Package 'EWCE'

October 14, 2021

Type Package

Title Expression Weighted Celltype Enrichment

Version 1.0.1

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

License Artistic-2.0

Depends R(>= 4.1), RNOmni (>= 1.0)

VignetteBuilder knitr

Imports AnnotationHub, ewceData, ExperimentHub, ggplot2, grDevices, grid, reshape2, biomaRt, limma, stringr, cowplot, HGNChelper, ggdendro, gridExtra, Matrix, methods, parallel, future, scales, SummarizedExperiment, stats, utils

Suggests devtools, knitr, BiocStyle, rmarkdown, testthat (>= 3.0.0), data.table, sctransform, readxl, SingleCellExperiment, memoise, markdown

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

RoxygenNote 7.1.1

Encoding UTF-8

Config/testthat/edition 3

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

git_branch RELEASE_3_13

git_last_commit 680dd7d

git_last_commit_date 2021-06-17

Date/Publication 2021-10-14

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

Maintainer Alan Murphy <alanmurph94@hotmail.com>

R topics documented:

`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
library(ewceData)
ctd <- ctd()

# Load the data
tt_alzh <- tt_alzh()
#tt_alzh_BA36 <- 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) intom a matrix of specificity quantiles

Usage

```
bin_columns_into_quantiles(matrixIn, numberOfBins = 40)
```

Arguments

matrixIn	The matrix of specificity values
numberOfBins	Number of quantile 'bins' to use (40 is recommended)

Value

A matrix with same shape as matrixIn but with columns storing quantiles instead of specificity

Examples

```
library(ewceData)
ctd <- 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)
```

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)

Value

A ctd with $specificity_quantiles

Examples

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

`bootstrap_enrichment_test` *Bootstrap celltype 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 = NA,
  hits = NA,
  bg = NA,
  genelistSpecies = "mouse",
  sctSpecies = "mouse",
  reps = 100,
  annotLevel = 1,
  geneSizeControl = FALSE,
  controlledCT = NULL
)
```

Arguments

`sct_data`	List generated using `generate_celltype_data`
`hits`	Array of MGI gene symbols containing the target gene list. Must be HGNC symbols if geneSizeControl=TRUE
`bg`	Array of MGI gene symbols containing the background gene list. Must be HGNC symbols if geneSizeControl=TRUE
`genelistSpecies`	Either 'mouse' or 'human' depending on whether MGI or HGNC symbols are used for gene lists
`sctSpecies`	Either 'mouse' or 'human' depending on whether MGI or HGNC symbols are used for the single cell dataset
`reps`	Number of random gene lists to generate (default=100 but should be over 10000 for publication quality results)
`annotLevel`	an integer indicating which level of the annotation to analyse. Default = 1.

`geneSizeControl`
: a logical indicating whether you want to control for GC content and transcript length. Recommended if the gene list originates from genetic studies. Default is FALSE. If set to TRUE then human gene lists should be used rather than mouse.

`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

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

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

# Set the parameters for the analysis
# Use 3 bootstrap lists for speed, for publishable analysis use >10000
reps <- 3

# Load the gene list and get human orthologs
example_genelist <- example_genelist()
mouse_to_human_homologs <- mouse_to_human_homologs()
m2h <- unique(mouse_to_human_homologs[, c("HGNC.symbol", "MGI.symbol")])
mouse.hits <-
    unique(m2h[m2h$HGNC.symbol %in% example_genelist, "MGI.symbol"])
#subset mouse.bg for speed but ensure it still contains the hits
mouse.bg <- unique(c(m2h$MGI.symbol[1:100],mouse.hits))

# Bootstrap significance test, no control for transcript length or GC content
full_results <- bootstrap_enrichment_test(
    sct_data = ctd, hits = mouse.hits,
    bg = mouse.bg, reps = reps, annotLevel = 2, sctSpecies = "mouse",
    genelistSpecies = "mouse"
)
```

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,
  genelistSpecies,
  sctSpecies,
  geneSizeControl = FALSE
)
```

Arguments

sct_data	List generated using generate_celltype_data
hits	Array of MGI/HGNC gene symbols containing the target gene list.
bg	Array of MGI/HGNC gene symbols containing the background gene list.
genelistSpecies	Either 'mouse' or 'human' depending on whether MGI or HGNC symbols are used for gene lists
sctSpecies	Either 'mouse' or 'human' depending on whether MGI or HGNC symbols are used for the single cell dataset
geneSizeControl	Will genelists sampled control for GC content and transcript length? Boolean.

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

```
library(ewceData)
# Called from "bootstrap_enrichment_test()" and "generate_bootstrap_plots()"
ctd <- ctd()
example_genelist <- example_genelist()
mouse_to_human_homologs <- mouse_to_human_homologs()
m2h <- unique(mouse_to_human_homologs[, c("HGNC.symbol", "MGI.symbol")])
mouse.hits <-
     unique(m2h[m2h$HGNC.symbol %in% example_genelist, "MGI.symbol"])
#subset mouse.bg for speed but ensure it still contains the hits
mouse.bg <- unique(c(m2h$MGI.symbol[1:100],mouse.hits))
checkedLists <- check_ewce_genelist_inputs(
    sct_data = ctd, hits = mouse.hits,
    bg = mouse.bg, genelistSpecies = "mouse", sctSpecies = "mouse"
)
```

---

controlled_geneset_enrichment

Celltype controlled geneset enrichment

Description

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

Usage

```
controlled_geneset_enrichment(
  disease_genes,
  functional_genes,
  bg_genes,
  sct_data,
  annotLevel,
  reps,
  controlledCT
)
```

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 geneset within the disease_genes is tested. Must be from same species as the single cell transcriptome dataset.
`bg_genes`	Array of gene symbols containing the background gene list.

sct_data	List generated using generate_celltype_data
annotLevel	an integer indicating which level of the annotation to analyse. Default = 1.
reps	Number of random gene lists to generate (default=100 but should be over 10000 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

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

```
library(ewceData)
# See the vignette for more detailed explanations
# Gene set enrichment analysis controlling for cell type expression
# set seed for bootstrap reproducibility
set.seed(12345678)
ctd <- ctd()
mouse_to_human_homologs <- mouse_to_human_homologs()
m2h = unique(mouse_to_human_homologs[,c("HGNC.symbol","MGI.symbol")])
schiz_genes <- schiz_genes()
id_genes <- id_genes()
mouse.hits.schiz = unique(m2h[m2h$HGNC.symbol %in% schiz_genes,"MGI.symbol"])
mouse.bg  = unique(m2h$MGI.symbol)
hpsd_genes <- hpsd_genes()
mouse.hpsd = unique(m2h[m2h$HGNC.symbol %in% hpsd_genes,"MGI.symbol"])
# Use 3 bootstrap lists for speed, for publishable analysis use >10000
reps=3
res_hpsd_schiz =
    controlled_geneset_enrichment(disease_genes=mouse.hits.schiz,
                                  functional_genes = mouse.hpsd,
                                  bg_genes = mouse.bg,
                                  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 celltype data structure containing $mean_exp and $specificity
lvl	The annotation level to extract

Value

celltype_data The old style data structure

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 celltype data 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

ctd The new style data structure

`drop_uninformative_genes` *drop_uninformative_genes*

Description

`drop_uninformative_genes` drops genes from an SCT expression matrix if they do not significantly vary between any celltypes. Makes this decision based on use of an ANOVA (implemented with Limma). If the F-statistic for variation amongst type2 annotations is less than a strict p-threshold, then the gene is dropped.

Usage

```
drop_uninformative_genes(exp, level2annot)
```

Arguments

`exp`	Expression matrix with gene names as rownames.
`level2annot`	Array of cell types, with each sequentially corresponding a column in the expression matrix

Value

exp Expression matrix with gene names as rownames.

Examples

```
library(ewceData)
cortex_mrna <- cortex_mrna()
cortex_mrna$exp <- cortex_mrna$exp[1:300, ]
exp2 <- drop_uninformative_genes(exp = cortex_mrna$exp,
    level2annot = cortex_mrna$annot$level2class)
```

`ewce_expression_data` *Bootstrap celltype enrichment test for transcriptome data*

Description

`ewce_expression_data` takes a differential expression table and determines the probability of cell-type enrichment in the up & down regulated genes

Usage

```
ewce_expression_data(
  sct_data,
  annotLevel = 1,
  tt,
  sortBy = "t",
  thresh = 250,
  reps = 100,
  ttSpecies = "mouse",
  sctSpecies = "mouse"
)
```

Arguments

`sct_data`	List generated using `generate_celltype_data`
`annotLevel`	an integer indicating which level of the annotation to analyse. Default = 1.
`tt`	Differential expression table. Can be output of limma::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 either HGNC or MGI 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 over 10000 for publication quality results)
`ttSpecies`	Either 'mouse' or 'human' depending on which species the differential expression table was generated from
`sctSpecies`	Either 'mouse' or 'human' depending on which species the single cell data was generated from

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

```
library(ewceData)
# Load the single cell data
ctd <- 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 <- tt_alzh()

tt_results <- 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)
```

Arguments

total_res	results dataframe 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.
mtc_method	method to be used for multiple testing correction. Argument is passed to p.adjust. Valid options are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY",
ctd	Should be provided so that the dendrogram can be taken from it and added to plots

Value

A ggplot containing the plot

Examples

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

# Set the parameters for the analysis
# Use 3 bootstrap lists for speed, for publishable analysis use >10000
reps <- 3

# Load the gene list and get human orthologs
example_genelist <- example_genelist()
mouse_to_human_homologs <- mouse_to_human_homologs()
m2h <- unique(mouse_to_human_homologs[, c("HGNC.symbol", "MGI.symbol")])
mouse.hits <-
     unique(m2h[m2h$HGNC.symbol %in% example_genelist, "MGI.symbol"])
mouse.bg <- unique(c(m2h$MGI.symbol[1:100],mouse.hits))

# Bootstrap significance test, no control for transcript length or GC content
full_results <- bootstrap_enrichment_test(
    sct_data = ctd, hits = mouse.hits,
    bg = mouse.bg, reps = reps, annotLevel = 2,
    sctSpecies = "mouse", genelistSpecies = "mouse"
)

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

---

filter_genes_without_1to1_homolog

filter_genes_without_1to1_homolog

Description

filter_genes_without_1to1_homolog Takes the filenames of celltype_data files, loads them, and drops any genes which don't have a 1:1 homolog based on biomart. The new files are saved to disc, appending '_1to1only' to the file tag of the previous file.

Usage

```
filter_genes_without_1to1_homolog(filenames)
```

Arguments

filenames	Array of filenames for sct_data saved as .Rda files

Value

Array of filenames included the ones with only 1:1 homologs

Examples

```
library(ewceData)
# Load the single cell data
cortex_mrna <- cortex_mrna()
expData <- cortex_mrna$exp
expData <- expData[1:100, ] # Use only a subset to keep the example quick
l1 <- cortex_mrna$annot$level1class
l2 <- cortex_mrna$annot$level2class
annotLevels <- list(level1class = l1, level2class = l2)
fNames_ALLCELLS <- generate_celltype_data(exp = expData,
    annotLevels = annotLevels,
    groupName = "allKImouse")
fNames_ALLCELLS <- filter_genes_without_1to1_homolog(fNames_ALLCELLS)
```

---

fix_bad_hgnc_symbols *fix_bad_hgnc_symbols - 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.*

Description

fix_bad_hgnc_symbols - 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)
```

Arguments

exp	An expression matrix where the rows are HGNC symbols or an SingleCellExperiment (SCE) or other Ranged Summarized Experiment (SE) type object.
dropNonHGNC	Boolean. Should symbols not recognised as HGNC symbols be dropped?

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)
```

Arguments

`exp`	An expression matrix where the rows are MGI symbols or an SingleCellExperiment (SCE) or other Ranged Summarized Experiment (SE) type object.
`mrk_file_path`	Path to the MRK_List2 file which can be downloaded from www.informatics.jax.org/downloads/reports/ir
`printAllBadSymbols`	Output to console all the bad gene symbols

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

```
library(ewceData)
# Load the single cell data
cortex_mrna <- 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)
```

`generate_bootstrap_plots` *Generate bootstrap plots*

Description

`generate_bootstrap_plots` takes a genelist 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,
  hits,
  bg,
  genelistSpecies = "mouse",
  sctSpecies = "mouse",
  reps,
  annotLevel = 1,
  full_results = NA,
  listFileName = "",
  savePath = tempdir()
)
```

Arguments

`sct_data`	List generated using `generate_celltype_data`
`hits`	Array of MGI/HGNC gene symbols containing the target gene list.
`bg`	Array of MGI/HGNC gene symbols containing the background gene list.
`genelistSpecies`	Either 'mouse' or 'human' depending on whether MGI or HGNC symbols are used for gene lists
`sctSpecies`	Either 'mouse' or 'human' depending on whether MGI or HGNC symbols are used for the single cell dataset
`reps`	Number of random gene lists to generate (default=100 but should be over 10000 for publication quality results)

annotLevel	an integer indicating which level of the annotation to analyse. Default = 1.
full_results	The full output of bootstrap_enrichment_test for the same genelist
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

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 celltype. 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

```
library(ewceData)
# Load the single cell data
ctd <- 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
example_genelist <- example_genelist()
mouse_to_human_homologs <- mouse_to_human_homologs()
m2h <- unique(mouse_to_human_homologs[, c("HGNC.symbol", "MGI.symbol")])
mouse.hits <-
    unique(m2h[m2h$HGNC.symbol %in% example_genelist, "MGI.symbol"])
#subset mouse.bg for speed but ensure it still contains the hits
mouse.bg <- unique(c(m2h$MGI.symbol[1:100],mouse.hits))

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

plot_file_pth <- generate_bootstrap_plots(
    sct_data = ctd, hits = mouse.hits, bg = mouse.bg,
    reps = reps, full_results = full_results, listFileName = "Example",
    genelistSpecies = "mouse", sctSpecies = "mouse", annotLevel = 1,
    savePath=tempdir()
)
```

`generate_bootstrap_plots_for_transcriptome` *Generate bootstrap plots*

Description

`generate_bootstrap_plots_for_transcriptome` takes a genelist 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,
  thresh = 250,
  annotLevel = 1,
  reps,
  full_results = NA,
  listFileName = "",
  showGNameThresh = 25,
  ttSpecies = "mouse",
  sctSpecies = "mouse",
  sortBy = "t",
  onlySignif = TRUE,
  savePath = tempdir()
)
```

Arguments

`sct_data`	List generated using `generate_celltype_data`
`tt`	Differential expression table. Can be output of limma::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 either HGNC or MGI symbols.
`thresh`	The number of up- and down- regulated genes to be included in each analysis. Dafault=250
`annotLevel`	an integer indicating which level of the annotation to analyse. Default = 1.
`reps`	Number of random gene lists to generate (default=100 but should be over 10000 for publication quality results)
`full_results`	The full output of `bootstrap_enrichment_test` for the same genelist
`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 celltype, then list the gene name

ttSpecies	Either 'mouse' or 'human' depending on which species the differential expression table was generated from
sctSpecies	Either 'mouse' or 'human' depending on which species the single cell data was generated from
sortBy	Column name of metric in tt which should be used to sort up- from down- regulated genes. Default="t"
onlySignif	Should plots only be generated for cells which have significant changes?
savePath	Directory where the BootstrapPlots folder should be saved, default is a temp directory

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 celltype. 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

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

# Set the parameters for the analysis
# Use 3 bootstrap lists for speed, for publishable analysis use >10000
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 <- tt_alzh()

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

# Bootstrap significance test, no control for transcript length or GC content
full_results <- 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", savePath=tempdir()
```

```
)
```

---

generate_celltype_data
generate_celltype_data

Description

generate_celltype_data Takes expression & cell type annotations and creates celltype_data files which contain the mean and specificity matrices

Usage

```
generate_celltype_data(
  exp,
  annotLevels,
  groupName,
  no_cores = 1,
  savePath = tempdir(),
  normSpec = FALSE
)
```

Arguments

exp	Numerical matrix with row for each gene and column for each cell. Row names are MGI/HGNC 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 refering to the dataset being loaded
no_cores	Number of cores that should be used to speedup the computation. Use no_cores = 1 when using this package in windows system.
savePath	Directory where the CTD file should be saved
normSpec	Boolean indicating whether specificity data should be transformed to a normal distribution by cell type, giving equivalent scores across all cell types.

Value

Filenames for the saved celltype_data files

Examples

```
library(ewceData)
# Load the single cell data
cortex_mrna <- cortex_mrna()
expData <- cortex_mrna$exp
expData <- expData[1:100, ] # Use only a subset to keep the example quick
l1 <- cortex_mrna$annot$level1class
l2 <- cortex_mrna$annot$level2class
annotLevels <- list(l1 = l1, l2 = l2)
fNames_ALLCELLS <-
    generate_celltype_data(exp = expData, annotLevels, "allKImouse",
        savePath=tempdir())
```

---

generate_controlled_bootstrap_geneset

generate_controlled_bootstrap_geneset

Description

`generate_controlled_bootstrap_geneset` Used to generated celltype controlled bootstraped

Usage

```
generate_controlled_bootstrap_geneset(
  hitGenes,
  sct_data,
  annotLevel,
  reps,
  controlledCT = NULL
)
```

Arguments

`hitGenes`	Array of gene names. The target gene set.
`sct_data`	The cell type data list (with specificity and mean_exp)
`annotLevel`	The level of annotation in sct_data to analyse
`reps`	The number of gene lists to sample
`controlledCT`	Name of a celltype (from colnames of sct_data[[x]]$specificity).

Value

Matrix of genes (nrows=length(hitGenes),ncols=reps), where each column is a gene list

Examples

```
# See controlled_geneset_enrichment.r
```

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 celltype, 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

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

get_summed_proportions *Get summed proportions*

Description

get_summed_proportions Given the target geneset, randomly sample genelists 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(
  hitGenes,
  sct_data,
  annotLevel,
  reps,
  geneSizeControl,
  controlledCT = NULL,
  control_network = NULL
)
```

Arguments

hitGenes	Array of gene names. The target gene set.
sct_data	The cell type data list (with specificity and mean_exp)
annotLevel	The level of annotation in sct_data to analyse
reps	The number of gene lists to sample
geneSizeControl	Boolean. Should the sampled gene lists control for GC content and transcript length?
controlledCT	Name of a celltype (from colnames of sct_data[[x]]$specificity).
control_network	If geneSizeControl=TRUE, then must provide the control network

Value

A list containing three data frames:

- 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
- controlledCT: the controlled celltype (if applicable)

Examples

```
# See bootstrap_enrichment_test.r
```

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 over 10000 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

```
library(ewceData)
# Load the single cell data
ctd <- 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 <- tt_alzh_BA36()
tt_alzh_BA44 <- tt_alzh_BA44()

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

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

---

merge_two_expfiles *merge_two_expfiles*

Description

merge_two_expfiles Used to combine two single cell type datasets

Usage

```
merge_two_expfiles(exp1, exp2, annot1, annot2, name1 = "", name2 = "")
```

Arguments

exp1	Numerical expression matrix for dataset1 with row for each gene and column for each cell. Row names are MGI/HGNC 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 MGI/HGNC 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.

Value

List containing merged exp and annot

Examples

```
# See the EWCE vignette for use case explanation
```

prepare_genesize_control_network
Prepare genesize control network

Description

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

Usage

```
prepare_genesize_control_network(
  hits,
  bg,
  numBOOT = 10000,
  sctSpecies,
  sct_data
)
```

Arguments

hits	Array of MGI gene symbols containing the target gene list. Must be HGNC symbols.
bg	Array of MGI gene symbols containing the background gene list (including hit genes). Must be HGNC symbols.
numBOOT	Number of gene lists to sample
sctSpecies	Either 'mouse' or 'human' depending on whether MGI or HGNC symbols are used for the single cell dataset
sct_data	List generated using generate_celltype_data

Value

A list containing three data frames:

- hitGenes: 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

Examples

```
# Called by bootstrap_enrichment_test
```

prep_dendro *prep_dendro*

Description

prep_dendro adds a dendrogram to a cts

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 ctd with dendrogram plotting info added

Examples

```
library(ewceData)
ctd <- ctd()
ctd <- lapply(ctd, EWCE::bin_specificity_into_quantiles, numberOfBins = 40)
ctd <- lapply(ctd, EWCE::prep_dendro)
```

Index