Package ‘EWCE’

May 27, 2024

Type Package
Title Expression Weighted Celltype Enrichment
Version 1.12.0
Description Used to determine which cell types are enriched within gene lists. The package provides tools for testing enrichments within simple gene lists (such as human disease associated genes) and those resulting from differential expression studies. The package does not depend upon any particular Single Cell Transcriptome dataset and user defined datasets can be loaded in and used in the analyses.

URL https://github.com/NathanSkene/EWCE
BugReports https://github.com/NathanSkene/EWCE/issues
License GPL-3
Depends R (>= 4.2), RNOmni (>= 1.0)
VignetteBuilder knitr
Imports stats, utils, methods, ewceData (>= 1.7.1), dplyr, ggplot2, reshape2, limma, stringr, HGNChelper, Matrix, parallel, SingleCellExperiment, SummarizedExperiment, DelayedArray, BiocParallel, orthogene (>= 0.99.8), data.table
Suggests rworkflows, remotes, knitr, BiocStyle, markdown, testthat (>= 3.0.0), readxl, memoise, markdown, sctransform, DESeq2, MAST, DelayedMatrixStats, ggdendro, scales, patchwork
biocViews GeneExpression, Transcription, DifferentialExpression, GeneSetEnrichment, Genetics, Microarray, mRNAMicroarray, OneChannel, RNASeq, BiomedicalInformatics, Proteomics, Visualization, FunctionalGenomics, SingleCell
RoxygenNote 7.2.3
Encoding UTF-8
Config/testthat/edition 3
git_url https://git.bioconductor.org/packages/EWCE
git_branch RELEASE_3_19
git_last_commit d697d5e
Contents

EWCE-package .......................................................... 4
add_res_to_merging_list ............................................ 5
assign_cores .......................................................... 6
bin_columns_into_quantiles ......................................... 6
bin_specificity_into_quantiles ..................................... 7
bootstrap_enrichment_test .......................................... 8
bootstrap_plot ......................................................... 11
bootstrap_plots_for_transcriptome .................................. 12
calculate_meancexp_for_level ...................................... 12
calculate_specificity_for_level .................................... 13
cell_list_dist ........................................................ 13
check_annotLevels .................................................... 14
check_args_for_bootstrap_plot_generation ...................... 14
check_bootstrap_args ............................................... 15
check_controlled_args ............................................... 16
check_ewce_expression_data_args .................................. 17
check_ewce_genelist_inputs ........................................ 18
check_full_results ................................................... 20
check_generate_controlled_bootstrap_geneset .................... 20
check_group_name ..................................................... 21
check_nas ............................................................ 21
check_numeric ........................................................ 22
check_percent_hits ................................................... 22
check_sce ............................................................ 23
check_species ........................................................ 23
compute_gene_counts ............................................... 24
compute_gene_scores ................................................. 25
controlled_geneset_enrichment ..................................... 26
convert_new_ewce_to_old ............................................ 28
convert_old_ewce_to_new ............................................ 28
create_background_multilist ....................................... 29
create_list_network .................................................. 30
ctd_to_sce ........................................................... 30
delayedarray_normalize .............................................. 31
drop_nonexpressed_cells ............................................ 32
drop_nonexpressed_genes ............................................ 32
### Contents

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>drop_uninformative_genes</td>
<td>33</td>
</tr>
<tr>
<td>dt_to_df</td>
<td>36</td>
</tr>
<tr>
<td>ewce_expression_data</td>
<td>37</td>
</tr>
<tr>
<td>ewce_plot</td>
<td>39</td>
</tr>
<tr>
<td>example_bootstrap_results</td>
<td>40</td>
</tr>
<tr>
<td>example_transcriptome_results</td>
<td>41</td>
</tr>
<tr>
<td>extract_matrix</td>
<td>42</td>
</tr>
<tr>
<td>filter_crd_genes</td>
<td>45</td>
</tr>
<tr>
<td>filter_genes_without_1to1_homolog</td>
<td>46</td>
</tr>
<tr>
<td>filter_nonorthologs</td>
<td>47</td>
</tr>
<tr>
<td>filter_variance_quantiles</td>
<td>50</td>
</tr>
<tr>
<td>fix_bad_hgnc_symbols</td>
<td>51</td>
</tr>
<tr>
<td>fix_bad_mgi_symbols</td>
<td>52</td>
</tr>
<tr>
<td>fix_celltype_names</td>
<td>53</td>
</tr>
<tr>
<td>fix_celltype_names_full_results</td>
<td>54</td>
</tr>
<tr>
<td>generate_bootstrap_plots</td>
<td>54</td>
</tr>
<tr>
<td>generate_bootstrap_plots_for_transcriptome</td>
<td>57</td>
</tr>
<tr>
<td>generate_celltype_data</td>
<td>60</td>
</tr>
<tr>
<td>generate_controlled_bootstrap_geneset</td>
<td>64</td>
</tr>
<tr>
<td>get_celltype_table</td>
<td>65</td>
</tr>
<tr>
<td>get_crd_levels</td>
<td>65</td>
</tr>
<tr>
<td>get_crd_matrix_names</td>
<td>66</td>
</tr>
<tr>
<td>get_exp_data_for_bootstrapped_genes</td>
<td>66</td>
</tr>
<tr>
<td>get_sig_results</td>
<td>67</td>
</tr>
<tr>
<td>get_summed_proportions</td>
<td>68</td>
</tr>
<tr>
<td>is_32bit</td>
<td>69</td>
</tr>
<tr>
<td>is_celltypedataset</td>
<td>70</td>
</tr>
<tr>
<td>is_crd_standardised</td>
<td>70</td>
</tr>
<tr>
<td>is_delayed_array</td>
<td>71</td>
</tr>
<tr>
<td>is_matrix</td>
<td>71</td>
</tr>
<tr>
<td>is_sparse_matrix</td>
<td>72</td>
</tr>
<tr>
<td>list_species</td>
<td>72</td>
</tr>
<tr>
<td>load_rdata</td>
<td>73</td>
</tr>
<tr>
<td>max_crd_depth</td>
<td>73</td>
</tr>
<tr>
<td>merged_crd</td>
<td>74</td>
</tr>
<tr>
<td>merge_crd</td>
<td>75</td>
</tr>
<tr>
<td>merge_sce</td>
<td>77</td>
</tr>
<tr>
<td>merge_sce_list</td>
<td>78</td>
</tr>
<tr>
<td>merge_two_expfiles</td>
<td>79</td>
</tr>
<tr>
<td>messager</td>
<td>80</td>
</tr>
<tr>
<td>message_parallel</td>
<td>80</td>
</tr>
<tr>
<td>myScalesComma</td>
<td>81</td>
</tr>
<tr>
<td>plot_crd</td>
<td>81</td>
</tr>
<tr>
<td>plot_log_bootstrap_distributions</td>
<td>82</td>
</tr>
<tr>
<td>plot_with_bootstrap_distributions</td>
<td>82</td>
</tr>
<tr>
<td>prep.dendro</td>
<td>83</td>
</tr>
<tr>
<td>prepare_genesize_control_network</td>
<td>84</td>
</tr>
<tr>
<td>prepare_tt</td>
<td>85</td>
</tr>
</tbody>
</table>
**Description**

Used to determine which cell types are enriched within gene lists. The package provides tools for testing enrichments within simple gene lists (such as human disease associated genes) and those resulting from differential expression studies. The package does not depend upon any particular Single Cell Transcriptome dataset and user defined datasets can be loaded in and used in the analyses.

**Details**

**EWCE: Expression Weighted Celltype Enrichment**

Used to determine which cell types are enriched within gene lists. The package provides tools for testing enrichments within simple gene lists (such as human disease associated genes) and those resulting from differential expression studies.

The package does not depend upon any particular Single Cell Transcriptome dataset and user defined datasets can be loaded in and used in the analyses.

**Author(s)**

**Maintainer**: Alan Murphy <alanmurph94@hotmail.com> (ORCID)

Authors:

- Brian Schilder <brian_schilder@alumni.brown.edu> (ORCID)
- Nathan Skene <nathan.skene@gmail.com> (ORCID)
add_res_to_merging_list

See Also

Useful links:

- https://github.com/NathanSkene/EWCE
- Report bugs at https://github.com/NathanSkene/EWCE/issues

add_res_to_merging_list

Add to results to merging list

Description

add_res_to_merging_list adds EWCE results to a list for merging analysis.

Usage

add_res_to_merging_list(full_res, existing_results = NULL)

Arguments

full_res Results list generated using bootstrap_enrichment_test or ewce_expression_data functions. Multiple results tables can be merged into one results table, as long as the 'list' column is set to distinguish them.

existing_results Output of previous rounds from adding results to list. Leave empty if this is the first item in the list.

Value

Merged results list.

Examples

# Load the single cell data
csd <- ewceData::ctd()

# Load the data
tt_alzh <- ewceData::tt_alzh()
# tt_alzh_BA36 <- ewceData::tt_alzh_BA36()
# Use 3 bootstrap lists for speed, for publishable analysis use >10000
reps <- 3
# Use 5 up/down regulated genes (thresh) for speed, default is 250
thresh <- 5
# Run EWCE analysis
# tt_results <- ewce_expression_data(
#   sct_data = csd, tt = tt_alzh, annotLevel = 1, thresh = thresh,
#   reps = reps, ttSpecies = "human", sctSpecies = "mouse"
# )
# tt_results_36 <- ewce_expression_data(
# sct_data = ctd, tt = tt_alzh_BA36, annotLevel = 1, thresh = thresh,
# reps = reps, ttSpecies = "human", sctSpecies = "mouse"
# )

# Fill a list with the results
results <- add_res_to_merging_list(tt_alzh)
# results <- add_res_to_merging_list(tt_alzh_BA36, results)

desc_table

<table>
<thead>
<tr>
<th>assign_cores</th>
<th>Assign cores</th>
</tr>
</thead>
</table>

Description

Assign cores automatically for parallel processing, while reserving some.

Usage

assign_cores(worker_cores = 0.9, verbose = TRUE)

Arguments

- worker_cores: Number (>1) or proportion (<1) of worker cores to use.
- verbose: Print messages.

Value

List of core allocations.

---

bin_columns_into_quantiles

bin_columns_into_quantiles

Description

bin_columns_into_quantiles is an internal function used to convert a vector of specificity into a vector of specificity quantiles. This function can be iterated across a matrix using apply to create a matrix of specificity quantiles.

Usage

bin_columns_into_quantiles(
  vec,
  numberOfBins = 40,
  defaultBin = as.integer(numberOfBins/2)
)
**Arguments**

- **vec**: The vector of gene of specificity values.
- **numberOfBins**: Number of quantile bins to use (40 is recommended).
- **defaultBin**: Which bin to assign when there’s only one non-zero quantile. In situations where there’s only one non-zero quantile, `cut` throws an error. Avoid these situations by using a default quantile.

**Value**

A vector with same length as `vec` but with columns storing quantiles instead of specificity.

**Examples**

```r
ctd <- ewceData::ctd()
ctd[[1]]$specificity_quantiles <- apply(ctd[[1]]$specificity, 2,
    FUN = bin_columns_into_quantiles)
```

---

**Description**

`bin_specificity_into_quantiles` is an internal function used to convert add `$specificity_quantiles` to a ctd.

**Usage**

```r
bin_specificity_into_quantiles(
    ctdIN,
    numberOfBins,
    matrix_name = "specificity_quantiles",
    as_sparse = TRUE,
    verbose = TRUE
)
```

**Arguments**

- **ctdIN**: A single annotLevel of a ctd, i.e. `ctd[[1]]` (the function is intended to be used via `apply`).
- **numberOfBins**: Number of quantile ‘bins’ to use (40 is recommended).
- **matrix_name**: Name of the specificity matrix to create (default: "specificity_quantiles").
- **as_sparse**: Convert to `sparseMatrix`.
- **verbose**: Print messages.
Value

A ctd with "specificity_quantiles" matrix in each level (or whatever matrix_name was set to.).

Examples

ctd <- ewceData::ctd()
ctd <- lapply(ctd, EWCE::bin_specificity_into_quantiles, numberOfBins = 40)
print(ctd[[1]]$specificity_quantiles[1:3, ])

bootstrap_enrichment_test

Bootstrap cell type enrichment test

Description

bootstrap_enrichment_test takes a genelist and a single cell type transcriptome dataset and determines the probability of enrichment and fold changes for each cell type.

Usage

bootstrap_enrichment_test(
  sct_data = NULL,
  hits = NULL,
  bg = NULL,
  genelistSpecies = NULL,
  sctSpecies = NULL,
  sctSpecies_origin = sctSpecies,
  output_species = "human",
  method = "homologene",
  reps = 100,
  no_cores = 1,
  annotLevel = 1,
  geneSizeControl = FALSE,
  controlledCT = NULL,
  mtc_method = "BH",
  sort_results = TRUE,
  standardise_sct_data = TRUE,
  standardise_hits = FALSE,
  verbose = TRUE,
  localHub = FALSE,
  store_gene_data = TRUE
)
Arguments

**sct_data** List generated using `generate_celltype_data`.

**hits** List of gene symbols containing the target gene list. Will automatically be converted to human gene symbols if `geneSizeControl=TRUE`.

**bg** List of gene symbols containing the background gene list (including hit genes). If bg=NULL, an appropriate gene background will be created automatically.

**genelistSpecies** Species that hits genes came from (no longer limited to just "mouse" and "human"). See `list_species` for all available species.

**sctSpecies** Species that sct_data is currently formatted as (no longer limited to just "mouse" and "human"). See `list_species` for all available species.

**sctSpecies_origin** Species that the sct_data originally came from, regardless of its current gene format (e.g. it was previously converted from mouse to human gene orthologs). This is used for computing an appropriate background.

**output_species** Species to convert sct_data and hits to (Default: "human"). See `list_species` for all available species.

**method** R package to use for gene mapping:

- "gprofiler": Slower but more species and genes.
- "homologene": Faster but fewer species and genes.
- "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on several different data sources.

**reps** Number of random gene lists to generate (Default: 100, but should be >10,000 for publication-quality results).

**no_cores** Number of cores to parallelise bootstrapping reps over.

**annotLevel** An integer indicating which level of sct_data to analyse (Default: 1).

**geneSizeControl** Whether you want to control for GC content and transcript length. Recommended if the gene list originates from genetic studies (Default: FALSE). If set to TRUE, then hits must be from humans.

**controlledCT** [Optional] If not NULL, and instead is the name of a cell type, then the bootstrapping controls for expression within that cell type.

**mtc_method** Multiple-testing correction method (passed to `p.adjust`).

**sort_results** Sort enrichment results from smallest to largest p-values.

**standardise_sct_data**

- When `sctSpecies!=output_species` the sct_data will be checked for object formatting and the genes will be converted to the orthologs of the output_species with `standardise_ctd` (which calls `map_genes` internally).
- When `sctSpecies==output_species`, the sct_data will be checked for object formatting with `standardise_ctd`, but the gene names will remain untouched.
standardise_hits

Should hits be standardised? If TRUE:

- When genelistSpecies!=output_species, the genes will be converted to the orthologs of the output_species with convert_orthologs.
- When genelistSpecies==output_species, the genes will be standardised with map_genes.

If FALSE, hits will be passed on to subsequent steps as-is.

verbose
Print messages.

localHub
If working offline, add argument localHub=TRUE to work with a local, non-updated hub; It will only have resources available that have previously been downloaded. If offline, Please also see BiocManager vignette section on offline use to ensure proper functionality.

store_gene_data
Store sampled gene data for every bootstrap iteration. When the number of bootstrap reps is very high (>=100k) and/or the number of genes in hits is very high, you may want to set store_gene_data=FALSE to avoid using excessive amounts of CPU memory.

Value

A list containing three elements:

- hit.cells: vector containing the summed proportion of expression in each cell type for the target list.
- gene_data: data.table showing the number of time each gene appeared in the bootstrap sample.
- bootstrap_data: matrix in which each row represents the summed proportion of expression in each cell type for one of the random lists
- controlledCT: the controlled cell type (if applicable)

Examples

# Load the single cell data
sct_data <- ewceData::ctd()
# Set the parameters for the analysis
# Use 3 bootstrap lists for speed, for publishable analysis use >=10,000
reps <- 3
# Load gene list from Alzheimer's disease GWAS
hits <- ewceData::example_genelist()

# Bootstrap significance test, no control for transcript length or GC content
full_results <- EWCE::bootstrap_enrichment_test(
  sct_data = sct_data,
  hits = hits,
  reps = reps,
  annotLevel = 1,
  sctSpecies = "mouse",
  genelistSpecies = "human"
)
bootstrap_plot

Description
Plot bootstrap enrichment results. Support function for generate_bootstrap_plots.

Usage
bootstrap_plot(
  gene_data,
  exp_mats = NULL,
  save_dir = file.path(tempdir(), "BootstrapPlots"),
  listFileName,
  signif_ct = NULL,
  hit_thresh = 25,
  facets = "CellType",
  scales = "free_x",
  show_plot = TRUE,
  verbose = TRUE
)

Arguments
gene_data  Output from compute_gene_scores.
exp_mats  Output of generate_bootstrap_plots_exp_mats.
save_dir  Directory to save plots to.
listFileName  listFileName
signif_ct  Significant celltypes to include the plots.
facets  [Deprecated] Please use rows and cols instead.
scales  Are scales shared across all facets (the default, "fixed"), or do they vary across rows ("free_x"), columns ("free_y"), or both rows and columns ("free")?
show_plot  Print the plot.

Value
Null output.
**bootstrap_plots_for_transcriptome**

*Bootstrap plot*

**Description**

Plot results of `generate_bootstrap_plots_for_transcriptome`.

**Usage**

```r
bootstrap_plots_for_transcriptome(
  dat,
  tag,
  listFileName,
  cc,
  showGNameThresh,
  graph_theme,
  maxX,
  save_dir = file.path(tempdir(), paste0("BootstrapPlots", "_for_transcriptome")),
  height = 3.5,
  width = 3.5,
  show_plot = TRUE
)
```

**Value**

Null result.

**calculate_meanexp_for_level**

**Description**

`calculate_meanexp_for_level`

**Usage**

```r
calculate_meanexp_for_level(
  ctd_oneLevel,
  expMatrix,
  as_sparse = TRUE,
  verbose = TRUE
)
```
**calculate_specificity_for_level**

*Calculate specificity for one CTD level*

**Value**

One level of a CellTypeDataset.

**Description**

Calculate specificity for one CellTypeDataset (CTD) level.

**Usage**

```r
calculate_specificity_for_level(
  ctd_oneLevel,
  matrix_name = "mean_exp",
  as_sparse = TRUE,
  verbose = TRUE
)
```

**Arguments**

- `ctd_oneLevel` One level from a CTD.
- `matrix_name` Name of the matrix to extract.
- `as_sparse` Whether to convert exp to sparse matrix
- `verbose` Print messages.

**Value**

One CTD level.

**cell_list_dist**

*cell_list_dist*

**Description**

Specificity is generated in the main_CellTypeAnalysis_Preperation.r file

**Usage**

```r
cell_list_dist(hits, sct_data, annotLevel)
```
Arguments

<table>
<thead>
<tr>
<th>hit</th>
<th>List of gene symbols containing the target gene list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sct_data</td>
<td>List generated using generate_celltype_data.</td>
</tr>
<tr>
<td>annotLevel</td>
<td>An integer indicating which level of sct_data to analyse (Default: 1).</td>
</tr>
</tbody>
</table>

Value

The summed specificity of each celltype across a set of hits.

check_annotLevels

Description

check_annotLevels First, check the number of annotations equals the number of columns in the expression data.

Usage

check_annotLevels(annotLevels, exp)

Arguments

| exp | exp (#fix). |

Value

Null output.

check_args_for_bootstrap_plot_generation

Description

Check the input arguments of the generate_bootstrap_plots_for_transcriptome.
Usage

check_args_for_bootstrap_plot_generation(
    sct_data,
    tt,
    thresh,
    annotLevel,
    reps,
    full_results,
    listFileName,
    showGNameThresh,
    sortBy
)

Arguments

sct_data   List generated using generate_celltype_data.
tt         Differential expression table, Can be output of topTable function. Minimum requirement is that one column stores a metric of increased/decreased expression (i.e. log fold change, t-statistic for differential expression etc) and another contains gene symbols.
thresh     The number of up- and down- regulated genes to be included in each analysis (Default: 250).
annotLevel An integer indicating which level of sct_data to analyse (Default: 1).
reps       Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
full_results The full output of ewce_expression_data for the same gene list.
listFileName String used as the root for files saved using this function.
showGNameThresh Integer. If a gene has over X percent of it’s expression proportion in a cell type, then list the gene name.
sortBy     Column name of metric in tt which should be used to sort up- from down-regulated genes (Default: "t").

Value

Null output.

Description

Check the input arguments of the bootstrap_enrichment_test.
check_controlled_args

Usage

check_bootstrap_args(
  sct_data,
  hits,
  annotLevel,
  reps,
  controlledCT = NULL,
  fix_celltypes = TRUE
)

Arguments

sct_data List generated using generate_celltype_data.
hits List of gene symbols containing the target gene list. Will automatically be con-verted to human gene symbols if geneSizeControl=TRUE.
annotLevel An integer indicating which level of sct_data to analyse (Default: 1).
reps Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
controlledCT [Optional] If not NULL, and instead is the name of a cell type, then the boot-strapping controls for expression within that cell type.

Value

Null output.

check_controlled_args

Description

Check the input arguments of the controlled_geneset_enrichment.

Usage

check_controlled_args(
  bg,
  sct_data,
  annotLevel,
  disease_genes,
  hits,
  functional_genes,
  funcGenes,
  combinedGenes
)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bg</td>
<td>List of gene symbols containing the background gene list (including hit genes). If bg=NULL, an appropriate gene background will be created automatically.</td>
</tr>
<tr>
<td>sct_data</td>
<td>List generated using generate_celltype_data.</td>
</tr>
<tr>
<td>annotLevel</td>
<td>An integer indicating which level of sct_data to analyse (Default: 1).</td>
</tr>
<tr>
<td>disease_genes</td>
<td>Array of gene symbols containing the disease gene list. Does not have to be disease genes. Must be from same species as the single cell transcriptome dataset.</td>
</tr>
<tr>
<td>hits</td>
<td>Hit genes.</td>
</tr>
<tr>
<td>functional_genes</td>
<td>Array of gene symbols containing the functional gene list. The enrichment of this gene set within the disease_genes is tested. Must be from same species as the single cell transcriptome dataset.</td>
</tr>
<tr>
<td>funcGenes</td>
<td>functional_genes that are within combinedGenes.</td>
</tr>
<tr>
<td>combinedGenes</td>
<td>sct_data genes that are in the background bg.</td>
</tr>
</tbody>
</table>

Value

Null output.

Description

Check the input arguments of the ewce_expression_data.

Usage

```r
check_ewce_expression_data_args(sortBy, tt, thresh)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sortBy</td>
<td>Column name of metric in tt which should be used to sort up- from down-regulated genes (Default: &quot;t&quot;).</td>
</tr>
<tr>
<td>tt</td>
<td>Differential expression table. Can be output of topTable function. Minimum requirement is that one column stores a metric of increased/decreased expression (i.e. log fold change, t-statistic for differential expression etc) and another contains gene symbols.</td>
</tr>
<tr>
<td>thresh</td>
<td>The number of up- and down-regulated genes to be included in each analysis (Default: 250).</td>
</tr>
</tbody>
</table>

Value

Null output.
**check_ewce_genelist_inputs**

**Description**

`check_ewce_genelist_inputs` is used to check that hits and bg gene lists passed to EWCE are set up correctly. Checks they are the appropriate length. Checks all hits are in bg. Checks the species match and if not reduces to 1:1 orthologs.

**Usage**

```r
check_ewce_genelist_inputs(
  sct_data,
  hits,
  bg = NULL,
  genelistSpecies = NULL,
  sctSpecies = NULL,
  sctSpecies_origin = sctSpecies,
  output_species = "human",
  method = "homologene",
  geneSizeControl = FALSE,
  standardise_sct_data = TRUE,
  standardise_hits = FALSE,
  min_genes = 4,
  verbose = TRUE
)
```

**Arguments**

- **sct_data**
  List generated using `generate_celltype_data`.

- **hits**
  List of gene symbols containing the target gene list. Will automatically be converted to human gene symbols if `geneSizeControl=TRUE`.

- **bg**
  List of gene symbols containing the background gene list (including hit genes). If `bg=NULL`, an appropriate gene background will be created automatically.

- **genelistSpecies**
  Species that hits genes came from (no longer limited to just "mouse" and "human"). See `list_species` for all available species.

- **sctSpecies**
  Species that `sct_data` is currently formatted as (no longer limited to just "mouse" and "human"). See `list_species` for all available species.

- **sctSpecies_origin**
  Species that the `sct_data` originally came from, regardless of its current gene format (e.g. it was previously converted from mouse to human gene orthologs). This is used for computing an appropriate background.

- **output_species**
  Species to convert `sct_data` and `hits` to (Default: "human"). See `list_species` for all available species.
method

R package to use for gene mapping:
- "gprofiler": Slower but more species and genes.
- "homologene": Faster but fewer species and genes.
- "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on several different data sources.

geneSizeControl

Whether you want to control for GC content and transcript length. Recommended if the gene list originates from genetic studies (Default: FALSE). If set to TRUE, then hits must be from humans.

standardise_sct_data

Should sct_data be standardised? If TRUE:
- When sctSpecies!=output_species the sct_data will be checked for object formatting and the genes will be converted to the orthologs of the output_species with standardise_ctd (which calls map_genes internally).
- When sctSpecies==output_species, the sct_data will be checked for object formatting with standardise_ctd, but the gene names will remain untouched.

standardise_hits

Should hits be standardised? If TRUE:
- When genelistSpecies!=output_species, the genes will be converted to the orthologs of the output_species with convert_orthologs.
- When genelistSpecies==output_species, the genes will be standardised with map_genes.

If FALSE, hits will be passed on to subsequent steps as-is.

min_genes

Minimum number of genes in a gene list to test.

verbose

Print messages.

Value

A list containing
- hits: Array of MGI/HGNC gene symbols containing the target gene list.
- bg: Array of MGI/HGNC gene symbols containing the background gene list.

Examples

```r
ctd <- ewceData::ctd()
example_genelist <- ewceData::example_genelist()

called from "bootstrap_enrichment_test()" and "generate_bootstrap_plots()"

checkedLists <- EWCE::check_ewce_genelist_inputs(
sct_data = ctd,
hits = example_genelist,
sctSpecies = "mouse",
genelistSpecies = "human"
)
```

check_full_results

Description
Check full results generated by bootstrap_enrichment_test.

Usage
check_full_results(full_results, sct_data)

Arguments
- full_results: The full output of bootstrap_enrichment_test for the same gene list.
- sct_data: List generated using generate_celltype_data.

Value
Null output.

check_generate_controlled_bootstrap_geneset

describe

describe

describe

describe

describe

describe

describe

describe

describe

describe

generate_controlled_bootstrap_geneset

Description
Check input arguments to generate_controlled_bootstrap_geneset.

Usage
cHECK GENERATE CONTROLLED BOOTSTRAP GENESET

Arguments
- controlledCT: [Optional] If not NULL, and instead is the name of a cell type, then the bootstrapping controls for expression within that cell type.
- annotLevel: An integer indicating which level of sct_data to analyse (Default: 1).
- sct_data: List generated using generate_celltype_data.
- hits: List of gene symbols containing the target gene list. Will automatically be converted to human gene symbols if geneSizeControl=TRUE.
check_group_name

Value

Null output.

Description

Ensure `groupName` argument is provided to `generate_celltype_data`.

Usage

`check_group_name(groupName)`

Arguments

`groupName` A human readable name for referring to the dataset being used.

Value

Null output.

check_nas

Check NAs

Description

Check for any NAs in an expression matrix.

Usage

`check_nas(exp)`

Arguments

`exp` Expression matrix.

Value

Null output.
### check_numeric

**Check numeric**

Ensure that a matrix is numeric. If not, it will be converted to numeric.

#### Usage

```r
check_numeric(exp)
```

#### Arguments

- **exp**
  - Input matrix.

#### Value

Numeric expression matrix.

### check_percent_hits

**Get percentage of target cell type hits**

After you run `bootstrap_enrichment_test`, check what percentage of significantly enriched cell types match an expected cell type.

#### Usage

```r
check_percent_hits(
  full_results,
  target_celltype,
  mtc_method = "bonferroni",
  q_threshold = 0.05,
  verbose = TRUE
)
```

#### Arguments

- **full_results**
  - `bootstrap_enrichment_test` results.
- **target_celltype**
  - Substring to search to matching cell types (case-insensitive).
- **mtc_method**
  - Multiple-testing correction method.
- **q_threshold**
  - Corrected significance threshold.
- **verbose**
  - Print messages.
check_sce

Value

Report list.

Examples

```r
## Bootstrap significance test,
## no control for transcript length or GC content
## Use pre-computed results to speed up example
full_results <- EWCE::example_bootstrap_results()

report <- EWCE::check_percent_hits(
  full_results = full_results,
  target_celltype = "microglia"
)
```

check_sce  

Check SingleCellExperiment

Description

Check whether `exp` is a SingleCellExperiment (SCE) object and extract the relevant components.

Usage

```r
check_sce(exp, verbose = TRUE)
```

Value

List of extracted SCE components.

check_species  

Check species

Description

If species arguments are `NULL`, set default species.

Usage

```r
check_species(
  genelistSpecies = NULL,
  sctSpecies = NULL,
  sctSpecies-origin = NULL,
  sctSpecies-origin_default = "mouse",
  verbose = TRUE
)
```
compute_gene_counts

Arguments

genelistSpecies
Species that hits genes came from (no longer limited to just "mouse" and "human"). See list_species for all available species.

sctSpecies
Species that sct_data is currently formatted as (no longer limited to just "mouse" and "human"). See list_species for all available species.

sctSpecies_origin
Species that the sct_data originally came from, regardless of its current gene format (e.g. it was previously converted from mouse to human gene orthologs). This is used for computing an appropriate background.

sctSpecies_origin_default
Default value for sctSpecies_origin.

verbose
Print messages.

Value
List of corrected species names.

compute_gene_counts  Compute gene counts

Description
Counts the number of times each gene appeared in the randomly sampled gene lists.

Usage
compute_gene_counts(bootstrap_list, verbose = TRUE)

Arguments

bootstrap_list  The output of get_summed_proportions_iterate.

verbose  Print messages.

Value
data.table
compute_gene_scores  Compute gene counts

Description

Aggregate gene-level scores across all bootstrap iterations.

- boot: Mean specificity of all genes within a given cell type.
- hit: Mean specificity of a hit gene within a given cell type.

Usage

```r
compute_gene_scores(
  sct_data,
  annotLevel,
  bootstrap_list = NULL,
  hits,
  combinedGenes,
  reps = NULL,
  exp_mats = NULL,
  return_hit_exp = FALSE,
  verbose = TRUE
)
```

Arguments

- `sct_data`: List generated using `generate_celltype_data`.
- `annotLevel`: An integer indicating which level of `sct_data` to analyse (Default: 1).
- `bootstrap_list`: The output of `get_summed_proportions_iterate`.
- `hits`: list of gene names. The target gene set.
- `reps`: Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
- `return_hit_exp`: Return the expression of each hit gene.
- `verbose`: Print messages.

Value

`data.table`
Description

controlled_geneset_enrichment tests whether a functional gene set is still enriched in a disease gene set after controlling for the disease gene set’s enrichment in a particular cell type (the 'controlledCT')

Usage

```r
controlled_geneset_enrichment(
  disease_genes,
  functional_genes,
  bg = NULL,
  sct_data,
  sctSpecies = NULL,
  output_species = "human",
  disease_genes_species = NULL,
  functional_genes_species = NULL,
  method = "homologene",
  annotLevel,
  reps = 100,
  controlledCT,
  use_intersect = FALSE,
  verbose = TRUE
)
```

Arguments

disease_genes Array of gene symbols containing the disease gene list. Does not have to be disease genes. Must be from same species as the single cell transcriptome dataset.

functional_genes Array of gene symbols containing the functional gene list. The enrichment of this gene set within the disease_genes is tested. Must be from same species as the single cell transcriptome dataset.

bg List of gene symbols containing the background gene list (including hit genes). If bg=NULL, an appropriate gene background will be created automatically.

sct_data List generated using `generate_celltype_data`.

sctSpecies Species that sct_data is currently formatted as (no longer limited to just "mouse" and "human"). See list_species for all available species.

output_species Species to convert sct_data and hits to (Default: "human"). See list_species for all available species.

disease_genes_species Species of the disease_genes gene set.
controlled_geneset_enrichment

functional_genes_species
Species of the functional_genes gene set.

method
R package to use for gene mapping:
  • "gprofiler": Slower but more species and genes.
  • "homologene": Faster but fewer species and genes.
  • "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.

annotLevel
An integer indicating which level of sct_data to analyse (Default: 1).

reps
Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).

controlledCT
[Optional] If not NULL, and instead is the name of a cell type, then the bootstrapping controls for expression within that cell type.

use_intersect
When species1 and species2 are both different from output_species, this argument will determine whether to use the intersect (TRUE) or union (FALSE) of all genes from species1 and species2.

verbose
Print messages.

Value
A list containing three data frames:
  • p_controlled The probability that functional_genes are enriched in disease_genes while controlling for the level of specificity in controlledCT
  • z_controlled The z-score that functional_genes are enriched in disease_genes while controlling for the level of specificity in controlledCT
  • p_uncontrolled The probability that functional_genes are enriched in disease_genes WITHOUT controlling for the level of specificity in controlledCT
  • z_uncontrolled The z-score that functional_genes are enriched in disease_genes WITHOUT controlling for the level of specificity in controlledCT
  • reps=reps
  • controlledCT
  • actualOverlap=actual The number of genes that overlap between functional and disease gene sets

Examples
# See the vignette for more detailed explanations
# Gene set enrichment analysis controlling for cell type expression
# set seed for bootstrap reproducibility
set.seed(12345678)
## load merged dataset from vignette
cdt <- ewceData::ctd()
schiz_genes <- ewceData::schiz_genes()
hspsd_genes <- ewceData::hspsd_genes()
# Use 3 bootstrap lists for speed, for publishable analysis use >10000
reps <- 3
res_hpsd_schiz <- EWCE::controlled_geneset_enrichment(
    disease_genes = schiz_genes,
    functional_genes = hpsd_genes,
    sct_data = ctd,
    annotLevel = 1,
    reps = reps,
    controlledCT = "pyramidal CA1"
)

convert_new_ewce_to_old

convert_new_ewce_to_old

Description

convert_new_ewce_to_old Used to get an old style EWCE ctd file from a new one

Usage

convert_new_ewce_to_old(ctd, lvl)

Arguments

ctd A cell type data structure containing "mean_exp" and "specificity".
lvl The annotation level to extract.

Value

CellTypeData in the old data structure style.

convert_old_ewce_to_new

convert_old_ewce_to_new

Description

convert_old_ewce_to_new Used to get a new style EWCE ctd file (mean_exp/specificity) from old ones (all_scts).

Usage

convert_old_ewce_to_new(level1 = NA, level2 = NA, celltype_data = NA)
create_background_multilist

Create background gene list for multiple species

Description
Create background gene list for the intersection/union between multiple species (gene_list1_species, gene_list2_species, and sctSpecies), and then filter the gene lists to only include genes within the background.

Usage
create_background_multilist(
  gene_list1,
  gene_list2,
  gene_list1_species,
  gene_list2_species,
  output_species = "human",
  bg = NULL,
  use_intersect = FALSE,
  method = "homologene",
  verbose = TRUE
)

Arguments
output_species Species to convert all genes from species1 and species2 to first. Default="human", but can be to either any species supported by orthogene, including species1 or species2.
bg User supplied background list that will be returned to the user after removing duplicate genes.

Details
If you’ve already loaded it and want to pass it as a celltype_data structure, then don’t set level1 or level2.

Value
CellTypeData in the new data structure style.

Arguments
level1 File path to old level1 of EWCE ctd.
level2 File path to old level2 of EWCE ctd.
celltype_data The ctd to be converted.
**ctd_to_sce**

**Description**

Copied from scKirby, which is not yet on CRAN or Bioconductor.

**Usage**

```r
ctd_to_sce(object, as_sparse = TRUE, as_DelayedArray = FALSE, verbose = TRUE)
```

**Arguments**

- **object**: CellTypeDataset object.
- **as_sparse**: Store SingleCellExperiment matrices as sparse.
- **as_DelayedArray**: Store SingleCellExperiment matrices as DelayedArray.
- **verbose**: Print messages.

---

**create_list_network**

**Description**

Support function for prepare_genesize_control_network.

**Usage**

```r
create_list_network(data_byGene2, hits_NEW, reps = 10000, no_cores = 1)
```

**Value**

List network

---

**use_intersect**

When species1 and species2 are both different from output_species, this argument will determine whether to use the intersect (TRUE) or union (FALSE) of all genes from species1 and species2.

**method**

R package to use for gene mapping:
- "gprofiler": Slower but more species and genes.
- "homologene": Faster but fewer species and genes.
- "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on several different data sources.

**verbose**

Print messages.

**Value**

Background and gene list.
delayedarray_normalize

Value

SingleCellExperiment

Examples

ctd <- ewceData::ctd()
sce <- EWCE::ctd_to_sce(ctd)

delayedarray_normalize

Efficiently normalize a DelayedArray

Description

The following is a matrix normalization procedure that takes advantage of functions designed to be more efficient for DelayedArray objects.

Usage

delayedarray_normalize(
  exp,
  log_norm = TRUE,
  min_max = TRUE,
  plot_hists = FALSE,
  no_cores = 1
)

Arguments

exp Input matrix (e.g. gene expression).
log_norm Whether to first log-normalise exp with log1p.
min_max Whether to min/max-normalise exp.
no_cores Number of cores to parallelise across.

Value

Normalised matrix.
**drop_nonexpressed_cells**  
*Drop cells with zero gene counts*

**Description**  
Remove columns (cells) in which (gene) counts sum to zero.

**Usage**  
drop_nonexpressed_cells(exp, annotLevels, verbose = TRUE)

**Arguments**
- **exp**: Gene expression matrix.
- **annotLevels**: Cell-wise annotations to be subset if some cells are dropped.
- **verbose**: Print messages.

**Value**  
List of filtered exp and annotLevels.

**drop_nonexpressed_genes**  
*Drop genes with zero counts*

**Description**  
Remove rows (genes) in which counts sum to zero.

**Usage**  
drop_nonexpressed_genes(exp, verbose = TRUE)

**Arguments**
- **exp**: Gene expression matrix.
- **verbose**: Print messages.

**Value**  
List of filtered exp.
Description

drop_uninformative_genes drops uninformative genes in order to reduce compute time and noise in subsequent steps. It achieves this through several steps, each of which are optional:

- Drop non-1:1 orthologs:
  Removes genes that don’t have 1:1 orthologs with the output_species ("human" by default).
- Drop non-varying genes:
  Removes genes that don’t vary across cells based on variance deciles.
- Drop non-differentially expressed genes (DEGs):
  Removes genes that are not significantly differentially expressed across cell-types (multiple DEG methods available).

Usage

drop_uninformative_genes(
  exp,
  level2annot,
  mtc_method = "BH",
  adj_pval_thresh = 1e-05,
  convert_orths = FALSE,
  input_species = NULL,
  output_species = "human",
  non121_strategy = "drop_both_species",
  method = "homologene",
  as_sparse = TRUE,
  as_DelayedArray = FALSE,
  return_sce = FALSE,
  no_cores = 1,
  verbose = TRUE,
  ...
)

Arguments

exp          Expression matrix with gene names as rownames.
level2annot  Array of cell types, with each sequentially corresponding a column in the expression matrix.
mtc_method   Multiple-testing correction method used by DGE step. See p.adjust for more details.
adj_pval_thresh Minimum differential expression significance that a gene must demonstrate across level2annot (i.e. cell types).
convert_orths: If input_species! = output_species and convert_orths = TRUE, will drop genes without 1:1 output_species orthologs and then convert exp gene names to those of output_species.

input_species: Which species the gene names in exp come from. See list_species for all available species.

output_species: Which species' genes names to convert exp to. See list_species for all available species.

non121_strategy: How to handle genes that don't have 1:1 mappings between input_species:output_species. Options include:

- "drop_both_species" or "dbs" or 1:
  Drop genes that have duplicate mappings in either the input_species or output_species (DEFAULT).
- "drop_input_species" or "dis" or 2:
  Only drop genes that have duplicate mappings in the input_species.
- "drop_output_species" or "dos" or 3:
  Only drop genes that have duplicate mappings in the output_species.
- "keep_both_species" or "kbs" or 4:
  Keep all genes regardless of whether they have duplicate mappings in either species.
- "keep_popular" or "kp" or 5:
  Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.
- "sum", "mean", "median", "min" or "max": When gene_df is a matrix and gene_output = "rownames", these options will aggregate many-to-one gene mappings (input_species-to-output_species) after dropping any duplicate genes in the output_species.

method: R package to use for gene mapping:
- "gprofiler": Slower but more species and genes.
- "homologene": Faster but fewer species and genes.
- "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on several different data sources.

as_sparse: Convert exp to sparse matrix.

as_DelayedArray: Convert exp to DelayedArray for scalable processing.

return_sce: Whether to return the filtered results as an expression matrix or a SingleCellExperiment.

no_cores: Number of cores to parallelise across. Set to NULL to automatically optimise.

verbose: Print messages. # @inheritParams orthogene::convert_orthologs

...: Arguments passed on to orthogene::convert_orthologs
gene_df  Data object containing the genes (see gene_input for options on how the genes can be stored within the object). Can be one of the following formats:

- **matrix**: A sparse or dense matrix.
- **data.frame**: A data.frame, data.table or tibble.
- **codelist**: A list or character vector.

Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the ... arguments.

*Note:* If you set method="homologene", you must either supply genes in gene symbol format (e.g. "Sox2") OR set standardise_genes=TRUE.

gene_input  Which aspect of gene_df to get gene names from:

- **"rownames"**: From row names of data.frame/matrix.
- **"colnames"**: From column names of data.frame/matrix.
- **<column name>**: From a column in gene_df, e.g. "gene_names".

gene_output  How to return genes. Options include:

- **"rownames"**: As row names of gene_df.
- **"colnames"**: As column names of gene_df.
- **"columns"**: As new columns "input_gene", "ortholog_gene" (and "input_gene_standard" if standardise_genes=TRUE) in gene_df.
- **"dict"**: As a dictionary (named list) where the names are input_gene and the values are ortholog_gene.
- **"dict_rev"**: As a reversed dictionary (named list) where the names are ortholog_gene and the values are input_gene.

standardise_genes  If TRUE AND gene_output="columns", a new column "input_gene_standard" will be added to gene_df containing standardised HGNC symbols identified by gorth.

drop_nonorths  Drop genes that don’t have an ortholog in the output_species.

agg_fun  Aggregation function passed to aggregate_mapped_genes. Set to NULL to skip aggregation step (default).
mthreshold  Maximum number of ortholog names per gene to show. Passed to `gorth`. Only used when method="gprofiler" (DEFAULT : Inf).
sort_rows  Sort gene_df rows alphanumerically.
gene_map  A data.frame that maps the current gene names to new gene names. This function’s behaviour will adapt to different situations as follows:

- gene_map=<data.frame>:
  When a data.frame containing the gene key:value columns (specified by input_col and output_col, respectively) is provided, this will be used to perform aggregation/expansion.
- gene_map=NULL and input_species!=output_species:
  A gene_map is automatically generated by `map_orthologs` to perform inter-species gene aggregation/expansion.
- gene_map=NULL and input_species==output_species:
  A gene_map is automatically generated by `map_genes` to perform within-species gene symbol standardization and aggregation/expansion.

input_col  Column name within gene_map with gene names matching the row names of X.
output_col  Column name within gene_map with gene names that you wish you map the row names of X onto.

Value

exp  Expression matrix with gene names as row names.

Examples

cortex_mrna <- ewceData::cortex_mrna()
# Use only a subset of genes to keep the example quick
cortex_mrna$exp <- cortex_mrna$exp[1:300,]

## Convert orthologs at the same time
cortex_mrna$exp <- drop_uninformative_genes(
  exp = cortex_mrna$exp,
  level2annot = cortex_mrna$annot$level2class,
  input_species = "mouse"
)

dt_to_df  Convert a data.table to a data.frame.

Description

Converts a data.table to a data.frame by setting the first column as the rownames.

Usage

dt_to_df(exp)
ewce_expression_data

Value
data.frame

ewce_expression_data  Bootstrap cell type enrichment test for transcriptome data

Description

ewce_expression_data takes a differential gene expression (DGE) results table and determines the probability of cell type enrichment in the up- and down-regulated genes.

Usage

ewce_expression_data(
  sct_data,
  annotLevel = 1,
  tt,
  sortBy = "t",
  thresh = 250,
  reps = 100,
  ttSpecies = NULL,
  sctSpecies = NULL,
  output_species = NULL,
  bg = NULL,
  method = "homologene",
  verbose = TRUE,
  localHub = FALSE
)

Arguments

sct_data  List generated using generate_celltype_data.
annotLevel  An integer indicating which level of sct_data to analyse (Default: 1).
tt  Differential expression table. Can be output of topTable function. Minimum requirement is that one column stores a metric of increased/decreased expression (i.e. log fold change, t-statistic for differential expression etc) and another contains gene symbols.
sortBy  Column name of metric in tt which should be used to sort up- from down-regulated genes (Default: "t").
thresh  The number of up- and down-regulated genes to be included in each analysis (Default: 250).
reps  Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
ttSpecies  The species the differential expression table was generated from.
sctSpecies Species that sct_data is currently formatted as (no longer limited to just "mouse" and "human"). See list_species for all available species.

output_species Species to convert sct_data and hits to (Default: "human"). See list_species for all available species.

bg List of gene symbols containing the background gene list (including hit genes). If bg=NULL, an appropriate gene background will be created automatically.

method R package to use for gene mapping:
  • "gprofiler": Slower but more species and genes.
  • "homologene": Faster but fewer species and genes.
  • "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.

verbose Print messages.

localHub If working offline, add argument localHub=TRUE to work with a local, non-updated hub; It will only have resources available that have previously been downloaded. If offline, Please also see BiocManager vignette section on offline use to ensure proper functionality.

Value

A list containing five data frames:
  • results: dataframe in which each row gives the statistics (p-value, fold change and number of standard deviations from the mean) associated with the enrichment of the stated cell type in the gene list. An additional column *Direction* stores whether it the result is from the up or downregulated set.
  • hit.cells.up: vector containing the summed proportion of expression in each cell type for the target list.
  • hit.cells.down: vector containing the summed proportion of expression in each cell type for the target list.
  • bootstrap_data.up: matrix in which each row represents the summed proportion of expression in each cell type for one of the random lists.
  • bootstrap_data.down: matrix in which each row represents the summed proportion of expression in each cell type for one of the random lists.

Examples

# Load the single cell data
ctd <- ewceData::ctd()

# Set the parameters for the analysis
# Use 3 bootstrap lists for speed, for publishable analysis use >10000
reps <- 3
# Use 5 up/down regulated genes (thresh) for speed, default is 250
thresh <- 5
annotLevel <- 1 # <- Use cell level annotations (i.e. Interneurons)

# Load the top table
tt_alzh <- ewceData::tt_alzh()

tt_results <- EWCE::ewce_expression_data(
  sct_data = ctd,
  tt = tt_alzh,
  annotLevel = 1,
  thresh = thresh,
  reps = reps,
  ttSpecies = "human",
  sctSpecies = "mouse"
)

---

**ewce_plot**  
*Plot EWCE results*

**Description**

`ewce_plot` generates plots of EWCE enrichment results

**Usage**

`ewce_plot(`
  `total_res,`
  `mtc_method = "bonferroni",`
  `q_threshold = 0.05,`
  `ctd = NULL,`
  `annotLevel = 1,`
  `heights = c(0.3, 1),`
  `make_dendro = FALSE,`
  `verbose = TRUE`
`)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>total_res</strong></td>
<td>Results data.frame generated using <code>bootstrap_enrichment_test</code> or <code>ewce_expression_data</code> functions. Multiple results tables can be merged into one results table, as long as the 'list' column is set to distinguish them. Multiple testing correction is then applied across all merged results.</td>
</tr>
<tr>
<td><strong>mtc_method</strong></td>
<td>Method to be used for multiple testing correction. Argument is passed to <code>p.adjust</code> (DEFAULT: &quot;bonferroni&quot;).</td>
</tr>
<tr>
<td><strong>q_threshold</strong></td>
<td>Corrected significance threshold.</td>
</tr>
<tr>
<td><strong>ctd</strong></td>
<td>CellTypeDataset object. Should be provided so that the dendrogram can be taken from it and added to plots.</td>
</tr>
<tr>
<td><strong>annotLevel</strong></td>
<td>An integer indicating which level of ctd to analyse (Default: 1).</td>
</tr>
<tr>
<td><strong>heights</strong></td>
<td>The relative heights row in the grid. Will get repeated to match the dimensions of the grid. Passed to <code>wrap_plots</code>.</td>
</tr>
<tr>
<td><strong>make_dendro</strong></td>
<td>Add a dendrogram (requires ctd).</td>
</tr>
<tr>
<td><strong>verbose</strong></td>
<td>Print messages.</td>
</tr>
</tbody>
</table>
Value

A named list containing versions of the ggplot with and without the dendrogram. Note that cell type order on the x-axis is based on hierarchical clustering for both plots if make_dendro = TRUE.

Examples

```r
## Bootstrap significance test,
## no control for transcript length or GC content
## Use pre-computed results to speed up example
total_res <- EWCE::example_bootstrap_results()$results
plt <- ewce_plot(total_res = total_res)
```

Description

Example cell type enrichment results produced by bootstrap_enrichment_test.

Usage

```r
example_bootstrap_results(verbose = TRUE, localHub = FALSE)
```

Arguments

- `verbose` Print messages.
- `localHub` If working offline, add argument localHub=TRUE to work with a local, non-updated hub; It will only have resources available that have previously been downloaded. If offline, Please also see BiocManager vignette section on offline use to ensure proper functionality.

Value

List with 3 items.

Source

```r
# Load the single cell data
tfd <- ewceData::ctd()
# Set the parameters for the analysis
# Use 3 bootstrap lists for speed, for publishable analysis use >=10,000
reps <- 3
# Load gene list from Alzheimer's disease GWAS
td <- ewceData::example_genelist()
```
# Bootstrap significance test, no control for transcript length or GC content
full_results <- EWCE::bootstrap_enrichment_test( sct_data = ctd, hits = example_genelist, reps = reps, annotLevel = 1, sctSpecies = "mouse", genelistSpecies = "human" )
bootstrap_results <- full_results
save(bootstrap_results, file = "inst/extdata/bootstrap_results.rda")

Examples
full_results <- example_bootstrap_results()

Example bootstrap celltype enrichment test for transcriptome data

Description
Example celltype enrichment results produced by ewce_expression_data.

Usage
example_transcriptome_results( verbose = TRUE, localHub = FALSE )

Arguments
verbose
Print messages.
localHub
If working offline, add argument localHub=TRUE to work with a local, non-updated hub; It will only have resources available that have previously been downloaded. If offline, Please also see BiocManager vignette section on offline use to ensure proper functionality.

Value
List with 5 items.

Source
## Load the single cell data
cdt <- ewceData::ctd()
## Set the parameters for the analysis
## Use 3 bootstrap lists for speed, for publishable analysis use >10,000
reps <- 3
annotLevel <- 1 # <- Use cell level annotations (i.e. Interneurons)
## Use 5 up/down regulated genes (thresh) for speed, default is 250
thresh <- 5
```r
## Load the top table

tt_alzh <- ewceData::tt_alzh()

tt_results <- EWCE::ewce_expression_data(sct_data = ctd, tt = tt_alzh, annotLevel = 1, thresh = thresh, reps = reps, ttSpecies = "human", sctSpecies = "mouse")

save(tt_results, file = "inst/extdata/tt_results.rda")

Examples

tt_results <- EWCE::example_transcriptome_results()
```

---

### extract_matrix

**Extract a matrix from a CellTypeDataset**

**Description**

Extracts a particular matrix (e.g., mean_exp, specificity) from a CellTypeDataset object.

**Usage**

```r
extract_matrix(
  ctd,
  dataset,
  level = 1,
  input_species = NULL,
  output_species = "human",
  metric = "specificity",
  non121_strategy = "drop_both_species",
  method = "homologene",
  numberOfBins = 40,
  remove_unlabeled_clusters = FALSE,
  force_new_quantiles = FALSE,
  as_sparse = TRUE,
  as_DelayedArray = FALSE,
  rename_columns = TRUE,
  make_columns_unique = FALSE,
  verbose = TRUE,
  ...
)
```

**Arguments**

- **ctd** Input CellTypeData.
- **dataset** CellTypeData. name.
- **level** CTD level to extract from.
- **input_species** Which species the gene names in exp come from. See `list_species` for all available species.
output_species Which species’ genes names to convert exp to. See list_species for all available species.

metric Name of the matrix to extract.

non121_strategy How to handle genes that don’t have 1:1 mappings between input_species:output_species. Options include:

- "drop_both_species" or "dbs" or 1: Drop genes that have duplicate mappings in either the input_species or output_species (DEFAULT).
- "drop_input_species" or "dis" or 2: Only drop genes that have duplicate mappings in the input_species.
- "drop_output_species" or "dos" or 3: Only drop genes that have duplicate mappings in the output_species.
- "keep_both_species" or "kbs" or 4: Keep all genes regardless of whether they have duplicate mappings in either species.
- "keep_popular" or "kp" or 5: Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.
- "sum","mean","median","min" or "max": When gene_df is a matrix and gene_output="rownames", these options will aggregate many-to-one gene mappings(input_species-to-output_species) after dropping any duplicate genes in the output_species.

method R package to use for gene mapping:

- "gprofiler": Slower but more species and genes.
- "homologene": Faster but fewer species and genes.
- "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.

numberOfBins Number of non-zero quantile bins.

remove_unlabeled_clusters Remove any samples that have numeric column names.

force_new_quantiles By default, quantile computation is skipped if they have already been computed. Set =FALSE to override this and generate new quantiles.

as_sparse Convert to sparse matrix.

as_DelayedArray Convert to DelayedArray.

rename_columns Remove replace_chars from column names.

make_columns_unique Rename each columns with the prefix dataset.species.celltype.

verbose Print messages. Set verbose=2 if you want to print all messages from internal functions as well.
Arguments passed on to `orthogene::convert_orthologs`

gene_df  Data object containing the genes (see `gene_input` for options on how the genes can be stored within the object).
Can be one of the following formats:

- **matrix**: A sparse or dense matrix.
- **data.frame**: A data.frame, data.table, or tibble.
- **codelist**: A list or character vector.

Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the ... arguments.

**Note**: If you set `method="homologene"`, you must either supply genes in gene symbol format (e.g. "Sox2") OR set `standardise_genes=TRUE`.

gene_input  Which aspect of gene_df to get gene names from:

- "rownames": From row names of data.frame/matrix.
- "colnames": From column names of data.frame/matrix.
- `<column name>`: From a column in gene_df, e.g. "gene_names".

gene_output  How to return genes. Options include:

- "rownames": As row names of gene_df.
- "colnames": As column names of gene_df.
- "columns": As new columns "input_gene", "ortholog_gene" (and "input_gene_standard" if `standardise_genes=TRUE`) in gene_df.
- "dict": As a dictionary (named list) where the names are input_gene and the values are ortholog_gene.
- "dict_rev": As a reversed dictionary (named list) where the names are ortholog_gene and the values are input_gene.

**standardise_genes** If TRUE AND gene_output="columns", a new column "input_gene_standard" will be added to gene_df containing standardised HGNC symbols identified by `gorth`.

drop_nonorths  Drop genes that don’t have an ortholog in the output_species.

**agg_fun** Aggregation function passed to `aggregate_mapped_genes`. Set to `NULL` to skip aggregation step (default).
filter_ctd_genes

mthreshold  Maximum number of ortholog names per gene to show. Passed to gorth. Only used when method="gprofiler" (DEFAULT : Inf).
sort_rows  Sort gene_df rows alphanumerically.
gene_map  A data.frame that maps the current gene names to new gene names. This function’s behaviour will adapt to different situations as follows:
   • gene_map=<data.frame>:
     When a data.frame containing the gene key:value columns (specified by input_col and output_col, respectively) is provided, this will be used to perform aggregation/expansion.
   • gene_map=NULL and input_species!=output_species:
     A gene_map is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion.
   • gene_map=NULL and input_species==output_species:
     A gene_map is automatically generated by map_genes to perform within-species gene symbol standardization and aggregation/expansion.

input_col  Column name within gene_map with gene names matching the row names of X.
output_col  Column name within gene_map with gene names that you wish you map the row names of X onto.

Value

(specificity) matrix.

filter_ctd_genes  Filter genes in a CellTypeDataset

Description

Removes rows from each matrix within a CellTypeDataset (CTD) that are not within gene_subset.

Usage

filter_ctd_genes(ctd, gene_subset)

Arguments

ctd  CellTypeDataset.
gene_subset  Genes to subset to.

Value

Filtered CellTypeDataset.
Examples

c <- ewceData::ctd()
c <- standardise_ctd(c, input_species="mouse")
gene_subset <- rownames(c[[1]]$mean.exp)[1:100]
c_subset <- EWCE::filter_ctd_genes(c = c, gene_subset = gene_subset)

filter_genes_without_1to1_homolog

Description

Deprecated function. Please use filter_nonorthologs instead.

Usage

filter_genes_without_1to1_homolog(
  filenames,
  input_species = "mouse",
  convert_nonhuman_genes = TRUE,
  annot_levels = NULL,
  suffix = "_orthologs",
  verbose = TRUE
)

Arguments

filenames List of file names for sct_data saved as .rda files.
input_species Which species the gene names in exp come from.
convert_nonhuman_genes Whether to convert the exp row names to human gene names.
annot_levels [Optional] Names of each annotation level.
suffix Suffix to add to the file name (right before .rda).
verbose Print messages.

Details

Note: This function replaces the original filter_genes_without_1to1_homolog function. filter_genes_without_1to1_homolog is now a wrapper for filter_nonorthologs.

Value

List of the filtered CellTypeData file names.
Examples

# Load the single cell data
cwd <- ewceData::ctd()
tmp <- tempfile()
save(ctd, file = tmp)
fNames_ALLCELLS_orths <- EWCE::filter_nonorthologs(filenames = tmp)

Description

filter_nonorthologs Takes the filenames of CellTypeData files, loads them, drops any genes which don’t have a 1:1 orthologs with humans, and then convert the gene to human orthologs. The new files are then saved to disk, appending '_orthologs' to the file name.

Usage

filter_nonorthologs(
  filenames,
  input_species = NULL,
  convert_nonhuman_genes = TRUE,
  annot_levels = NULL,
  suffix = "_orthologs",
  method = "homologene",
  non121_strategy = "drop_both_species",
  verbose = TRUE,
  ...
)

Arguments

filenames List of file names for sct_data saved as .rda files.
input_species Which species the gene names in exp come from.
convert_nonhuman_genes Whether to convert the exp row names to human gene names.
annot_levels [Optional] Names of each annotation level.
suffix Suffix to add to the file name (right before .rda).
method R package to use for gene mapping:
  • "gprofiler" : Slower but more species and genes.
  • "homologene" : Faster but fewer species and genes.
  • "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
non121_strategy

How to handle genes that don’t have 1:1 mappings between input_species:output_species. Options include:

- "drop_both_species" or "dbs" or 1:
  Drop genes that have duplicate mappings in either the input_species or output_species (DEFAULT).
- "drop_input_species" or "dis" or 2:
  Only drop genes that have duplicate mappings in the input_species.
- "drop_output_species" or "dos" or 3:
  Only drop genes that have duplicate mappings in the output_species.
- "keep_both_species" or "kbs" or 4:
  Keep all genes regardless of whether they have duplicate mappings in either species.
- "keep_popular" or "kp" or 5:
  Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.
- "sum", "mean", "median", "min" or "max":
  When gene_df is a matrix and gene_output="rownames", these options will aggregate many-to-one gene mappings (input_species-to-output_species) after dropping any duplicate genes in the output_species.

verbose

Print messages.

... Arguments passed on to orthogene::convert_orthologs

gene_df Data object containing the genes (see gene_input for options on how the genes can be stored within the object). Can be one of the following formats:

- matrix:
  A sparse or dense matrix.
- data.frame:
  A data.frame, data.table or tibble.
- codelist:
  A list or character vector.

Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the ... arguments.

Note: If you set method="homologene", you must either supply genes in gene symbol format (e.g. "Sox2") OR set standardise_genes=TRUE.

gene_input Which aspect of gene_df to get gene names from:

- "rownames":
  From row names of data.frame/matrix.
gene_output  How to return genes. Options include:

- "rownames":
  As row names of gene_df.
- "colnames":
  As column names of gene_df.
- "columns":
  As new columns "input_gene", "ortholog_gene" (and "input_gene_standard" if standardise_genes=TRUE) in gene_df.
- "dict":
  As a dictionary (named list) where the names are input_gene and the values are ortholog_gene.
- "dict_rev":
  As a reversed dictionary (named list) where the names are ortholog_gene and the values are input_gene.

standardise_genes  If TRUE AND gene_output="columns", a new column "input_gene_standard" will be added to gene_df containing standardised HGNC symbols identified by gorth.

output_species  Name of the output species (e.g. "human","chicken"). Use map_species to return a full list of available species.

drop_nonorths  Drop genes that don’t have an ortholog in the output_species.

agg_fun  Aggregation function passed to aggregate_mapped_genes. Set to NULL to skip aggregation step (default).

mthreshold  Maximum number of ortholog names per gene to show. Passed to gorth. Only used when method="gprofiler" (DEFAULT : Inf).

as_sparse  Convert gene_df to a sparse matrix. Only works if gene_df is one of the following classes:

- matrix
- Matrix
- data.frame
- data.table
- tibble

If gene_df is a sparse matrix to begin with, it will be returned as a sparse matrix (so long as gene_output= "rownames" or "colnames").

as_DelayedArray  Convert aggregated matrix to DelayedArray.

sort_rows  Sort gene_df rows alphanumerically.

gene_map  A data.frame that maps the current gene names to new gene names.

This function’s behaviour will adapt to different situations as follows:

- gene_map=<\text{data.frame}>:
  When a data.frame containing the gene key:value columns (specified
by input_col and output_col, respectively) is provided, this will be used to perform aggregation/expansion.

- gene_map=NULL and input_species!=output_species:
  A gene_map is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion.
- gene_map=NULL and input_species==output_species:
  A gene_map is automatically generated by map_genes to perform within-species gene gene symbol standardization and aggregation/expansion.

input_col Column name within gene_map with gene names matching the row names of X.
output_col Column name within gene_map with gene names that you wish you map the row names of X onto.

Details

Note: This function replaces the original filter_genes_without_1to1_homolog function. filter_genes_without_1to1_homolog is now a wrapper for filter_nonorthologs.

Value

List of the filtered CellTypeData file names.

Examples

```r
# Load the single cell data
ctd <- ewceData::ctd()
tmp <- tempfile()
save(ctd, file = tmp)
fNames_ALLCELLS_orths <- EWCE::filter_nonorthologs(filenames = tmp)
```

### filter_variance_quantiles

*Filter variance quantiles*

**Description**

Remove rows in exp that do not vary substantially across rows.

**Usage**

```r
filter_variance_quantiles(
  exp,
  log10_norm = TRUE,
  n_quantiles = 10,
  min_variance_quantile = as.integer(n_quantiles/2),
  verbose = TRUE
)
```
**fix_bad_hgnc_symbols**

### Arguments

- **exp**
  - Gene expression matrix.
- **log10_norm**
  - Log10-normalise exp before computing variance.
- **n_quantiles**
  - Number of quantile bins to use. Defaults to deciles (n_quantiles=10).
- **min_variance_quantile**
  - The minimum variance quantile to keep values from.
- **verbose**
  - Print messages.

### Value

Filtered exp.

### Description

Given an expression matrix, wherein the rows are supposed to be HGNC symbols, find those symbols which are not official HGNC symbols, then correct them if possible. Return the expression matrix with corrected symbols.

### Usage

```r
fix_bad_hgnc_symbols(
  exp,
  dropNonHGNC = FALSE,
  as_sparse = TRUE,
  verbose = TRUE,
  localHub = FALSE
)
```

### Arguments

- **exp**
  - An expression matrix where the rows are HGNC symbols or a SingleCellExperiment (SCE) or other Ranged Summarized Experiment (SE) type object.
- **dropNonHGNC**
  - Boolean. Should symbols not recognised as HGNC symbols be dropped?
- **as_sparse**
  - Convert exp to sparse matrix.
- **verbose**
  - Print messages.
- **localHub**
  - If working offline, add argument localHub=TRUE to work with a local, non-updated hub; It will only have resources available that have previously been downloaded. If offline, Please also see BiocManager vignette section on offline use to ensure proper functionality.
fix_bad_mgi_symbols

Value

Returns the expression matrix with the rownames corrected and rows representing the same gene merged. If a SingleCellExperiment (SCE) or other Ranged Summarized Experiment (SE) type object was inputted this will be returned with the corrected expression matrix under counts.

Examples

# create example expression matrix, could be part of a exp, annot list obj
exp <- matrix(data = runif(70), ncol = 10)
# Add HGNC gene names but add with an error:
# MARCH8 is a HGNC symbol which if opened in excel will convert to Mar-08
rownames(exp) <-
c("MT-TF", "MT-RNR1", "MT-TV", "MT-RNR2", "MT-TL1", "MT-ND1", "Mar-08")
exp <- fix_bad_hgnc_symbols(exp)
# fix_bad_hgnc_symbols warns the user of this possible issue

fix_bad_mgi_symbols

fix_bad_mgi_symbols - Given an expression matrix, wherein the rows are supposed to be MGI symbols, find those symbols which are not official MGI symbols, then check in the MGI synonym database for whether they match to a proper MGI symbol. Where a symbol is found to be an aliases for a gene that is already in the dataset, the combined reads are summed together.

Description

Also checks whether any gene names contain "Sep", "Mar" or "Feb". These should be checked for any suggestion that excel has corrupted the gene names.

Usage

fix_bad_mgi_symbols(
  exp,
  mrk_file_path = NULL,
  printAllBadSymbols = FALSE,
  as_sparse = TRUE,
  verbose = TRUE,
  localHub = FALSE
)

Arguments

exp An expression matrix where the rows are MGI symbols, or a SingleCellExperiment (SCE) or other Ranged Summarized Experiment (SE) type object.

mrk_file_path Path to the MRK_List2 file which can be downloaded from www.informatics.jax.org/downloads/reports/index.html

printAllBadSymbols Output to console all the bad gene symbols
fix_celltype_names

Description

Make sure celltypes don’t contain characters that could interfere with downstream analyses. For example, the R package MAGMA.Celltyping cannot have spaces in celltype names because spaces are used as a delimiter in later steps.

Usage

```r
fix_celltype_names(
  celltypes,
  replace_chars = "[-]\.[ ]\[\//\][\\/\\]",
  make_unique = TRUE
)
```

Arguments

- `celltypes`: Character vector of celltype names.
- `replace_chars`: Regex string of characters to replace with "_" when renaming columns.
- `make_unique`: Make all entries unique.

Value

Fixed celltype names.
Examples

ct <- c("microglia", "astrocytes", "Pyramidal SS")
ct_fixed <- fix_celltype_names(celltypes = ct)

fix_celltype_names_full_results

Fix celltype name in full results

Description

Aligns celltype names in full results generated by bootstrap_enrichment_test with the standardised CellTypeDataset (CTD) produced by standardise_ctd.

Usage

fix_celltype_names_full_results(full_results, verbose = TRUE)

Arguments

full_results Cell-type enrichment results generated by bootstrap_enrichment_test.
verbose Print messages.

Value

Fixed full results.

generate_bootstrap_plots

Generate bootstrap plots

Description

generate_bootstrap_plots takes a gene list and a single cell type transcriptome dataset and generates plots which show how the expression of the genes in the list compares to those in randomly generated gene lists.
Usage

generate_bootstrap_plots(
  sct_data = NULL,
  hits = NULL,
  bg = NULL,
  genelistSpecies = NULL,
  sctSpecies = NULL,
  output_species = "human",
  method = "homologene",
  reps = 100,
  annotLevel = 1,
  geneSizeControl = FALSE,
  full_results = NULL,
  listFileName = paste0("_level", annotLevel),
  adj_pval_thresh = 0.05,
  facets = "CellType",
  scales = "free_x",
  save_plot = TRUE,
  verbose = TRUE
)

Arguments

- **sct_data** List generated using `generate_celltype_data`.
- **hits** List of gene symbols containing the target gene list. Will automatically be converted to human gene symbols if `geneSizeControl=TRUE`.
- **bg** List of gene symbols containing the background gene list (including hit genes). If `bg=NULL`, an appropriate gene background will be created automatically.
- **genelistSpecies** Species that `hits` genes came from (no longer limited to just "mouse" and "human"). See `list_species` for all available species.
- **sctSpecies** Species that `sct_data` is currently formatted as (no longer limited to just "mouse" and "human"). See `list_species` for all available species.
- **output_species** Species to convert `sct_data` and `hits` to (Default: "human"). See `list_species` for all available species.
- **method** R package to use for gene mapping:
  - "gprofiler": Slower but more species and genes.
  - "homologene": Faster but fewer species and genes.
  - "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
- **reps** Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
- **annotLevel** An integer indicating which level of `sct_data` to analyse (Default: 1).
geneSizeControl  
Whether you want to control for GC content and transcript length. Recommended if the gene list originates from genetic studies (Default: FALSE). If set to TRUE, then hits must be from humans.

full_results  
The full output of bootstrap_enrichment_test for the same gene list.

listFileName  
String used as the root for files saved using this function.

adj_pval_thresh  
Adjusted p-value threshold of celltypes to include in plots.

facets  
[Deprecated] Please use rows and cols instead.

scales  
Are scales shared across all facets (the default, "fixed"), or do they vary across rows ("free_x"), columns ("free_y"), or both rows and columns ("free")?

save_dir  
Directory where the BootstrapPlots folder should be saved, default is a temp directory.

show_plot  
Print the plot.

verbose  
Print messages.

Value

Saves a set of pdf files containing graphs and returns the file where they are saved. These will be saved with the file name adjusted using the value of listFileName. The files are saved into the 'BootstrapPlot' folder. Files start with one of the following:

- **qqplot_noText**: sorts the gene list according to how enriched it is in the relevant cell type. Plots the value in the target list against the mean value in the bootstrapped lists.
- **qqplot_wtGSym**: as above but labels the gene symbols for the highest expressed genes.
- **bootDists**: rather than just showing the mean of the bootstrapped lists, a boxplot shows the distribution of values.
- **bootDists_LOG**: shows the bootstrapped distributions with the y-axis shown on a log scale.

Examples

```r
## Load the single cell data
sct_data <- ewceData::ctd()

## Set the parameters for the analysis
## Use 5 bootstrap lists for speed, for publishable analysis use >10000
reps <- 5

## Load the gene list and get human orthologs
hits <- ewceData::example_genelist()

## Bootstrap significance test,
## no control for transcript length or GC content
## Use pre-computed results to speed up example
full_results <- EWCE::example_bootstrap_results()

### Skip this for example purposes
# full_results <- EWCE::bootstrap_enrichment_test(
```
generate_bootstrap_plots_for_transcriptome

Generate bootstrap plots

Description

Takes a gene list and a single cell type transcriptome dataset and generates plots which show how the expression of the genes in the list compares to those in randomly generated gene lists.

Usage

generate_bootstrap_plots_for_transcriptome(
  sct_data, 
  tt, 
  bg = NULL, 
  thresh = 250, 
  annotLevel = 1, 
  reps = 100, 
  full_results = NA, 
  listFileName = "", 
  showGNameThresh = 25, 
  ttSpecies = NULL, 
  sctSpecies = NULL, 
  output_species = NULL, 
  sortBy = "t", 
  sig_only = TRUE, 
  sig_col = "q", 
  sig_thresh = 0.05, 
  celltype_col = "CellType", 
  plot_types = c("bootstrap", "bootstrap_distributions", "log_bootstrap_distributions"),
generate_bootstrap_plots_for_transcriptome

save_dir = file.path(tempdir(), "BootstrapPlots"),
method = "homologene",
verbose = TRUE
)

Arguments

sct_data List generated using generate_celltype_data.
tt Differential expression table. Can be output of topTable function. Minimum requirement is that one column stores a metric of increased/decreased expression (i.e. log fold change, t-statistic for differential expression etc) and another contains gene symbols.
bg List of gene symbols containing the background gene list (including hit genes). If bg=NULL, an appropriate gene background will be created automatically.
thresh The number of up- and down- regulated genes to be included in each analysis (Default: 250).
annotLevel An integer indicating which level of sct_data to analyse (Default: 1).
reps Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
full_results The full output of ewce_expression_data for the same gene list.
listFileName String used as the root for files saved using this function.
showGNameThresh Integer. If a gene has over X percent of it’s expression proportion in a cell type, then list the gene name.
ttSpecies The species the differential expression table was generated from.
sctSpecies Species that sct_data is currently formatted as (no longer limited to just "mouse" and "human"). See list_species for all available species.
output_species Species to convert sct_data and hits to (Default: "human"). See list_species for all available species.
sortBy Column name of metric in tt which should be used to sort up- from down-regulated genes (Default: "t").
sig_only Should plots only be generated for cells which have significant changes?
sig_col Column name in tt that contains the significance values.
sig_thresh Threshold by which to filter tt by sig.col.
celltype_col Column within tt that contains celltype names.
plot_types Plot types to generate.
save_dir Directory where the BootstrapPlots folder should be saved, default is a temp directory.
method R package to use for gene mapping:
  • "gprofiler" : Slower but more species and genes.
  • "homologene" : Faster but fewer species and genes.
  • "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
verbose Print messages.
**generate_bootstrap_plots_for_transcriptome**

**Value**

Saves a set of PDF files containing graphs. Then returns a nested list with each plot and the path where it was saved to. Files start with one of the following:

- **qqplot_noText**: sorts the gene list according to how enriched it is in the relevant cell type. Plots the value in the target list against the mean value in the bootstrapped lists.
- **qqplot_wtGSym**: as above but labels the gene symbols for the highest expressed genes.
- **bootDists**: rather than just showing the mean of the bootstrapped lists, a boxplot shows the distribution of values
- **bootDists_LOG**: shows the bootstrapped distributions with the y-axis shown on a log scale

**Examples**

```r
## Load the single cell data
ctd <- ewceData::ctd()

## Set the parameters for the analysis
## Use 3 bootstrap lists for speed, for publishable analysis use >10,000
reps <- 3
annotLevel <- 1 # <- Use cell level annotations (i.e. Interneurons)
## Use 5 up/down regulated genes (thresh) for speed, default is 250
thresh <- 5

## Load the top table
tt_alzh <- ewceData::tt_alzh()

## See ?example_transcriptome_results for full code to produce tt_results
tt_results <- EWCE::example_transcriptome_results()

## Bootstrap significance test,
## no control for transcript length or GC content
savePath <- EWCE::generate_bootstrap_plots_for_transcriptome(
  sct_data = ctd,
  tt = tt_alzh,
  thresh = thresh,
  annotLevel = 1,
  full_results = tt_results,
  listFileName = "examples",
  reps = reps,
  ttSpecies = "human",
  sctSpecies = "mouse",
  # Only do one plot type for demo purposes
  plot_types = "bootstrap"
)
```
generate_celltype_data

Generate CellTypeData (CTD) file

Description

generate_celltype_data takes gene expression data and cell type annotations and creates CellTypeData (CTD) files which contain matrices of mean expression and specificity per cell type.

Usage

generate_celltype_data(
  exp,
  annotLevels,
  groupName,
  no_cores = 1,
  savePath = tempdir(),
  file_prefix = "ctd",
  as_sparse = TRUE,
  as_DelayedArray = FALSE,
  normSpec = FALSE,
  convert_orths = FALSE,
  input_species = "mouse",
  output_species = "human",
  non121_strategy = "drop_both_species",
  method = "homologene",
  force_new_file = TRUE,
  specificity_quantiles = TRUE,
  numberOfBins = 40,
  dendrograms = TRUE,
  return_ctd = FALSE,
  verbose = TRUE,
  ...
)

Arguments

exp Numerical matrix with row for each gene and column for each cell. Row names are gene symbols. Column names are cell IDs which can be cross referenced against the annot data frame.

annotLevels List with arrays of strings containing the cell type names associated with each column in exp.

groupName A human readable name for referring to the dataset being used.

no_cores Number of cores that should be used to speedup the computation. NOTE: Use no_cores=1 when using this package in windows system.

savePath Directory where the CTD file should be saved.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_prefix</td>
<td>Prefix to add to saved CTD file name.</td>
</tr>
<tr>
<td>as_sparse</td>
<td>Convert exp to a sparse Matrix.</td>
</tr>
<tr>
<td>as_DelayedArray</td>
<td>Convert exp toDelayedArray.</td>
</tr>
<tr>
<td>normSpec</td>
<td>Boolean indicating whether specificity data should be transformed to a normal distribution by cell type, giving equivalent scores across all cell types.</td>
</tr>
<tr>
<td>convert_orths</td>
<td>If input_species!=output_species and convert_orths=TRUE, will drop genes without 1:1 output_species orthologs and then convert exp gene names to those of output_species.</td>
</tr>
<tr>
<td>input_species</td>
<td>The species that the exp dataset comes from. See list_species for all available species.</td>
</tr>
<tr>
<td>output_species</td>
<td>Species to convert exp to (Default: &quot;human&quot;). See list_species for all available species.</td>
</tr>
<tr>
<td>non121_strategy</td>
<td>How to handle genes that don’t have 1:1 mappings between input_species:output_species. Options include:</td>
</tr>
<tr>
<td></td>
<td>• &quot;drop_both_species&quot; or &quot;dbs&quot; or 1: Drop genes that have duplicate mappings in either the input_species or output_species (DEFAULT).</td>
</tr>
<tr>
<td></td>
<td>• &quot;drop_input_species&quot; or &quot;dis&quot; or 2: Only drop genes that have duplicate mappings in the input_species.</td>
</tr>
<tr>
<td></td>
<td>• &quot;drop_output_species&quot; or &quot;dos&quot; or 3: Only drop genes that have duplicate mappings in the output_species.</td>
</tr>
<tr>
<td></td>
<td>• &quot;keep_both_species&quot; or &quot;kbs&quot; or 4: Keep all genes regardless of whether they have duplicate mappings in either species.</td>
</tr>
<tr>
<td></td>
<td>• &quot;keep_popular&quot; or &quot;kp&quot; or 5: Return only the most &quot;popular&quot; interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.</td>
</tr>
<tr>
<td></td>
<td>• &quot;sum&quot;,&quot;mean&quot;,&quot;median&quot;,&quot;min&quot; or &quot;max&quot;: When gene_df is a matrix and gene_output=&quot;rownames&quot;, these options will aggregate many-to-one gene mappings (input_species-to-output_species) after dropping any duplicate genes in the output_species.</td>
</tr>
<tr>
<td>method</td>
<td>R package to use for gene mapping:</td>
</tr>
<tr>
<td></td>
<td>• &quot;gprofiler&quot; : Slower but more species and genes.</td>
</tr>
<tr>
<td></td>
<td>• &quot;homologene&quot; : Faster but fewer species and genes.</td>
</tr>
<tr>
<td></td>
<td>• &quot;babelgene&quot; : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</td>
</tr>
<tr>
<td>force_new_file</td>
<td>If a file of the same name as the one being created already exists, overwrite it.</td>
</tr>
<tr>
<td>specificity_quantiles</td>
<td>Compute specificity quantiles. Recommended to set to TRUE.</td>
</tr>
</tbody>
</table>
**generate_celltype_data**

- **numberOfBins**: Number of quantile 'bins' to use (40 is recommended).
- **dendrograms**: Add dendrogram plots.
- **return_ctd**: Return the CTD object in a list along with the file name, instead of just the file name.
- **verbose**: Print messages.

Arguments passed on to `orthogene::convert_orthologs`

**gene_df**: Data object containing the genes (see `gene_input` for options on how the genes can be stored within the object). Can be one of the following formats:

- **matrix**: A sparse or dense matrix.
- **data.frame**: A `data.frame`, `data.table`, or `tibble`.
- **codelist**: A list or character vector.

Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the ... arguments.

*Note*: If you set `method="homologene"`, you must either supply genes in gene symbol format (e.g. "Sox2") OR set `standardise_genes=TRUE`.

**gene_input**: Which aspect of `gene_df` to get gene names from:

- **"rownames"**: From row names of `data.frame/matrix`.
- **"colnames"**: From column names of `data.frame/matrix`.
- **<column name>**: From a column in `gene_df`, e.g. "gene_names".

**gene_output**: How to return genes. Options include:

- **"rownames"**: As row names of `gene_df`.
- **"colnames"**: As column names of `gene_df`.
- **"columns"**: As new columns "input_gene", "ortholog_gene" (and "input_gene_standard" if `standardise_genes=TRUE`) in `gene_df`.
- **"dict"**: As a dictionary (named list) where the names are input_gene and the values are ortholog_gene.
- **"dict_rev"**: As a reversed dictionary (named list) where the names are ortholog_gene and the values are input_gene.
```r
# Load the single cell data
cortex_mrna <- ewceData::cortex_mrna()
# Use only a subset to keep the example quick
expData <- cortex_mrna$exp[1:100,]
l1 <- cortex_mrna$annot$level1class
l2 <- cortex_mrna$annot$level2class
annotLevels <- list(l1 = l1, l2 = l2)

# Generate celltype data
fNames_ALLCELLS <- EWCE::generate_celltype_data(
  exp = expData,
  annotLevels = annotLevels,
  groupName = "allKImouse"
)
```
Description

Used to generate cell type-controlled bootstrapped gene sets.

Usage

```r
generate_controlled_bootstrap_geneset(
  hits,
  sct_data,
  annotLevel,
  reps,
  controlledCT = FALSE,
  verbose = TRUE
)
```

Arguments

- **hits**: List of gene symbols containing the target gene list. Will automatically be converted to human gene symbols if geneSizeControl=TRUE.
- **sct_data**: List generated using `generate_celltype_data`.
- **annotLevel**: An integer indicating which level of `sct_data` to analyse (Default: 1).
- **reps**: Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
- **controlledCT**: [Optional] If not NULL, and instead is the name of a cell type, then the bootstrapping controls for expression within that cell type.
- **verbose**: Print messages.

Details

See `controlled_genefset_enrichment` for examples.

Value

Matrix of genes (such that `nrows=length(hits)` and `ncols=reps`), where each column is a gene list.
get_celltype_table

Description

get_celltype_table Generates a table that can be used for supplementary tables of publications. The table lists how many cells are associated with each cell type, the level of annotation, and the dataset from which it was generated.

Usage

get_celltype_table(annot)

Arguments

annot An annotation dataframe, which columns named 'level1class', 'level2class' and 'dataset_name'

Value

A dataframe with columns 'name', 'level', 'freq' and 'dataset_name'

Examples

# See PrepLDSC.Rmd for origin of merged_ALLCELLS$annot
cortex_mrna <- ewceData::cortex_mrna()
cortex_mrna$annot$dataset_name <- "cortex_mrna"
celltype_table <- EWCE::get_celltype_table(cortex_mrna$annot)

get_ctd_levels

Get the names of CellTypeDataset levels

Description

Returns the level names of a CellTypeDataset. If none are available, will instead return a vector of numbers (one number per level).

Usage

get_ctd_levels(ctd, max_only = FALSE)

Arguments

ctd CellTypeDataset.
max_only Only return the level with the greatest depth (e.g. "level3" in c("level1", "level2", "level3")).
get_exp_data_for_bootstrapped_genes

Description

Support function for `generate_bootstrap_plots_for_transcriptome`.

ctd

CellTypeDataset. If set to `NULL` (default), will simply return all possible matrix names.

matrices

Matrix names to search for.

description

List of matrix names.

get_ctd_matrix_names

Get CTD matrix names

Description

Find the names of all data matrices in a CellTypeDataset.

Usage

```r
get_ctd_matrix_names(
  ctd = NULL,
  matrices = c("mean_exp", "median_exp", "specificity", "median_specificity",
                "specificity_quantiles"),
  verbose = TRUE
)
```

Arguments

description

- `ctd`: CellTypeDataset. If set to `NULL` (default), will simply return all possible matrix names.
- `matrices`: Matrix names to search for.
- `verbose`: Print messages.

Value

List of matrix names.
Usage

get_exp_data_for_bootstrapped_genes(
    results,
    signif_res,
    sct_data,
    hits,
    combinedGenes,
    annotLevel,
    nReps = 100,
    as_sparse = TRUE,
    verbose = TRUE
)

Arguments

signif_res  signif_res (#fix).
sct_data    List generated using generate_celltype_data.
hits        Gene hits.
combinedGenes Combined list of genes from sct_data, hits, and background bg.
annotLevel  An integer indicating which level of sct_data to analyse (Default: 1).
verbose     Print messages.
full_results full_results (#fix).

Value

exp_mats

get_sig_results

Extract significant results

Description

Extract significant results from output of bootstrap_enrichment_test.

Usage

get_sig_results(
    full_results,
    mtc_method = "BH",
    q_threshold = 0.05,
    verbose = TRUE
)
get_summed_proportions

Arguments

- full_results: Output of `bootstrap_enrichment_test`.
- mtc_method: Multiple-testing correction method (passed to `p.adjust`).
- q_threshold: Maximum multiple-testing-corrected p-value to include.
- verbose: Print messages.

Value

Filtered enrichment results table.

get_summed_proportions

*Get summed proportions*

Description

get_summed_proportions Given the target gene set, randomly sample gene lists of equal length, obtain the specificity of these and then obtain the mean specificity in each sampled list (and the target list).

Usage

```r
get_summed_proportions(
  hits,
  sct_data,
  annotLevel,
  reps,
  no_cores = 1,
  geneSizeControl,
  controlledCT = NULL,
  control_network = NULL,
  store_gene_data = TRUE,
  verbose = TRUE
)
```

Arguments

- hits: list of gene names. The target gene set.
- sct_data: List generated using `generate_celltype_data`.
- annotLevel: An integer indicating which level of sct_data to analyse (Default: 1).
- reps: Number of random gene lists to generate (Default: 100, but should be >=10,000 for publication-quality results).
- no_cores: Number of cores to parallelise bootstrapping reps over.
geneSizeControl
Whether you want to control for GC content and transcript length. Recommended if the gene list originates from genetic studies (Default: FALSE). If set to TRUE, then hits must be from humans.

controlledCT
[Optional] If not NULL, and instead is the name of a cell type, then the bootstrapping controls for expression within that cell type.

control_network
If geneSizeControl=TRUE, then must provide the control network.

store_gene_data
Store sampled gene data for every bootstrap iteration. When the number of bootstrap reps is very high (>=100k) and/or the number of genes in hits is very high, you may want to set store_gene_data=FALSE to avoid using excessive amounts of CPU memory.

verbose
Print messages.

Details
See bootstrap_enrichment_test for examples.

Value
A list containing three elements:

- hit.cells: vector containing the summed proportion of expression in each cell type for the target list.
- gene_data: data.table showing the number of time each gene appeared in the bootstrap sample.
- bootstrap_data: matrix in which each row represents the summed proportion of expression in each cell type for one of the random lists
- controlledCT: the controlled cell type (if applicable)

is_32bit
Checks whether OS is a 32-bit Windows

Description
Helper function to avoid duplicate test runs on Windows OS.

Usage
is_32bit()
is_celltypedataset  
*Check whether object is a CellTypeDataset*

**Description**
Check whether an object is a CellTypeDataset.

**Usage**

```r
is_celltypedataset(ctd)
```

**Arguments**

- **ctd**  
  Object.

**Value**

boolean

---

is_ctd_standardised  
*Check whether a CellTypeDataset is standardised*

**Description**
Check whether a CellTypeDataset was previously standardised using `standardise_ctd`.

**Usage**

```r
is_ctd_standardised(ctd)
```

**Arguments**

- **ctd**  
  CellTypeDataset.

**Value**

Whether the ctd is standardised.
**is_delayed_array**

Assess whether an object is a DelayedArray.

---

**Description**

Assess whether an object is a DelayedArray or one of its derived object types.

**Usage**

\[
\text{is\_delayed\_array}(X)
\]

**Arguments**

- \( X \) Object.

**Value**

boolean

---

**is_matrix**

Assess whether an object is a Matrix.

---

**Description**

Assess whether an object is a Matrix or one of its derived object types.

**Usage**

\[
\text{is\_matrix}(X)
\]

**Arguments**

- \( X \) Object.

**Value**

boolean
is_sparse_matrix Assess whether an object is a sparse matrix

Description
Assess whether an object is a sparse matrix or one of its derived object types.

Usage
is_sparse_matrix(X)

Arguments
X Object.

Value
boolean

list_species List all species

Description
List all species that EWCE can convert genes from/to. Wrapper function for map_species.

Usage
list_species(VERBOSE = TRUE)

Arguments
VERBOSE Print messages.

Value
List of species EWCE can input/output genes as.

Examples
list_species()
**load_rdata**

Description

Load processed data (.rda format) using a function that assigns it to a specific variable (so you don’t have to guess what the loaded variable name is).

Usage

```r
load_rdata(fileName)
```

Arguments

- `fileName` Name of the file to load.

Value

Data object.

Examples

```r
tmp <- tempfile()
save(mtcars, file = tmp)
mtcars2 <- load_rdata(tmp)
```

**max_ctd_depth**

Get max CTD depth

Description

Get the maximum level depth from a list of CellTypeDataset objects.

Usage

```r
max_ctd_depth(CTD_list)
```

Arguments

- `CTD_list` A list of CellTypeDataset objects.

Value

integer
merged_ewce

Multiple EWCE results from multiple studies

Description

merged_ewce combines enrichment results from multiple studies targeting the same scientific problem.

Usage

merged_ewce(results, reps = 100)

Arguments

- `results`: a list of EWCE results generated using `add_res_to_merging_list`.
- `reps`: Number of random gene lists to generate (Default=100 but should be >=10,000 for publication-quality results).

Value

dataframe in which each row gives the statistics (p-value, fold change and number of standard deviations from the mean) associated with the enrichment of the stated cell type in the gene list.

Examples

```R
# Load the single cell data
c td <- ewceData::ctd()

# Use 3 bootstrap lists for speed, for publishable analysis use >10000
reps <- 3
# Use 5 up/down regulated genes (thresh) for speed, default is 250
thresh <- 5

# Load the data
tt_alzh_BA36 <- ewceData::tt_alzh_BA36()
tt_alzh_BA44 <- ewceData::tt_alzh_BA44()

# Run EWCE analysis
tt_results_36 <- EWCE::ewce_expression_data(
  sct_data = ctd,
  tt = tt_alzh_BA36,
  thresh = thresh,
  annotLevel = 1,
  reps = reps,
  ttSpecies = "human",
  sctSpecies = "mouse"
)
tt_results_44 <- EWCE::ewce_expression_data(
  sct_data = ctd,
```
merge_crd

Merge multiple CellTypeDataset references

Description

Import CellTypeDataset (CTD) references from a remote repository, standardize each, and then merge into one CTD. Optionally, can return these as a merged SingleCellExperiment.

Usage

merge_crd(
    CTD_list,
    save_dir = tempdir(),
    standardise_CTD = FALSE,
    as_SCE = FALSE,
    gene_union = TRUE,
    merge_levels = seq(1, 5),
    save_split_SCE = FALSE,
    save_split_CTD = FALSE,
    save_merged_SCE = TRUE,
    force_new_quantiles = FALSE,
    number_of_bins = 40,
    as_sparse = TRUE,
    as_DelayedArray = FALSE,
    verbose = TRUE,
    ...
)
Arguments

CTD_list (Named) list of CellTypeDatasets.
save_dir The directory to save merged files in.
standardise_CTD Whether to run standardise_ctd.

as_SCE If TRUE (default), returns the merged results as a named list of SingleCellExperiments. If FALSE, returns as a CTD object.
gene_union Whether to take the gene union or intersection when merging matrices (mean_exp, specificity, etc.).
merge_levels Which CTD levels you want to merge. Can be a single value (e.g. merge_levels=5) or a list (e.g. merge_levels=c(1:5)). If some CTD don’t have the same number of levels, the maximum level depth available in that CTD will be used instead.

save_split_SCE Whether to save individual SCE files in the subdirectory standardized_CTD_SCE.
save_split_CTD Whether to save individual CTD files in the subdirectory standardized_CTD.
save_merged_SCE Save the final merged SCE object, or simply to return it.
force_new_quantiles If specificity quantiles matrix already exists, create a new one.

numberOfBins Number of bins to compute specificity quantiles with.
as_sparse Convert matrices to sparse matrix.
as_DelayedArray Convert matrices to DelayedArray.
verbose Print messages.
...

Additional arguments to be passed to standardise_cfd.

Value

List of CellTypeDatasets or SingleCellExperiments.

Examples

## Let's pretend these are different CTD datasets
c1d <- ewceData::ctd()
c2d <- ctd1
CTD_list <- list(c1d, c2d)
CTD_merged <- EWCE::merge_cfd(CTD_list = CTD_list)
**merge_sce**

*Merge multiple SingleCellExperiment objects*

**Description**

Merge several SingleCellExperiment (SCE) objects from different batches/experiments. Extracted from the scMerge package.

**Usage**

```r
merge_sce(
  sce_list,
  method = "intersect",
  cut_off_batch = 0.01,
  cut_off_overall = 0.01,
  use_assays = NULL,
  colData_names = NULL,
  batch_names = NULL,
  verbose = TRUE
)
```

**Arguments**

- **sce_list**
  A list contains the SingleCellExperiment Object from each batch.
- **method**
  A string indicates the method of combining the gene expression matrix, either union or intersect. Default to intersect. union only supports matrix class.
- **cut_off_batch**
  A numeric vector indicating the cut-off for the proportion of a gene is expressed within each batch.
- **cut_off_overall**
  A numeric vector indicating the cut-off for the proportion of a gene is expressed overall data.
- **use_assays**
  A string vector indicating the expression matrices to be combined. The first assay named will be used to determine the proportion of zeros.
- **colData_names**
  A string vector indicating the colData that are combined.
- **batch_names**
  A string vector indicating the batch names for the output SCE object.
- **verbose**
  Print messages.

**Value**

A SingleCellExperiment object with the list of SCE objects combined.

**Author(s)**

Yingxin Lin (modified by Brian Schilder)
merge_sce_list

Merge of list of SingleCellExperiment objects

Description

Merge of list of CellTypeDatasets stored as SingleCellExperiment objects into one SingleCellExperiment object.

Usage

merge_sce_list(
SCE_lists = NULL,
parent_folder = NULL,
pattern = ".rds$",
merge_levels = seq(1, 5),
gene_union = TRUE,
as_sparse = TRUE,
as_DelayedArray = TRUE,
verbose = TRUE
)

Arguments

SCE_lists A list of SingleCellExperiment objects.
parent_folder Can supply the path to a folder instead of SCE_lists. Any SingleCellExperiment objects matching pattern will be imported.
merge_levels CellTypeDataset levels to merge.

Value

SingleCellExperiment
merge_two_expfiles

Merge two exp files

Description

merge_two_expfiles Used to combine two single cell type datasets.

Usage

merge_two_expfiles(
  exp1,
  exp2,
  annot1,
  annot2,
  name1 = "",
  name2 = "",
  as_sparse = TRUE,
  as_DelayedArray = FALSE,
  verbose = TRUE
)

Arguments

exp1 Numerical expression matrix for dataset1 with row for each gene and column for each cell. Row names are gene symbols. Column names are cell IDs which can be cross referenced against the annot data frame.

exp2 Numerical expression matrix for dataset2 with row for each gene and column for each cell. Row names are gene symbols. Column names are cell IDs which can be cross referenced against the annot data frame.

annot1 Annotation data frame for dataset1 which contains three columns at least: cell_id, level1class and level2class

annot2 Annotation data frame for dataset2 which contains three columns at least: cell_id, level1class and level2class

name1 Name used to refer to dataset 1. Leave blank if it’s already a merged dataset.

name2 Name used to refer to dataset 2. Leave blank if it’s already a merged dataset.

as_sparse Convert the merged exp to a sparse matrix.

as_DELAYED_ARRAY Convert the merged exp to a DelayedArray.

verbose Print messages.

Value

List containing merged exp and annot.
Examples

cortex_mrna <- ewceData::cortex_mrna()
exp1 <- cortex_mrna$exp[, 1:50]
exp2 <- cortex_mrna$exp[, 51:100]
annot1 <- cortex_mrna$annot[1:50, ]
annot2 <- cortex_mrna$annot[51:100, ]
merged_res <- EWCE::merge_two_expfiles(
  exp1 = exp1,
  exp2 = exp2,
  annot1 = annot1,
  annot2 = annot2,
  name1 = "dataset1",
  name2 = "dataset2"
)

messager

Print messages

Description

Print messages with option to silence.

Usage

messager(..., v = TRUE)

Arguments

... Message input.

v Whether to print messages.

Value

Null output.

message_parallel

Print messages

Description

Print messages even from within parallelised functions.

Usage

message_parallel(...)
Arguments

... Message input.

Value

Null output.

Description

Adjusts `ggplot` label display. See `comma` for details. Support function for `plot_log_bootstrap_distributions`.

Usage

`myScalesComma(x)`

Value

Numeric vector

---

`plot_ctd`  

*Plot* CellTypeData *metrics*

Description

Plot CellTypeData metrics such as mean_exp, specificity and/or specificity_quantiles.

Usage

`plot_ctd(ctd, genes, level = 1, metric = "specificity", show_plot = TRUE)`

Arguments

- `ctd` CellTypeDataset.
- `genes` Which genes in ctd to plot.
- `level` Annotation level in ctd to plot.
- `metric` Which metric in the ctd to plot:
  - "mean_exp"
  - "specificity"
  - "specificity_quantiles"
- `show_plot` Whether to print the plot or simply return it.
Value

ggplot object.

Examples

c <- ewceData::ctd()
plt <- EWCE::plot_ctd(c, genes = c("Apoe", "Gfap", "Gapdh"))

Description

Plot log bootstrap distributions

Usage

plot_log_bootstrap_distributions(
  dat,
  exp_mats,
  cc,
  hit_exp,
  tag,
  listFileName,
  graph_theme,
  save_dir = file.path(tempdir(), paste0("BootstrapPlots", ".for_transcriptome")),
  height = 3.5,
  width = 3.5
)

Value

Null result.

Description

Plot results of \texttt{generate_bootstrap_plots_for_transcriptome}.

plot_with_bootstrap_distributions

\textit{Plot with bootstrap distributions}

Description

Plot results of \texttt{generate_bootstrap_plots_for_transcriptome}. 
Usage

plot_with_bootstrap_distributions(
    exp_mats,
    cc,
    hit_exp,
    tag,
    listFileName,
    graph_theme,
    save_dir = file.path(tempdir(), paste0("BootstrapPlots", "_for_transcriptome")),
    height = 3.5,
    width = 3.5
)

Value

Null result.

Description

prep_dendro adds a dendrogram to a CellTypeDataset (CTD).

Usage

prep.dendro(ctdIN)

Arguments

ctdIN A single annotLevel of a ctd, i.e. ctd[1]) (the function is intended to be used via apply).

Value

A CellTypeDataset with dendrogram plotting info added.
prepare_genesize_control_network

Prepare genesize control network

Description

prepare_genesize_control_network takes a gene list and finds semi-randomly selected gene lists which are matched for gene length and GC content.

Usage

prepare_genesize_control_network(
    hits,
    bg = NULL,
    reps = 10000,
    no_cores = 1,
    sctSpecies = NULL,
    genelistSpecies = NULL,
    verbose = TRUE,
    localHub = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hits</td>
<td>List of gene symbols containing the target gene list. Will automatically be converted to human gene symbols if geneSizeControl=TRUE.</td>
</tr>
<tr>
<td>bg</td>
<td>List of gene symbols containing the background gene list (including hit genes). If bg=NULL, an appropriate gene background will be created automatically.</td>
</tr>
<tr>
<td>reps</td>
<td>Number of gene lists to sample.</td>
</tr>
<tr>
<td>no_cores</td>
<td>Number of cores to parallelise bootstrapping reps over.</td>
</tr>
<tr>
<td>sctSpecies</td>
<td>Species that sct_data is currently formatted as (no longer limited to just &quot;mouse&quot; and &quot;human&quot;). See list_species for all available species.</td>
</tr>
<tr>
<td>genelistSpecies</td>
<td>Species that hits genes came from (no longer limited to just &quot;mouse&quot; and &quot;human&quot;). See list_species for all available species.</td>
</tr>
<tr>
<td>verbose</td>
<td>Print messages.</td>
</tr>
<tr>
<td>localHub</td>
<td>If working offline, add argument localHub=TRUE to work with a local, non-updated hub; It will only have resources available that have previously been downloaded. If offline, Please also see BiocManager vignette section on offline use to ensure proper functionality.</td>
</tr>
</tbody>
</table>

Value

A list containing three data frames:
prepare_tt

- hits: Array of HGNC symbols containing the hit genes. May be slightly reduced if gene length / GC content could not be found for all genes.
- list_network: The control gene lists as a data frame of HGNC symbols

**Description**

Prepare differential gene expression table for `generate_bootstrap_plots_for_transcriptome` or `ewce_expression_data`.

**Usage**

```r
prepare_tt(
  tt,
  tt_genecol = NULL,
  ttSpecies,
  output_species,
  method = "homologene",
  verbose = TRUE
)
```

**Arguments**

- **tt**
  Differential expression table. Can be output of `topTable` function. Minimum requirement is that one column stores a metric of increased/decreased expression (i.e. log fold change, t-statistic for differential expression etc) and another contains gene symbols.

- **ttSpecies**
  The species the differential expression table was generated from.

- **output_species**
  Species to convert `sct_data` and hits to (Default: "human"). See `list_species` for all available species.

- **method**
  R package to use for gene mapping:
  - "gprofiler": Slower but more species and genes.
  - "homologene": Faster but fewer species and genes.
  - "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on several different data sources.

- **verbose**
  Print messages.

**Value**

List of 3 items
prep_dendro  

Prepare dendrogram

Description

prep_dendro adds a dendrogram to a CellTypeDataset (CTD).

Usage

prep_dendro(ctdIN, expand = c(0, 0.66))

Arguments

ctdIN  
A single annotLevel of a ctd, i.e. ctd[[1]] (the function is intended to be used via apply).

Value

A CellTypeDataset with dendrogram plotting info added.

report_dge  

Report DGE

Description

Report differential gene expression (DGE) results

Usage

report_dge(exp, keep_genes, adj_pval_thresh = 0.05, verbose = TRUE)

Arguments

exp  
Gene expression matrix.

keep_genes  
Genes kept after DGE.

adj_pval_thresh  
Minimum differential expression significance that a gene must demonstrate across level2annot (i.e. cell types).

verbose  
Print messages. # @inheritParams orthogene::convert_orthologs

Value

Null output.
**report_results**  
*Report cell type enrichment results*

**Description**

Report cell type enrichment results generated by `bootstrap_enrichment_test`.

**Usage**

```r
report_results(results, sig_thresh = 0.05, verbose = TRUE)
```

**Value**

NULL output.

---

**run_deseq2**  
*Run DGE: DESeq2*

**Description**

Run Differential Gene Expression with **DESeq2**.

**Usage**

```r
run_deseq2(exp, level2annot, test = "LRT", no_cores = 1, verbose = TRUE, ...)
```

**Arguments**

- **exp**: Expression matrix with gene names as rownames.
- **level2annot**: Array of cell types, with each sequentially corresponding a column in the expression matrix.
- **test**: either "Wald" or "LRT", which will then use either Wald significance tests (defined by `nbinomWaldTest`), or the likelihood ratio test on the difference in deviance between a full and reduced model formula (defined by `nbinomLRT`)
- **no_cores**: Number of cores to parallelise across. Set to NULL to automatically optimise.
- **verbose**: Print messages. #`@inheritParams orthogene::convert_orthologs
- **...**: Additional arguments to be passed to `gorth` or `homologene`.

**NOTE**: To return only the most "popular" interspecies ortholog mappings, supply `mthreshold=1` here AND set `method="gprofiler"` above. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.

For more details, please see [here](#).
run_limma

Description

Run Differential Gene Expression with `limma`.

Usage

```r
run_limma(exp, level2annot, mtc_method = "BH", verbose = TRUE, ...)
```

Arguments

- `exp` Expression matrix with gene names as rownames.
- `level2annot` Array of cell types, with each sequentially corresponding a column in the expression matrix.
- `mtc_method` Multiple-testing correction method used by DGE step. See `p.adjust` for more details.
- `verbose` Print messages. See `@inheritParams orthogene::convert_orthologs`
- `...` Additional arguments to be passed to `gorth` or `homologene`.

NOTE: To return only the most "popular" interspecies ortholog mappings, supply `mthreshold=1` here AND set `method="gprofiler"` above. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.

For more details, please see `here`.

Value

`limma` results.
**run_mast**  \hspace{1cm}  **Run DGE: MAST**

**Description**

Run Differential Gene Expression with MAST.

**Usage**

```r
test = "LRT", mtc_method = "BH", no_cores = 1, ...
```

**Arguments**

- `exp` : Expression matrix with gene names as rownames.
- `level2annot` : Array of cell types, with each sequentially corresponding a column in the expression matrix.
- `mtc_method` : Multiple-testing correction method used by DGE step. See `p.adjust` for more details.
- `no_cores` : Number of cores to parallelise DGE across.
- `...` : Additional arguments to be passed to `gorth` or `homologene`.

**Value**

MAST results

**Source**

MAST tutorial
Description

Support function for EWCE::merge_sce_list.

Usage

\[
\text{sce_lists_apply}
(\text{SCE_lists}, \\
\text{return_genes} = \text{FALSE}, \\
\text{level} = 2, \\
\text{as_matrix} = \text{FALSE}, \\
\text{as_DelayedArray} = \text{FALSE})
\]

Value

List of SingleCellExperiments.

Description

Merge a list of SingleCellExperiments.

Usage

\[
\text{sce_merged_apply}(\text{SCE_merged}, \text{as_sparse} = \text{TRUE}, \text{as_DelayedArray} = \text{FALSE})
\]

Value

Merged SingleCellExperiment.
### sct_normalize

*Normalize expression matrix*

**Description**

Normalize expression matrix by accounting for library size. Uses `sctransform`.

**Usage**

```r
sct_normalize(exp, as_sparse = TRUE, verbose = TRUE)
```

**Arguments**

- `exp`: Gene x cell expression matrix.
- `as_sparse`: Convert `exp` to sparse matrix.
- `verbose`: Print messages.

**Value**

Normalised expression matrix.

**Examples**

```r
cortex_mrna <- ewceData::cortex_mrna()
exp_sct_normed <- EWCE::sct_normalize(exp = cortex_mrna$exp[1:300, ])
```

### standardise_ctd

*Convert a CellTypeDataset into standardized format*

**Description**

This function will take a CTD, drop all genes without 1:1 orthologs with the `output_species` ("human" by default), convert the remaining genes to gene symbols, assign names to each level, and convert all matrices to sparse matrices and/or `DelayedArray`.

**Usage**

```r
standardise_ctd(
  ctd,
  dataset,
  input_species = NULL,
  output_species = "human",
  sctSpecies_origin = input_species,
  non121_strategy = "drop_both_species",
  method = "homologene",
```

```
force_new_quantiles = TRUE,
force_standardise = FALSE,
remove_unlabeled_clusters = FALSE,
numberOfBins = 40,
keep_anot = TRUE,
keep_plots = TRUE,
as_sparse = TRUE,
as_DelayedArray = FALSE,
rename_columns = TRUE,
make_columns_unique = FALSE,
verbose = TRUE,
...)

Arguments

ctd Input CellTypeData.
dataset CellTypeData. name.
input_species Which species the gene names in exp come from. See list_species for all available species.
output_species Which species’ genes names to convert exp to. See list_species for all available species.
sctSpecies_origin Species that the sct_data originally came from, regardless of its current gene format (e.g. it was previously converted from mouse to human gene orthologs). This is used for computing an appropriate background.

non121_strategy How to handle genes that don’t have 1:1 mappings between input_species:output_species. Options include:

- "drop_both_species" or "dbs" or 1:
  Drop genes that have duplicate mappings in either the input_species or output_species (DEFAULT).
- "drop_input_species" or "dis" or 2:
  Only drop genes that have duplicate mappings in the input_species.
- "drop_output_species" or "dos" or 3:
  Only drop genes that have duplicate mappings in the output_species.
- "keep_both_species" or "kbs" or 4:
  Keep all genes regardless of whether they have duplicate mappings in either species.
- "keep_popular" or "kp" or 5:
  Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.
- "sum","mean","median","min" or "max":
  When gene_df is a matrix and gene_output="rownames", these options
will aggregate many-to-one gene mappings (input_species-to-output_species) after dropping any duplicate genes in the output_species.

**method**  
R package to use for gene mapping:  
- "gprofiler": Slower but more species and genes.  
- "homologene": Faster but fewer species and genes.  
- "babelgene": Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on several different data sources.

**force_new_quantiles**  
By default, quantile computation is skipped if they have already been computed. Set =TRUE to override this and generate new quantiles.

**force_standardise**  
If ctd has already been standardised, whether to rerun standardisation anyway (Default: FALSE).

**remove_unlabeled_clusters**  
Remove any samples that have numeric column names.

**numberOfBins**  
Number of non-zero quantile bins.

**keep_annot**  
Keep the column annotation data if provided.

**keep_plots**  
Keep the dendrograms if provided.

**as_sparse**  
Convert to sparse matrix.

**as_DelayedArray**  
Convert to DelayedArray.

**rename_columns**  
Remove replace_chars from column names.

**make_columns_unique**  
Rename each columns with the prefix dataset.species.celltype.

**verbose**  
Print messages. Set verbose=2 if you want to print all messages from internal functions as well.

**...**  
Arguments passed on to orthogene::convert_orthologs

**gene_df**  
Data object containing the genes (see gene_input for options on how the genes can be stored within the object). Can be one of the following formats:

- **matrix**: A sparse or dense matrix.  
- **data.frame**: A data.frame, data.table or tibble.  
- **codelist**: A list or character vector.

Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the ... arguments.

*Note*: If you set method="homologene", you must either supply genes in gene symbol format (e.g. "Sox2") OR set standardise_genes=TRUE.
gene_input  Which aspect of gene_df to get gene names from:

- "rownames":
  From row names of data.frame/matrix.
- "colnames":
  From column names of data.frame/matrix.
- <column name>:
  From a column in gene_df, e.g. "gene_names".

gene_output  How to return genes. Options include:

- "rownames":
  As row names of gene_df.
- "colnames":
  As column names of gene_df.
- "columns":
  As new columns "input_gene", "ortholog_gene" (and "input_gene_standard" if standardise_genes=TRUE) in gene_df.
- "dict":
  As a dictionary (named list) where the names are input_gene and the values are ortholog_gene.
- "dict_rev":
  As a reversed dictionary (named list) where the names are ortholog_gene and the values are input_gene.

standardise_genes  If TRUE AND gene_output=“columns”, a new column "input_gene_standard" will be added to gene_df containing standardised HGNC symbols identified by gorth.

drop_nonorths  Drop genes that don’t have an ortholog in the output_species.

agg_fun  Aggregation function passed to aggregate_mapped_genes. Set to NULL to skip aggregation step (default).

mthreshold  Maximum number of ortholog names per gene to show. Passed to gorth. Only used when method="gprofiler" (DEFAULT : Inf).

sort_rows  Sort gene_df rows alphanumerically.

gene_map  A data.frame that maps the current gene names to new gene names. This function’s behaviour will adapt to different situations as follows:

- gene_map=<data.frame>:
  When a data.frame containing the gene key: value columns (specified by input_col and output_col, respectively) is provided, this will be used to perform aggregation/expansion.
- gene_map=NULL and input_species!=output_species:
  A gene_map is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion.
- gene_map=NULL and input_species==output_species:
  A gene_map is automatically generated by map_genes to perform within-species gene symbol standardization and aggregation/expansion.

input_col  Column name within gene_map with gene names matching the row names of X.
output_col Column name within gene_map with gene names that you wish you map the row names of X onto.

Value

Standardised CellTypeDataset.

Examples

ctd <- ewceData::ctd()
ctd_std <- EWCE::standardise_ctd(
  ctd = ctd,
  input_species = "mouse",
  dataset = "Zeisel2016"
)

theme_graph

Get graph theme

Description

Get graph theme for plots created by generate_bootstrap_plots_for_transcriptome.

Usage

theme_graph()

Value

ggplot2 graph theme.

to_dataframe

Convert object to data.frame

Description

Convert a variety of object types to data.frame format.

Usage

to_dataframe(X, verbose = TRUE)

Arguments

X Object.
verbose Print messages.
**to_delayed_array**  
*Convert object to DelayedArray*

**Description**  
Convert a variety of object types to `DelayedArray` format.

**Usage**  
```r  
to_delayed_array(exp, as_DelayedArray = TRUE, verbose = TRUE)  
```

**Arguments**  
- `exp` Object.
- `as_DelayedArray` Whether to convert `exp` to `DelayedArray`.
- `verbose` Print messages.

**Value**  
`DelayedArray`.

---

**to_sparse_matrix**  
*Convert object to sparse matrix*

**Description**  
Convert a variety of object types to sparse matrix format.

**Usage**  
```r  
to_sparse_matrix(exp, as_sparse = TRUE, verbose = TRUE)  
```

**Arguments**  
- `exp` Object.
- `as_sparse` Whether to convert `exp` to sparse matrix
- `verbose` Print messages.

**Value**  
Sparse matrix.
Index

* internal
  assign_cores, 6
  bootstrap_plot, 11
  bootstrap_plots_for_transcriptome, 12
  calculate_meanexp_for_level, 12
  calculate_specificity_for_level, 13
  cell_list_dist, 13
  check_annotLevels, 14
  check_args_for_bootstrap_plot_generation, 14
  check_bootstrap_args, 15
  check_controlled_args, 16
  check_ewce_expression_data_args, 17
  check_full_results, 20
  check_generate_controlled_bootstrap_geneset, 20
  check_group_name, 21
  check_nas, 21
  check_numeric, 22
  check_sce, 23
  check_species, 23
  compute_gene_counts, 24
  compute_gene_scores, 25
  convert_new_ewce_to_old, 28
  convert_old_ewce_to_new, 28
  create_background_multilist, 29
  create_list_network, 30
  delayedarray_normalize, 31
  drop_nonexpressed_cells, 32
  drop_nonexpressed_genes, 32
  dt_to_df, 36
  extract_matrix, 42
  filter_variance_quantiles, 50
  fix_celltype_names_full_results, 54
  generate_controlled_bootstrap_geneset, 64
  get_ctd_levels, 65
  get_ctd_matrix_names, 66
  get_exp_data_for_bootstrapped_genes, 66
  get_sig_results, 67
  get_summed_proportions, 68
  is_32bit, 69
  is_celltypedataset, 70
  is_ctd_standardised, 70
  max_ctd_depth, 73
  merge_sce_list, 78
  message_parallel, 80
  messager, 80
  myScalesComma, 81
  plot_log_bootstrap_distributions, 82
  plot_with_bootstrap_distributions, 82
  prep_dendro, 86
  prepare_genesize_control_network, 84
  prepare_tt, 85
  report_dge, 86
  report_results, 87
  run_deseq2, 87
  run_limma, 88
  run_mast, 89
  sce_lists_apply, 90
  sce_merged_apply, 90
  theme_graph, 95
  to_dataframe, 95
  to_sparse_matrix, 96
  add_res_to_merging_list, 5, 74
  aggregate_mapped_genes, 35, 44, 49, 63, 94
  apply, 6
  assign_cores, 6
INDEX

bin_columns_into_quantiles, 6
bin_specificity_into_quantiles, 7
bootstrap_enrichment_test, 5, 8, 15, 20, 22, 39, 40, 54, 56, 67–69, 87
bootstrap_plot, 11
bootstrap_plots_for_transcriptome, 12
calculate_meanexp_for_level, 12
calculate_specificity_for_level, 13
cell_list_dist, 13
check_annotLevels, 14
check_args_for_bootstrap_plot_generation, 14
check_bootstrap_args, 15
check_controlled_args, 16
check_ewce_expression_data_args, 17
check_ewce_genelist_inputs, 18
check_full_results, 20
check_generate_controlled_bootstrap_geneset, 20
check_group_name, 21
check_nas, 21
check_numeric, 22
check_percent_hits, 22
check_sce, 23
check_species, 23
comma, 81
compute_gene_counts, 24
compute_gene_scores, 11, 25
controlled_geneset_enrichment, 16, 26, 64
convert_new_ewce_to_old, 28
convert_old_ewce_to_new, 28
convert_orthologs, 10, 19
create_background_multilist, 29
create_list_network, 30
ctd_to_sce, 30
cut, 7
data.frame, 36, 45, 49, 63, 94, 96
data.table, 24, 25
DelayedArray, 49, 96
delayedarray_normalize, 31
drop_nonexpressed_cells, 32
drop_nonexpressed_genes, 32
drop_uninformative_genes, 33
dt_to_df, 36

EWCE (EWCE-package), 4
EWCE-package, 4
ewce_expression_data, 5, 15, 17, 37, 39, 41, 58, 85
ewce_plot, 39
example_bootstrap_results, 40
example_transcriptome_results, 41
extract_matrix, 42
filter_ctd_genes, 45
filter_genes_without_1to1_homolog, 46
filter_nonorthologs, 46, 47
filter_variance_quantiles, 50
fix_bad hgnc_symbols, 51
fix_bad mgi_symbols, 52
fix_celltype_names, 53
fix_celltype_names_full_results, 54
generate_bootstrap_plots, 11, 54
generate_bootstrap_plots_for_transcriptome, 12, 14, 57, 66, 82, 85, 95
generate_celltype_data, 9, 14–18, 20, 21, 25, 26, 37, 55, 58, 60, 64, 67, 68
generate_controlled_bootstrap_geneset, 20, 64
generate_celltype_table, 65
get_crd_levels, 65
generate_celltype_matrix_names, 66
get_exp_data_for_bootstrapped_genes, 66
generate_celltype_table, 65
get_exp_data_for_bootstrapped_genes, 66
get_sig_results, 67
get_summed_proportions, 68
ggplot, 40
import, 35, 36, 44, 45, 49, 63, 87–89, 94
homologene, 87–89
is_32bit, 69
is_celltypedataset, 70
is_crd_standardised, 70
is_delayed_array, 71
is_matrix, 71
is_sparse_matrix, 72
list_species, 9, 18, 24, 26, 34, 38, 42, 43, 55, 58, 61, 72, 84, 85, 92
load_rdata, 73
log1p, 31
map_genes, 9, 10, 19, 36, 45, 50, 63, 94
map_orthologs, 36, 45, 50, 63, 94
map_species, 49, 72
max_ctd_depth, 73
merge_cfd, 75
merge_sce, 77
merge_sce_list, 78
merge_two_experiments, 79
merged_ewce, 74
message_parallel, 80
messager, 80
myScalesComma, 81
nbinomLRT, 87
nbinomWaldTest, 87
orthogene::convert_orthologs, 34, 44, 48, 62, 93
p.adjust, 9, 33, 39, 68, 88, 89
plot_cfd, 81
plot_log_bootstrap_distributions, 81, 82
plot_with_bootstrap_distributions, 82
prep.dendro, 83
prep_dendro, 86
prepare_genesize_control_network, 84
prepare_tt, 85
report_dge, 86
report_results, 87
run_deseq2, 87
run_limma, 88
run_mast, 89
sce_lists_apply, 90
sce_merged_apply, 90
sct_normalize, 91
SingleCellExperiment, 75, 76, 78
standardise_cfd, 9, 19, 54, 70, 91
theme_graph, 95
to_dataframe, 95
to_delayed_array, 96
to_sparse_matrix, 96
topTable, 15, 17, 37, 58, 85
wrap_plots, 39