Package 'EWCE'

December 5, 2024

Type Package

Title Expression Weighted Celltype Enrichment

Version 1.14.0

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, rmarkdown, 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 7.3.1

Encoding UTF-8

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/EWCE

git_branch RELEASE_3_20

git_last_commit a4535e3

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-12-05

Author Alan Murphy [cre] (<https://orcid.org/0000-0002-2487-8753>),
Brian Schilder [aut] (<https://orcid.org/0000-0001-5949-2191>),
Nathan Skene [aut] (<https://orcid.org/0000-0002-6807-3180>)

Maintainer Alan Murphy <alanmurph94@hotmail.com>

Contents

`EWCE-package` *EWCE: Expression Weighted Celltype Enrichment*

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)

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
ctd <- 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 = ctd, 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)
```

assign_cores *Assign cores*

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

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

`bin_specificity_into_quantiles` *bin_specificity_into_quantiles*

Description

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

Usage

```
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 backgrund.

`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).

`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`
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.

`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 *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

```
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` *calculate_meanexp_for_level*

Description

calculate_meanexp_for_level

Usage

```
calculate_meanexp_for_level(
  ctd_oneLevel,
  expMatrix,
  as_sparse = TRUE,
  verbose = TRUE
)
```

Value

One level of a CellTypeDataset.

`calculate_specificity_for_level` *Calculate specificity for one CTD level*

Description

Calculate specificity for one CellTypeDataset (CTD) level.

Usage

```
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

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

Arguments

`hits`	List of gene symbols containing the target gene list.
`sct_data`	List generated using generate_celltype_data.
`annotLevel`	An integer indicating which level of `sct_data` to analyse (*Default: 1*).

Value

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

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

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
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.

check_bootstrap_args *check_bootstrap_args*

Description

Check the input arguments of the bootstrap_enrichment_test.

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 converted 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 bootstrapping controls for expression within that cell type.

Value

Null output.

check_controlled_args *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

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.
annotLevel	An integer indicating which level of sct_data to analyse (*Default: 1*).
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.
hits	Hit genes.
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.
funcGenes	functional_genes that are within combinedGenes.
combinedGenes	sct_data genes that are in the background bg.

Value

Null output.

---

check_ewce_expression_data_args

check_ewce_expression_data_args

Description

Check the input arguments of the ewce_expression_data.

Usage

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

Arguments

sortBy	Column name of metric in tt which should be used to sort up- from down-regulated genes (Default: "t").
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).

Value

Null output.

check_ewce_genelist_inputs
check_ewce_genelist_inputs

Description

check_ewce_genelist_inputs Is used to check that hits and bg gene lists passed to EWCE are setup 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

```
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 backgrund.
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.

`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

```
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 *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 *generate_controlled_bootstrap_geneset*

Description

Check input arguments to generate_controlled_bootstrap_geneset.

Usage

```
check_generate_controlled_bootstrap_geneset(
  controlledCT,
  annotLevel,
  sct_data,
  hits
)
```

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.

Value

Null output.

check_group_name *Check group name*

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*

Description

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

Usage

```
check_numeric(exp)
```

Arguments

exp	Input matrix.

Value

Numeric expression matrix.

check_percent_hits *Get percentage of target cell type hits*

Description

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

Usage

```
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.

Value

Report list.

Examples

```
## 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

```
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

```
check_species(
  genelistSpecies = NULL,
  sctSpecies = NULL,
  sctSpecies_origin = NULL,
  sctSpecies_origin_default = "mouse",
  verbose = TRUE
)
```

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 backgrund.
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

```
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

`controlled_geneset_enrichment` *Celltype controlled geneset enrichment*

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

```
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.
`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
ctd <- ewceData::ctd()
schiz_genes <- ewceData::schiz_genes()
hpsd_genes <- ewceData::hpsd_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 an 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)
```

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.

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.

`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.
`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 a several different data sources.
`verbose`	Print messages.

Value

Background and gene list.

create_list_network create_list_network

Description

Support function for prepare_genesize_control_network.

Usage

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

Value

List network

ctd_to_sce *CellTypeDataset to SingleCellExperiment*

Description

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

Usage

```
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.

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.

drop_uninformative_genes *Drop uninformative genes*

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 a 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 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
exp2_orth <- 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)
```

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

`total_res`	Results data.frame 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. Multiple testing correction is then applied across all merged results.
`mtc_method`	Method to be used for multiple testing correction. Argument is passed to p.adjust (DEFAULT: "bonferroni).
`q_threshold`	Corrected significance threshold.
`ctd`	CellTypeDataset object. Should be provided so that the dendrogram can be taken from it and added to plots.
`annotLevel`	An integer indicating which level of `ctd` to analyse (*Default: 1*).
`heights`	The relative heights row in the grid. Will get repeated to match the dimensions of the grid. Passed to wrap_plots.
`make_dendro`	Add a dendrogram (requires `ctd`).
`verbose`	Print messages.

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

```
## 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)
```

`example_bootstrap_results`
Example bootstrap enrichment results

Description

Example cell type enrichment results produced by bootstrap_enrichment_test.

Usage

```
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

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

Load gene list from Alzheimer's disease GWAS

example_genelist <- 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_transcriptome_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

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()

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

```
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 =TRUE 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).

`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 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

```
ctd <- ewceData::ctd()
ctd <- standardise_ctd(ctd, input_species="mouse")
gene_subset <- rownames(ctd[[1]]$mean_exp)[1:100]
ctd_subset <- EWCE::filter_ctd_genes(ctd = ctd, gene_subset = gene_subset)
```

`filter_genes_without_1to1_homolog`
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_1
is now a wrapper for filter_nonorthologs.

Value

List of the filtered CellTypeData file names.

Examples

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

---

filter_nonorthologs *Filter non-orthologs*

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.
- `"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.

`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=<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_1` is now a wrapper for `filter_nonorthologs`.

Value

List of the filtered CellTypeData file names.

Examples

```
# 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

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

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.

---

fix_bad_hgnc_symbols *fix_bad_hgnc_symbols*

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

```
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.

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 synonm 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/repor
`printAllBadSymbols`	Output to console all the bad gene symbols
`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.

Value

Returns the expression matrix with the rownames corrected and rows representing the same gene merged. If no corrections are necessary, input expression matrix is returned. 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

```
# Load the single cell data
cortex_mrna <- ewceData::cortex_mrna()
# take a subset for speed
cortex_mrna$exp <- cortex_mrna$exp[1:50, 1:5]
cortex_mrna$exp <- fix_bad_mgi_symbols(cortex_mrna$exp)
```

---

fix_celltype_names *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

```
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", "astryocytes", "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_dir = file.path(tempdir(), "BootstrapPlots"),
  show_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

```
## 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(
#    sct_data = sct_data,
#    hits = hits,
#    reps = reps,
#    annotLevel = 1,
#    sctSpecies = "mouse",
#    genelistSpecies = "human"
# )

output <- EWCE::generate_bootstrap_plots(
    sct_data = sct_data,
    hits = hits,
    reps = reps,
    full_results = full_results,
    sctSpecies = "mouse",
    genelistSpecies = "human",
    annotLevel = 1
)
```

`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"),
  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.

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

```
## 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.
file_prefix	Prefix to add to saved CTD file name.
as_sparse	Convert exp to a sparse Matrix.
as_DelayedArray	Convert exp to DelayedArray.
normSpec	Boolean indicating whether specificity data should be transformed to a normal distribution by cell type, giving equivalent scores across all 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	The species that the exp dataset comes from. See list_species for all available species.
output_species	Species to convert exp to (Default: "human"). 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 a several different data sources.

force_new_file If a file of the same name as the one being created already exists, overwrite it.

specificity_quantiles
Compute specificity quantiles. Recommended to set to TRUE.

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.

`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 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

File names for the saved CellTypeData (CTD) files.

Examples

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

`generate_controlled_bootstrap_geneset` *generate_controlled_bootstrap_geneset*

Description

Used to generated cell type-controlled bootstrapped gene sets.

Usage

```
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_geneset_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 *get_celltype_table*

Description

get_celltype_table Generates a table that can be used for supplemenary 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")).

Value

List of levels in ctd.

get_ctd_matrix_names *Get CTD matrix names*

Description

Find the names of all data matrices in a CellTypeDataset.

Usage

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

Arguments

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.

get_exp_data_for_bootstrapped_genes *get_exp_data_for_bootstrapped_genes*

Description

Support function for generate_bootstrap_plots_for_transcriptome.

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
)
```

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

```
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()
```

Value

Null

`is_celltypedataset` *Check whether object is a CellTypeDataset*

Description

Check whether an object is a CellTypeDataset.

Usage

```
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

```
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

```
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

```
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 *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

```
load_rdata(fileName)
```

Arguments

fileName	Name of the file to load.

Value

Data object.

Examples

```
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

```
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 targetting 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

```
# Load the single cell data
ctd <- 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,
    tt = tt_alzh_BA44,
    thresh = thresh,
    annotLevel = 1,
    reps = reps,
    ttSpecies = "human",
    sctSpecies = "mouse"
)

# Fill a list with the results
results <- EWCE::add_res_to_merging_list(tt_results_36)
results <- EWCE::add_res_to_merging_list(tt_results_44, results)

# Perform the merged analysis
# For publication reps should be higher
merged_res <- EWCE::merged_ewce(
    results = results,
    reps = 2
)
print(merged_res)
```

`merge_ctd` *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_ctd(
  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,
  numberOfBins = 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 c(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_ctd.

Value

List of CellTypeDatasets or SingleCellExperiments.

Examples

```
## Let's pretend these are different CTD datasets
ctd1 <- ewceData::ctd()
ctd2 <- ctd1
CTD_list <- list(ctd1, ctd2)
CTD_merged <- EWCE::merge_ctd(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

```
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)

Source

scMerge.

Examples

```
ctd <- ewceData::ctd()
sce_list <- EWCE::ctd_to_sce(object = ctd)
sce_combine <- merge_sce(sce_list = sce_list)
```

---

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_DelayedArray`	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.

`myScalesComma` `myScalesComma`

Description

Adjusts **ggplot2** 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

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

plot_log_bootstrap_distributions *Plot log bootstrap distributions*

Description

Plot results of generate_bootstrap_plots_for_transcriptome.

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.

plot_with_bootstrap_distributions *Plot with bootstrap distributions*

Description

Plot results of 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.

`prep.dendro` *prep.dendro*

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

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.
reps	Number of gene lists to sample.
no_cores	Number of cores to parallelise bootstrapping reps over.
sctSpecies	Species that sct_data is currently formatted as (no longer limited to just "mouse" and "human"). See list_species for all available species.
genelistSpecies	Species that hits genes came from (no longer limited to just "mouse" and "human"). See list_species for all available species.
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 three data frames:

- 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

---

prepare_tt *Prepare differential gene expression table*

Description

Prepare differential gene expression table for generate_bootstrap_plots_for_transcriptome or ewce_expression_data.

Usage

```
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 a 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

```
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

```
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.

Value

`DESeq` results

`run_limma` *Run DGE:* **limma**

Description

Run Differential Gene Expression with **limma**.

Usage

```
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. #' @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 *Run DGE:* **MAST**

Description

Run Differential Gene Expression with **MAST**.

Usage

```
run_mast(exp, level2annot, 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. *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

MAST results

Source

MAST tutorial

---

sce_lists_apply *sce_lists_apply*

Description

Support function for EWCE::merge_sce_list.

Usage

```
sce_lists_apply(
  SCE_lists,
  return_genes = FALSE,
  level = 2,
  as_matrix = FALSE,
  as_DelayedArray = FALSE
)
```

Value

List of SingleCellExperiments.

---

sce_merged_apply *sce_merged_apply*

Description

Merge a list of SingleCellExperiments.

Usage

```
sce_merged_apply(SCE_merged, as_sparse = TRUE, as_DelayedArray = FALSE)
```

Value

Merged SingleCellExperiment.

sct_normalize *Normalize expression matrix*

Description

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

Usage

```
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

```
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

```
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_annot = 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 backgrund.
`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.

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 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.

Value

data.frame.

to_delayed_array *Convert object to DelayedArray*

Description

Convert a variety of object types to DelayedArray format.

Usage

```
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

```
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