cutoff` | Cutoff indicating the minimum fraction of cells of a cell type in a specific sample that are necessary to consider a gene (e.g. ligand/receptor) as expressed in a sample. |
| `contrast_tbl` | Data frame providing names for each of the contrasts in contrasts_oi in the 'contrast' column, and the corresponding group of interest in the 'group' column. Entries in the 'group' column should thus be present in the group_id column in the metadata. Example for 'contrasts_oi = c("'A-(B+C+D)/3', 'B-(A+C+D)/3'")': 'contrast_tbl = tibble(contrast = c("A-(B+C+D)/3","B-(A+C+D)/3"), group = c("A","B"))' |

sender_receiver_de
: Output of 'combine_sender_receiver_de'

lr_network
: Prior knowledge Ligand-Receptor network (columns: ligand, receptor)

ligand_activities_targets_DEgenes
: Output of 'get_ligand_activities_targets_DEgenes'

scenario
: Character vector indicating which prioritization weights should be used during the MultiNicheNet analysis. Currently 3 settings are implemented: "regular" (default), "lower_DE", and "no_frac_LR_expr". The setting "regular" is strongly recommended and gives each criterion equal weight. The setting "lower_DE" is recommended in cases your hypothesis is that the differential CCC patterns in your data are less likely to be driven by DE (eg in cases of differential migration into a niche). It halves the weight for DE criteria, and doubles the weight for ligand activity. "no_frac_LR_expr" is the scenario that will exclude the criterion "fraction of samples expressing the LR pair'. This may be beneficial in case of few samples per group.

ligand_activity_down
: For prioritization based on ligand activity: consider the max of up- and downregulation ('TRUE') or consider only upregulated activity ('FALSE', default from version 2 on).

## Value

List containing multiple data frames of prioritized senderLigand-receiverReceptor interactions (with sample- and group-based expression information), ligand activities and ligand-target links.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
abundance_data_receiver = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
abundance_data_sender = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")


celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g

receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
```

```
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
sender_receiver_info = combine_sender_receiver_info_ic(sender_info = sender_info_ic,receiver_info = receiver_inf

celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)

sender_receiver_de = combine_sender_receiver_de(
 sender_de = celltype_de,
 receiver_de = celltype_de,
 senders_oi = senders_oi,
 receivers_oi = receivers_oi,
 lr_network = lr_network)

ligand_activities_targets_DEgenes = get_ligand_activities_targets_DEgenes(
   receiver_de = celltype_de,
   receivers_oi = receivers_oi,
   receiver_frq_df_group = celltype_info$frq_df_group,
   ligand_target_matrix = ligand_target_matrix)


sender_receiver_tbl = sender_receiver_de %>% dplyr::distinct(sender, receiver)
metadata_combined = SummarizedExperiment::colData(sce) %>% tibble::as_tibble()
grouping_tbl = metadata_combined[,c(sample_id, group_id)] %>% tibble::as_tibble() %>% dplyr::distinct()
colnames(grouping_tbl) = c("sample","group")

frac_cutoff = 0.05
prioritization_tables = generate_prioritization_tables(
    sender_receiver_info = sender_receiver_info,
    sender_receiver_de = sender_receiver_de,
    ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
    contrast_tbl = contrast_tbl,
    sender_receiver_tbl = sender_receiver_tbl,
    grouping_tbl = grouping_tbl,
    fraction_cutoff = frac_cutoff, abundance_data_receiver, abundance_data_sender)

condition_specific_celltypes = "CAFs"
abundance_info = make_abundance_plots(sce = sce, sample_id = sample_id, group_id = group_id, celltype_id =  celltyp
prioritization_tables_with_condition_specific_celltype_sender = prioritize_condition_specific_sender(
abundance_info = abundance_info
abundance_expression_info = abundance_expression_info,
condition_specific_celltypes = condition_specific_celltypes,
grouping_tbl = grouping_tbl,
fraction_cutoff = fraction_cutoff,
contrast_tbl = contrast_tbl,
sender_receiver_de = sender_receiver_de,
lr_network = lr_network,
```

```
ligand_activities_targets_DEgenes = ligand_activities_targets_DEgenes,
scenario = "regular",
ligand_activity_down = FALSE
)



## End(Not run)
```

---

process_abund_info *process_abund_info*

---

## Description

process_abund_info Process cell type abundance information into intercellular communication focused information.

## Usage

```
process_abund_info(abund_data, ic_type = "sender")
```

## Arguments

| | |
|---|---|
| abund_data | Data frame with number of cells per cell type - sample combination |
| ic_type | "sender" or "receiver": indicates whether we should rename celltype to sender or receiver. |

## Value

Data frame with number of cells per cell type - sample combination; but now adapted to sender/receiver nomenclature

## Examples

```
## Not run:
library(dplyr)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
min_cells = 10
metadata_abundance = SummarizedExperiment::colData(sce)[,c(sample_id, group_id, celltype_id)]
colnames(metadata_abundance) =c("sample_id", "group_id", "celltype_id")
abundance_data = metadata_abundance %>% tibble::as_tibble() %>% dplyr::group_by(sample_id , celltype_id) %>% dplyr
abundance_data = abundance_data %>% dplyr::mutate(keep = n >= min_cells) %>% dplyr::mutate(keep = factor(keep, leve
receiver_abundance_data = process_info_to_ic(abund_data = abundance_data, ic_type = "receiver")
sender_abundance_data = process_info_to_ic(abund_data = abundance_data, ic_type = "sender")

## End(Not run)
```

process_abundance_expression_info
*process_abundance_expression_info*

## Description

process_abundance_expression_info Visualize cell type abundances. Calculate the average and fraction of expression of each gene per sample and per group. Calculate relative abundances of cell types as well. Under the hood, the following functions are used: 'get_avg_frac_exprs_abund', 'process_info_to_ic', 'combine_sender_receiver_info_ic'

## Usage

```
process_abundance_expression_info(sce, sample_id, group_id, celltype_id, min_cells = 10, senders_oi, r
```

## Arguments

| | |
|---|---|
| sce | SingleCellExperiment object of the scRNAseq data of interest. Contains both sender and receiver cell types. |
| sample_id | Name of the meta data column that indicates from which sample/patient a cell comes from |
| group_id | Name of the meta data column that indicates from which group/condition a cell comes from |
| celltype_id | Name of the column in the meta data of sce that indicates the cell type of a cell. |
| min_cells | Indicates the minimal number of cells that a sample should have to be considered in the DE analysis. Default: 10. See 'muscat::pbDS'. |
| senders_oi | Default NULL: all celltypes will be considered as senders. If you want to select specific senders_oi: you can add this here as character vector. |
| receivers_oi | Default NULL: all celltypes will be considered as receivers. If you want to select specific receivers_oi: you can add this here as character vector. |
| lr_network | Prior knowledge Ligand-Receptor network (columns: ligand, receptor) |
| batches | NA if no batches should be corrected for. If there should be corrected for batches during DE analysis and pseudobulk expression calculation, this argument should be the name(s) of the columns in the meta data that indicate the batch(s). Should be categorical. Pseudobulk expression values will be corrected for the first element of this vector. |
| frq_list | output of 'get_frac_exprs' |
| rel_abundance_df | 'rel_abundance_df' slot of 'get_abundance_info()' output. |

## Value

List containing data frames with average and fraction of expression per sample and per group, and relative cell type abundances as well.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
senders_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
abundance_info = get_abundance_info(sce = sce, sample_id = sample_id, group_id = group_id, celltype_id = celltype_
frq_list = get_frac_exprs(sce = sce, sample_id = sample_id, celltype_id = celltype_id, group_id = group_id)
abundance_celltype_info = process_abundance_expression_info(sce = sce, sample_id = sample_id, group_id = group_id,

## End(Not run)
```

---

# process_geneset_data *process_geneset_data*

---

## Description

process_geneset_data Determine ratio's of geneset_oi vs background for a certain logFC/p-val thresholds setting.

## Usage

```
process_geneset_data(contrast_oi, receiver_de, logFC_threshold = 0.5, p_val_adj = FALSE, p_val_thresho
```

## Arguments

| | |
|---|---|
| contrast_oi | Name of one of the contrasts in the celltype_DE / receiver_DE output tibble. |
| receiver_de | Differential expression analysis output for the receiver cell types. 'de_output_tidy' slot of the output of 'perform_muscat_de_analysis'. |
| logFC_threshold | For defining the gene set of interest for NicheNet ligand activity: what is the minimum logFC a gene should have to belong to this gene set? Default: 0.25/ |
| p_val_adj | For defining the gene set of interest for NicheNet ligand activity: should we look at the p-value corrected for multiple testing? Default: FALSE. |
| p_val_threshold | For defining the gene set of interest for NicheNet ligand activity: what is the maximam p-value a gene should have to belong to this gene set? Default: 0.05. |

## Value

Tibble indicating the nr of up- and down genes per contrast/cell type combination, and an indication whether this is in the recommended ranges

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
receivers_oi = SummarizedExperiment::colData(sce)[,celltype_id] %>% unique()
celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g
celltype_de = perform_muscat_de_analysis(
   sce = sce,
   sample_id = sample_id,
   celltype_id = celltype_id,
   group_id = group_id,
   batches = batches,
   contrasts = contrasts_oi)
receiver_de = celltype_de$de_output_tidy
geneset_assessment = contrast_tbl$contrast %>%
lapply(process_geneset_data, receiver_de) %>% bind_rows()

## End(Not run)
```

---

`process_info_to_ic` *process_info_to_ic*

---

## Description

`process_info_to_ic` Process cell type expression information into intercellular communication focused information. Only keep information of ligands for the sender cell type setting, and information of receptors for the receiver cell type.

## Usage

```
process_info_to_ic(info_object, ic_type = "sender", lr_network)
```

## Arguments

| | |
|---|---|
| `info_object` | Output of 'get_avg_frac_exprs_abund' |
| `ic_type` | "sender" or "receiver": indicates whether we should keep ligands or receptors respectively. |
| `lr_network` | Prior knowledge Ligand-Receptor network (columns: ligand, receptor) |

## Value

List with expression information of ligands (sender case) or receptors (receiver case) - similar to output of 'get_avg_frac_exprs_abund'.

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
celltype_info = get_avg_frac_exprs_abund(sce = sce, sample_id = sample_id, celltype_id =  celltype_id, group_id = g
receiver_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "receiver", lr_network = lr_network)
sender_info_ic = process_info_to_ic(info_object = celltype_info, ic_type = "sender", lr_network = lr_network)

## End(Not run)
```

---

# sce *SingleCellExperiment object containing scRNAseq data (subsampled)*

---

## Description

SingleCellExperiment object containing scRNAseq data (subsampled). Source of the data: Puram et al., Cell 2017: "Single-Cell Transcriptomic Analysis of Primary and Metastatic Tumor Ecosystems in Head and Neck Cancer.". This example data was downsampled (features and cells). Interesting metadata/colData columns: tumor, pEMT, pEMT_fine, celltype, batch

## Usage

```
sce
```

## Format

An object of class SingleCellExperiment

visualize_network *visualize_network*

## Description

visualize_network Visualize a network showing the gene regulatory links between ligands from sender cell types to their induced ligands/receptors in receiver cell types. Links are only drawn if the ligand/receptor in the receiver is a potential downstream target of the ligand (based on prior knowledge, and optionally with sufficient correlation in expression across the different samples).

## Usage

```
visualize_network(network, colors)
```

## Arguments

| | |
|---|---|
| network | Output of 'infer_intercellular_regulatory_network' |
| colors | Named vector of colors associated to each sender cell type. Vector = color, names = sender names. |

## Value

ggplot object with plot of LR–>Target links, together with the tidygraph and igraph objects themselves

## Examples

```
## Not run:
library(dplyr)
lr_network = readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network = lr_network %>% dplyr::rename(ligand = from, receptor = to) %>% dplyr::distinct(ligand, receptor)
ligand_target_matrix = readRDS(url("https://zenodo.org/record/3260758/files/ligand_target_matrix.rds"))
sample_id = "tumor"
group_id = "pEMT"
celltype_id = "celltype"
batches = NA
contrasts_oi = c("'High-Low','Low-High'")
contrast_tbl = tibble(contrast = c("High-Low","Low-High"), group = c("High","Low"))
output = multi_nichenet_analysis(
     sce = sce,
     celltype_id = celltype_id,
     sample_id = sample_id,
     group_id = group_id,
     batches = batches,
     lr_network = lr_network,
     ligand_target_matrix = ligand_target_matrix,
     contrasts_oi = contrasts_oi,
     contrast_tbl = contrast_tbl
```

```
    )
lr_target_prior_cor_filtered = output$lr_target_prior_cor %>% filter(scaled_prior_score > 0.50 & (pearson > 0.66 |
 prioritized_tbl_oi = output$prioritization_tables$group_prioritization_tbl %>% distinct(id, ligand, receptor, se
 prioritized_tbl_oi = prioritized_tbl_oi %>% filter(id %in% lr_target_prior_cor_filtered$id)
 prioritized_tbl_oi = prioritized_tbl_oi %>% group_by(ligand, sender, group) %>% top_n(2, prioritization_score)
 lr_target_df = lr_target_prior_cor_filtered  %>% distinct(group, sender, receiver, ligand, receptor, id, target, c
 network = infer_intercellular_regulatory_network(lr_target_df, prioritized_tbl_oi)
graph_plot = visualize_network(network, colors = c("purple","orange"))
graph_plot$plot

## End(Not run)
```

# Index