

# Package ‘EWCE’

April 3, 2023

**Type** Package

**Title** Expression Weighted Celltype Enrichment

**Version** 1.6.0

**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, dplyr, ggplot2, reshape2, limma, stringr, HGNCHELPER, Matrix, parallel, SingleCellExperiment, SummarizedExperiment, DelayedArray, BiocParallel, orthogene (>= 0.99.8), data.table

**Suggests** remotes, knitr, BiocStyle, rmarkdown, testthat (>= 3.0.0), readxl, memoise, markdown, sctransform, DESeq2, MAST, DelayedMatrixStats, cowplot, gg dendro, grDevices, grid, gridExtra, scales, magick, badger

**biocViews** GeneExpression, Transcription, DifferentialExpression, GeneSetEnrichment, Genetics, Microarray, mRNA Microarray, OneChannel, RNASeq, BiomedicalInformatics, Proteomics, Visualization, FunctionalGenomics, SingleCell

**RoxygenNote** 7.2.1

**Encoding** UTF-8

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/EWCE>

**git\_branch** RELEASE\_3\_16

**git\_last\_commit** e37a8f2

**git\_last\_commit\_date** 2022-11-01

**Date/Publication** 2023-04-03

**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>

## R topics documented:

EWCE-package . . . . .	3
add_res_to_merging_list . . . . .	4
bin_columns_into_quantiles . . . . .	5
bin_specificity_into_quantiles . . . . .	5
bootstrap_enrichment_test . . . . .	6
check_ewce_genelist_inputs . . . . .	8
check_percent_hits . . . . .	10
controlled_geneset_enrichment . . . . .	11
ctd_to_sce . . . . .	13
drop_uninformative_genes . . . . .	14
ewce_expression_data . . . . .	17
ewce_plot . . . . .	19
example_bootstrap_results . . . . .	21
example_transcriptome_results . . . . .	22
filter_ctd_genes . . . . .	23
filter_genes_without_1to1_homolog . . . . .	23
filter_nonorthologs . . . . .	24
fix_bad_hgnc_symbols . . . . .	27
fix_bad_mgi_symbols . . . . .	28
fix_celltype_names . . . . .	29
generate_bootstrap_plots . . . . .	30
generate_bootstrap_plots_for_transcriptome . . . . .	32
generate_celltype_data . . . . .	35
get_celltype_table . . . . .	38
is_delayed_array . . . . .	39
is_matrix . . . . .	39
is_sparse_matrix . . . . .	40
list_species . . . . .	40
load_rdata . . . . .	41
merged_ewce . . . . .	41
merge_ctd . . . . .	43
merge_sce . . . . .	44
merge_two_expfiles . . . . .	45
plot_ctd . . . . .	47
prep.dendro . . . . .	48
sct_normalize . . . . .	48
standardise_ctd . . . . .	49

---

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 <a href="#">bootstrap_enrichment_test</a> or <a href="#">ewce_expression_data</a> 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)
```

---

```
bin_columns_into_quantiles
      bin_columns_into_quantiles
```

---

### Description

bin\_columns\_into\_quantiles is an internal function used to convert a matrix of specificity (with columns of cell types) into a matrix of specificity quantiles

### Usage

```
bin_columns_into_quantiles(
  matrixIn,
  numberOfBins = 40,
  defaultBin = as.integer(numberOfBins/2)
)
```

### Arguments

matrixIn	The matrix 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 matrix with same shape as matrixIn 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,
  numberOfBins = 40
)
```

---

```
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,
  output_species = "human",
```

```

    method = "homologene",
    reps = 100,
    no_cores = 1,
    annotLevel = 1,
    geneSizeControl = FALSE,
    controlledCT = NULL,
    mtc_method = "BH",
    sort_results = TRUE,
    verbose = TRUE
)

```

## Arguments

sct_data	List generated using <a href="#">generate_celltype_data</a> .
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 <a href="#">list_species</a> for all available species.
sctSpecies	Species that sct_data came from (no longer limited to just "mouse" and "human"). See <a href="#">list_species</a> for all available species.
output_species	Species to convert sct_data and hits to (Default: "human"). See <a href="#">list_species</a> for all available species.
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
reps	Number of random gene lists to generate ( <i>Default: 100</i> , but should be $\geq 10,000$ 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 ( <i>Default: 1</i> ).
geneSizeControl	Whether you want to control for GC content and transcript length. Recommended if the gene list originates from genetic studies ( <i>Default: FALSE</i> ). 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 <a href="#">p.adjust</a> ).
sort_results	Sort enrichment results from smallest to largest p-values.
verbose	Print messages.

**Value**

A list containing three 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
- **hit.cells**: vector containing the summed proportion of expression in each cell type for the target list
- **bootstrap\_data**: 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 >=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"
)
```

---

```
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 genes 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,
    output_species = "human",
    method = "homologene",
    geneSizeControl = FALSE,
    standardise = FALSE,
    verbose = TRUE
)

```

## Arguments

sct_data	List generated using <a href="#">generate_celltype_data</a> .
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 <a href="#">list_species</a> for all available species.
sctSpecies	Species that sct_data came from (no longer limited to just "mouse" and "human"). See <a href="#">list_species</a> for all available species.
output_species	Species to convert sct_data and hits to (Default: "human"). See <a href="#">list_species</a> for all available species.
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
geneSizeControl	Whether you want to control for GC content and transcript length. Recommended if the gene list originates from genetic studies ( <i>Default: FALSE</i> ). If set to TRUE, then hits must be from humans.
standardise	If input_species==output_species, should the genes be standardised using <a href="#">map_genes</a> ?
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_percent_hits	<i>Get percentage of target cell type hits</i>
--------------------	--

---

**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"
)
```

---

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 <a href="#">generate_celltype_data</a> .
sctSpecies	Species that sct_data came from (no longer limited to just "mouse" and "human"). See <a href="#">list_species</a> for all available species.
output_species	Species to convert sct_data and hits to (Default: "human"). See <a href="#">list_species</a> 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: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
annotLevel	An integer indicating which level of sct_data to analyse ( <i>Default: 1</i> ).
reps	Number of random gene lists to generate ( <i>Default: 100</i> , but should be $\geq 10,000$ 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"
)
```

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)
```

---

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,
  dge_method = "limma",
  dge_test = "LRT",
  mtc_method = "BH",
  min_variance_decile = NULL,
  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.

dge_method	Which method to use for the Differential Gene Expression (DGE) step. Options: <ul style="list-style-type: none"> <li>• "limma": Uses <b>limma</b>.</li> <li>• "deseq2": Uses <b>DESeq2</b>.</li> <li>• "mast": Uses <b>MAST</b>.</li> </ul>
dge_test	test argument passed to DGE function. Only used when dge_method="deqseq2".
mtc_method	Multiple-testing correction method used by DGE step. See <a href="#">p.adjust</a> for more details.
min_variance_decile	If min_variance_decile!=NULL, calculates the variance of the mean gene expression across 'level2annot' (i.e. cell types), and then removes any genes that are below min_variance_decile (on a 0-1 scale). #' @inheritParams ortho-gene::convert_orthologs
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 <a href="#">list_species</a> for all available species.
output_species	Which species' genes names to convert exp to. See <a href="#">list_species</a> for all available species.
non121_strategy	How to handle genes that don't have 1:1 mappings between input_species:output_species. Options include: <ul style="list-style-type: none"> <li>• "drop_both_species" or "dbs" or 1 : Drop genes that have duplicate mappings in either the input_species or output_species (<i>DEFAULT</i>).</li> <li>• "drop_input_species" or "dis" or 2 : Only drop genes that have duplicate mappings in the input_species.</li> <li>• "drop_output_species" or "dos" or 3 : Only drop genes that have duplicate mappings in the output_species.</li> <li>• "keep_both_species" or "kbs" or 4 : Keep all genes regardless of whether they have duplicate mappings in either species.</li> <li>• "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.</li> <li>• "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.</li> </ul>

method	<p>R package to use for gene mapping:</p> <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
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 <b>SingleCellExperiment</b> .
no_cores	Number of cores to parallelise across. Set to NULL to automatically optimise.
verbose	Print messages.
...	<p>Arguments passed on to <a href="#">orthogene::convert_orthologs</a></p> <p>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:</p> <ul style="list-style-type: none"> <li>• matrix : A sparse or dense matrix.</li> <li>• data.frame : A data.frame, data.table. or tibble.</li> <li>• codelist : A list or character vector.</li> </ul> <p>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.</p> <p><i>Note:</i> If you set method="homologene", you must either supply genes in gene symbol format (e.g. "Sox2") OR set standardise_genes=TRUE.</p> <p>gene_input Which aspect of gene_df to get gene names from:</p> <ul style="list-style-type: none"> <li>• "rownames" : From row names of data.frame/matrix.</li> <li>• "colnames" : From column names of data.frame/matrix.</li> <li>• &lt;column name&gt; : From a column in gene_df, e.g. "gene_names".</li> </ul> <p>gene_output How to return genes. Options include:</p> <ul style="list-style-type: none"> <li>• "rownames" : As row names of gene_df.</li> <li>• "colnames" : As column names of gene_df.</li> </ul>



- "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.

## 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"
)
```

---

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
)
```

**Arguments**

sct_data	List generated using <a href="#">generate_celltype_data</a> .
annotLevel	An integer indicating which level of sct_data to analyse ( <i>Default: 1</i> ).
tt	Differential expression table. Can be output of <a href="#">topTable</a> 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 ( <i>Default: "t"</i> ).
thresh	The number of up- and down- regulated genes to be included in each analysis ( <i>Default: 250</i> ).
reps	Number of random gene lists to generate ( <i>Default: 100</i> , but should be $\geq 10,000$ for publication-quality results).
ttSpecies	The species the differential expression table was generated from.
sctSpecies	Species that sct_data came from (no longer limited to just "mouse" and "human"). See <a href="#">list_species</a> for all available species.
output_species	Species to convert sct_data and hits to ( <i>Default: "human"</i> ). See <a href="#">list_species</a> 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: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
verbose	Print messages.

**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",
  ctd = NULL,
  align = "v",
  rel_heights = c(0.3, 1),
  axis = "lr"
)
```

**Arguments**

<code>total_res</code>	Results data.frame generated using <a href="#">bootstrap_enrichment_test</a> or <a href="#">ewce_expression_data</a> 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.
<code>mtc_method</code>	Method to be used for multiple testing correction. Argument is passed to <a href="#">p.adjust</a> (DEFAULT: "bonferroni").
<code>ctd</code>	CellTypeDataset object. Should be provided so that the dendrogram can be taken from it and added to plots
<code>align</code>	(optional) Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. Options are "none" (default), "hv" (align in both directions), "h", and "v".
<code>rel_heights</code>	(optional) Numerical vector of relative rows heights. Works just as <code>rel_widths</code> does, but for rows rather than columns.
<code>axis</code>	(optional) Specifies whether graphs should be aligned by the left ("l"), right ("r"), top ("t"), or bottom ("b") margins. Options are "none" (default), or a string of any combination of l, r, t, and b in any order (e.g. "tblr" or "rlbt" for aligning all margins). Must be specified if any of the graphs are complex (e.g. faceted) and alignment is specified and desired. See <a href="#">align_plots()</a> for details.

**Value**

A ggplot containing the plot

**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()

## Generate the plot
print(EWCE::ewce_plot(
  total_res = full_results$results,
  mtc_method = "BH"
))
```

---

`example_bootstrap_results`*Example bootstrap enrichment results*

---

**Description**

Example cell type enrichment results produced by [bootstrap\\_enrichment\\_test](#).

**Usage**

```
example_bootstrap_results(verbose = TRUE)
```

**Arguments**

`verbose`            Print messages.

**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 <- EWCE::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)
```

**Arguments**

`verbose`            Print messages.

**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()
```

---

filter_ctd_genes	<i>Filter genes in a CellTypeDataset</i>
------------------	--

---

**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	<i>filter_genes_without_1to1_homolog</i>
-----------------------------------	--

---

**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 <i>.rda</i> 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 <i>.rda</i> ).
verbose	Print messages.

**Details**

**Note:** This function replaces the original filter\_genes\_without\_1to1\_homolog function. filter\_genes\_without\_1to1\_ 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	<i>Filter non-orthologs</i>
---------------------	-----------------------------

---

**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 <i>.rda</i> 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 <i>.rda</i> ).
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
non121_strategy	How to handle genes that don't have 1:1 mappings between input_species:output_species. Options include: <ul style="list-style-type: none"> <li>• "drop_both_species" or "dbs" or 1 : Drop genes that have duplicate mappings in either the input_species or output_species (DEFAULT).</li> <li>• "drop_input_species" or "dis" or 2 : Only drop genes that have duplicate mappings in the input_species.</li> <li>• "drop_output_species" or "dos" or 3 : Only drop genes that have duplicate mappings in the output_species.</li> <li>• "keep_both_species" or "kbs" or 4 : Keep all genes regardless of whether they have duplicate mappings in either species.</li> <li>• "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.</li> <li>• "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.</li> </ul>
verbose	Print messages.
...	Arguments passed on to <a href="#">orthogene::convert_orthologs</a>
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: <ul style="list-style-type: none"> <li>• matrix : A sparse or dense matrix.</li> </ul>

- `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.

## Details

**Note:** This function replaces the original filter\_genes\_without\_1to1\_homolog function. filter\_genes\_without\_1to1\_ 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(filenamees = tmp)
```

---

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
)
```

**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.

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

---

**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
)
```

**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 <a href="http://www.informatics.jax.org/downloads/reports/informatics.jax.org/downloads/reports/mrk_list2">www.informatics.jax.org/downloads/reports/informatics.jax.org/downloads/reports/mrk_list2</a>
printAllBadSymbols	Output to console all the bad gene symbols
as_sparse	Convert exp to sparse matrix.
verbose	Print messages.

**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	<i>Fix celltype names</i>
--------------------	---------------------------

---

**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 = "[~]|[.]|[_]|[/]|[\\]|[\\"]")
```

**Arguments**

celltypes	Character vector of celltype names.
replace_chars	Regex string of characters to replace with "_" when renaming columns.

**Value**

Fixed celltype names.

**Examples**

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

---

```
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,
  full_results = NA,
  listFileName = "",
  savePath = tempdir(),
  verbose = TRUE
)
```

**Arguments**

sct_data	List generated using <a href="#">generate_celltype_data</a> .
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 <a href="#">list_species</a> for all available species.
sctSpecies	Species that sct_data came from (no longer limited to just "mouse" and "human"). See <a href="#">list_species</a> for all available species.
output_species	Species to convert sct_data and hits to (Default: "human"). See <a href="#">list_species</a> for all available species.

method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
reps	Number of random gene lists to generate ( <i>Default: 100</i> , but should be $\geq 10,000$ for publication-quality results).
annotLevel	An integer indicating which level of sct_data to analyse ( <i>Default: 1</i> ).
full_results	The full output of <a href="#">bootstrap_enrichment_test</a> for the same gene list.
listFileName	String used as the root for files saved using this function.
savePath	Directory where the BootstrapPlots folder should be saved, default is a temp directory.
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 filename 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
ctd <- 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()[1:100]

## 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 = ctd,
```

```

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

plot_file_path <- EWCE::generate_bootstrap_plots(
  sct_data = ctd,
  hits = hits,
  reps = reps,
  full_results = full_results,
  listFileName = "Example",
  sctSpecies = "mouse",
  genelistSpecies = "human",
  annotLevel = 1,
  savePath = tempdir()
)

```

---

```

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"),
savePath = tempdir(),
method = "homologene",
verbose = TRUE
)

```

## Arguments

sct_data	List generated using <a href="#">generate_celltype_data</a> .
tt	Differential expression table. Can be output of <a href="#">topTable</a> 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 <a href="#">ewce_expression_data</a> 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 came from (no longer limited to just "mouse" and "human"). See <a href="#">list_species</a> for all available species.
output_species	Species to convert sct_data and hits to (Default: "human"). See <a href="#">list_species</a> 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.
savePath	Directory where the <i>BootstrapPlots</i> folder should be saved, default is a temp directory.
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
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 filename 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
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 Cell-TypeData (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. <i>NOTE:</i> 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 <a href="#">list_species</a> for all available species.
output_species	Species to convert exp to (Default: "human"). See <a href="#">list_species</a> for all available species.
non121_strategy	How to handle genes that don't have 1:1 mappings between input_species:output_species. Options include: <ul style="list-style-type: none"> <li>• "drop_both_species" or "dbs" or 1 : Drop genes that have duplicate mappings in either the input_species or output_species (<i>DEFAULT</i>).</li> <li>• "drop_input_species" or "dis" or 2 : Only drop genes that have duplicate mappings in the input_species.</li> <li>• "drop_output_species" or "dos" or 3 : Only drop genes that have duplicate mappings in the output_species.</li> <li>• "keep_both_species" or "kbs" or 4 : Keep all genes regardless of whether they have duplicate mappings in either species.</li> <li>• "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.</li> <li>• "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.</li> </ul>
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
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 <code>orthogene::convert_orthologs</code>

`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.

### 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"
)
```

---

`get_celltype_table`      *get\_celltype\_table*

---

### Description

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

### Usage

```
get_celltype_table(annot)
```

### Arguments

**annot**                      An annotation dataframe, which columns named 'level1class', 'level2class' and 'dataset\_name'

**Value**

A dataframe with columns 'name', 'level', 'freq' and 'dataset\_name'

**Examples**

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

---

is_delayed_array	<i>Assess whether an object is a DelayedArray.</i>
------------------	--

---

**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	<i>Assess whether an object is a Matrix</i>
-----------	---

---

**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	<i>Assess whether an object is a sparse matrix</i>
------------------	--

---

**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	<i>List all species</i>
--------------	-------------------------

---

**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)
```

---

merged_ewce	<i>Multiple EWCE results from multiple studies</i>
-------------	--

---

**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 <a href="#">SingleCellExperiments</a> . 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 <i>standardized_CTD_SCE</i> .
save_split_CTD	Whether to save individual CTD files in the subdirectory <i>standardized_CTD</i> .

```

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	<i>Merge multiple SingleCellExperiment objects</i>
-----------	--

---

### 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_two_expfiles	<i>Merge two exp files</i>
--------------------	----------------------------

---

**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"  
    )
```

---

plot_ctd	<i>Plot CellTypeData metrics</i>
----------	----------------------------------

---

**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: <ul style="list-style-type: none"><li>• "mean_exp"</li><li>• "specificity"</li><li>• "specificity_quantiles"</li></ul>
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"))
```

---

prep.dendro	<i>prep.dendro</i>
-------------	--------------------

---

**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.

---

sct_normalize	<i>Normalize expression matrix</i>
---------------	------------------------------------

---

**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	<i>Convert a CellTypeDataset into standardized format</i>
-----------------	---

---

## 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",
  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 <a href="#">list_species</a> for all available species.
output_species	Which species' genes names to convert exp to. See <a href="#">list_species</a> for all available species.
non121_strategy	How to handle genes that don't have 1:1 mappings between input_species:output_species. Options include:

	<ul style="list-style-type: none"> <li>• "drop_both_species" or "dbs" or 1 : Drop genes that have duplicate mappings in either the input_species or output_species (DEFAULT).</li> <li>• "drop_input_species" or "dis" or 2 : Only drop genes that have duplicate mappings in the input_species.</li> <li>• "drop_output_species" or "dos" or 3 : Only drop genes that have duplicate mappings in the output_species.</li> <li>• "keep_both_species" or "kbs" or 4 : Keep all genes regardless of whether they have duplicate mappings in either species.</li> <li>• "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.</li> <li>• "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.</li> </ul>
method	<p>R package to use for gene mapping:</p> <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
force_new_quantiles	<p>By default, quantile computation is skipped if they have already been computed. Set =TRUE to override this and generate new quantiles.</p>
force_standardise	<p>If ctd has already been standardised, whether to rerun standardisation anyway (Default: FALSE).</p>
remove_unlabeled_clusters	<p>Remove any samples that have numeric column names.</p>
numberOfBins	<p>Number of non-zero quantile bins.</p>
keep_annot	<p>Keep the column annotation data if provided.</p>
keep_plots	<p>Keep the dendrograms if provided.</p>
as_sparse	<p>Convert to sparse matrix.</p>
as_DelayedArray	<p>Convert to DelayedArray.</p>
rename_columns	<p>Remove replace_chars from column names.</p>
make_columns_unique	<p>Rename each columns with the prefix dataset.species.celltype.</p>
verbose	<p>Print messages. Set verbose=2 if you want to print all messages from internal functions as well.</p>
...	<p>Arguments passed on to <a href="#">orthogene::convert_orthologs</a></p>

**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.

**Value**

Standardised `CellTypeDataset`.

**Examples**

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

# Index

add\_res\_to\_merging\_list, [4](#), [41](#)  
aggregate\_mapped\_genes, [17](#), [26](#), [38](#), [51](#)  
align\_plots(), [20](#)  
  
bin\_columns\_into\_quantiles, [5](#)  
bin\_specificity\_into\_quantiles, [5](#)  
bootstrap\_enrichment\_test, [4](#), [6](#), [10](#), [20](#),  
[21](#), [31](#)  
  
check\_ewce\_genelist\_inputs, [8](#)  
check\_percent\_hits, [10](#)  
controlled\_geneset\_enrichment, [11](#)  
ctd\_to\_sce, [13](#)  
  
DelayedArray, [27](#)  
drop\_uninformative\_genes, [14](#)  
  
EWCE (EWCE-package), [3](#)  
EWCE-package, [3](#)  
ewce\_expression\_data, [4](#), [17](#), [20](#), [22](#), [33](#)  
ewce\_plot, [19](#)  
example\_bootstrap\_results, [21](#)  
example\_transcriptome\_results, [22](#)  
  
filter\_ctd\_genes, [23](#)  
filter\_genes\_without\_1to1\_homolog, [23](#)  
filter\_nonorthologs, [23](#), [24](#)  
fix\_bad\_hgnc\_symbols, [27](#)  
fix\_bad\_mgi\_symbols, [28](#)  
fix\_celltype\_names, [29](#)  
  
generate\_bootstrap\_plots, [30](#)  
generate\_bootstrap\_plots\_for\_transcriptome,  
[32](#)  
generate\_celltype\_data, [7](#), [9](#), [12](#), [18](#), [30](#),  
[33](#), [35](#)  
get\_celltype\_table, [38](#)  
gorth, [17](#), [26](#), [38](#), [51](#), [52](#)  
  
is\_delayed\_array, [39](#)  
is\_matrix, [39](#)  
  
is\_sparse\_matrix, [40](#)  
  
list\_species, [7](#), [9](#), [12](#), [15](#), [18](#), [30](#), [33](#), [36](#), [40](#),  
[49](#)  
load\_rdata, [41](#)  
  
map\_genes, [9](#)  
map\_species, [26](#), [40](#)  
merge\_ctd, [43](#)  
merge\_sce, [44](#)  
merge\_two\_expfiles, [45](#)  
merged\_ewce, [41](#)  
  
orthogene::convert\_orthologs, [16](#), [25](#), [37](#),  
[50](#)  
  
p.adjust, [7](#), [15](#), [20](#)  
plot\_ctd, [47](#)  
prep.dendro, [48](#)  
  
sct\_normalize, [48](#)  
SingleCellExperiment, [43](#)  
standardise\_ctd, [49](#)  
  
topTable, [18](#), [33](#)