Package ‘orthogene’

April 8, 2026

Type Package

Title Interspecies gene mapping

Version 1.17.2

Description `orthogene` is an R package for easy mapping of orthologous genes
across hundreds of species. It pulls up-to-date gene ortholog mappings
across **700+ organisms**.
It also provides various utility functions to aggregate/expand
common objects (e.g. data.frames, gene expression matrices, lists)
using **1:1**, **many:1**, **1:many** or **many:many** gene mappings,
both within- and between-species.

URL https://github.com/neurogenomics/orthogene

BugReports https://github.com/neurogenomics/orthogene/issues

License GPL-3

Depends R (>= 4.5.0)

VignetteBuilder knitr

biocViews Genetics, ComparativeGenomics, Preprocessing, Phylogenetics,
Transcriptomics, GeneExpression

Imports dplyr, methods, stats, utils, Matrix, jsonlite, homologene,
gprofiler2, babelgene, data.table, parallel, ggplot2, ggpubr,
patchwork, DelayedArray, repmis, ggtree, tools, magrittr

Suggests remotes, knitr, BiocStyle, markdown, rmarkdown, testthat (>=
3.0.0), piggyback, magick, GenomeInfoDbData, ape, phytools,
rphylopic (>= 1.0.0), TreeTools, ggimage, OmaDB

RoxygenNote 7.3.3

Encoding UTF-8

Config/testthat/edition 3

Config/rcmdcheck/_R_CHECK_FORCE_SUGGESTS_ false

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

git_branch devel

git_last_commit 3569148

git_last_commit_date 2026-01-07

Repository Bioconductor 3.23

Date/Publication 2026-04-07

Author Brian Schilder [cre, fnd] (ORCID: <`https://orcid.org/0000-0001-5949-2191`>)

Maintainer Brian Schilder <`brian_schilder@alumni.brown.edu`>

Contents

`orthogene-package` **orthogene**: *Interspecies gene mapping*

Description

orthogene is an R package for easy mapping of orthologous genes across hundreds of species.

Details

It pulls up-to-date interspecies gene ortholog mappings across 700+ organisms.
It also provides various utility functions to map common objects (e.g. data.frames, gene expression matrices, lists) onto 1:1 gene orthologs from any other species.

Author(s)

Maintainer: Brian Schilder <`brian_schilder@alumni.brown.edu`> (ORCID) [funder]

Source

GitHub : Source code and Issues submission.

Author Site : `orthogene` was created by Brian M. Schilder.

See Also

Useful links:

- `https://github.com/neurogenomics/orthogene`
- Report bugs at `https://github.com/neurogenomics/orthogene/issues`

`add_synonyms` *Add gene synonyms*

Description

Add gene synonyms back into `gene_map` `data.frame`.

Usage

```
add_synonyms(gene_map, syn_map)
```

Details

`gene_map` is the output of convert_orthologs.

Value

`gene_map` `data.frame`

`aggregate_mapped_genes` *Aggregate/expand a gene matrix by gene mappings*

Description

Aggregate/expand a gene matrix (`gene_df`) using a gene mapping data.frame (`gene_map`). Importantly, mappings can be performed across a variety of scenarios that can occur during within-species and between-species gene mapping:

1 gene : 1 gene

many genes : 1 gene

1 gene : many genes

many genes : many genes

For more details on how aggregation/expansion is performed, please see: many2many_rows.

Usage

```
aggregate_mapped_genes(
  gene_df,
  gene_map = NULL,
  input_col = "input_gene",
  output_col = "ortholog_gene",
  input_species = "human",
  output_species = input_species,
```

```
  method = c("gprofiler", "homologene", "babelgene"),
  agg_fun = "sum",
  agg_method = c("monocle3", "stats"),
  aggregate_orthologs = TRUE,
  transpose = FALSE,
  mthreshold = 1,
  target = "ENSG",
  numeric_ns = "",
  as_integers = FALSE,
  as_sparse = TRUE,
  as_DelayedArray = FALSE,
  dropNA = TRUE,
  sort_rows = FALSE,
  verbose = TRUE
)
```

Arguments

`gene_df`	Input matrix where row names are genes.
`gene_map`	A data.frame that maps the current gene names to new gene names. This function's behaviour will adapt to different situations as follows: `gene_map=<data.frame>` When a data.frame containing the gene key:value columns (specified by `input_col` and `output_col`, respectively) is provided, this will be used to perform aggregation/expansion. `gene_map=NULL` **and** `input_species!=output_species` A `gene_map` is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion. `gene_map=NULL` **and** `input_species==output_species` A `gene_map` is automatically generated by map_genes to perform within-species gene gene symbol standardization and aggregation/expansion.
`input_col`	Column name within `gene_map` with gene names matching the row names of `X`.
`output_col`	Column name within `gene_map` with gene names that you wish you map the row names of `X` onto.
`input_species`	Name of the input species (e.g., "mouse","fly"). Use map_species to return a full list of available species.
`output_species`	Name of the output species (e.g. "human","chicken"). Use map_species to return a full list of available species.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`agg_fun`	Aggregation function.
`agg_method`	Aggregation method.

`aggregate_orthologs`	[Optional] After performing an initial round of many:many aggregation/expansion with many2many_rows, ensure each orthologous gene only appears in one row by using the aggregate_rows function (default: `TRUE`).
`transpose`	Transpose `gene_df` before mapping genes.
`mthreshold`	maximum number of results per initial alias to show. Shows all by default.
`target`	target namespace.
`numeric_ns`	namespace to use for fully numeric IDs (list of available namespaces).
`as_integers`	Force all values in the matrix to become integers, by applying floor (default: `FALSE`).
`as_sparse`	Convert aggregated matrix to sparse matrix.
`as_DelayedArray`	Convert aggregated matrix to DelayedArray.
`dropNA`	Drop genes assigned to `NA` in `groupings`.
`sort_rows`	Sort `gene_df` rows alphanumerically.
`verbose`	Print messages.

Value

Aggregated matrix

Examples

```
#### Aggregate within species: gene synonyms ####
data("exp_mouse_enst")
X_agg <- aggregate_mapped_genes(gene_df = exp_mouse_enst,
                                input_species = "mouse")

#### Aggregate across species: gene orthologs ####
data("exp_mouse")
X_agg2 <- aggregate_mapped_genes(gene_df = exp_mouse,
                                 input_species = "mouse",
                                 output_species = "human",
                                 method="homologene")
```

`aggregate_rows` *Aggregate rows of matrix*

Description

Aggregate rows of a matrix for many:1 mappings, using a grouping vector.

Usage

```
aggregate_rows(
  X,
  groupings,
  agg_fun = "sum",
  agg_method = c("monocle3", "stats"),
  as_sparse = TRUE,
  as_DelayedArray = TRUE,
  dropNA = TRUE,
  verbose = TRUE
)
```

Arguments

`X`	Input matrix.
`groupings`	Gene groups of the same length as `nrow(X)`.
`agg_fun`	Aggregation function.
`agg_method`	Aggregation method.
`as_sparse`	Convert aggregated matrix to sparse matrix.
`as_DelayedArray`	Convert aggregated matrix to DelayedArray.
`dropNA`	Drop genes assigned to `NA` in `groupings`.
`verbose`	Print messages.

Value

Aggregated matrix

Source

`data("exp_mouse_enst")` `X` <- `exp_mouse_enst` `gene_map` <- `map_genes(genes` = `rownames(X),species`
= `"mouse")` `X_agg` <- `orthogene:::aggregate_rows(X` = `X, groupings` = `gene_map$name)` `sum(duplicated(rownames`
`0` `sum(duplicated(rownames(X)))` # 1215 `sum(duplicated(rownames(X_agg)))` # `0`

`aggregate_rows_monocle3`

Aggregate rows: monocle3

Description

Aggregate rows: monocle3

Usage

```
aggregate_rows_monocle3(
  x,
  groupings = NULL,
  form = NULL,
  fun = "sum",
  na.action = stats::na.omit
)
```

Arguments

`x`	Input matrix.
`groupings`	Gene groups of the same length as `nrow(X)`.
`form`	Formula.
`fun`	Aggregation function.
`na.action`	Na action.

Value

Aggregated matrix.

Source

```
 x <- Matrix::rsparsematrix(nrow = 1000, ncol = 2000, density = .10) groupings <- rep(c("A","B"),nrow(x)/2
X2 <- orthogene:::aggregate_rows_monocle3(x = x, groupings=groupings)
```

`all_genes` *Get all genes*

Description

Return all known genes from a given species.

Usage

```
all_genes(
  species,
  method = c("gprofiler", "homologene", "babelgene"),
  ensure_filter_nas = FALSE,
  run_map_species = TRUE,
  force = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`species`	Species to get all genes for. Will first be standardised with `map_species`.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`ensure_filter_nas`	Perform an extra check to remove genes that are `NA`s of any kind.
`run_map_species`	Standardise `species` names with map_species first (Default: `TRUE`).
`force`	Force a new files to be generated even if a cached one exists.
`verbose`	Print messages.
`...`	Additional arguments to be passed to gorth or homologene. *NOTE*: To return only the most "popular" interspecies ortholog mappings, supply `mthreshold=1` here AND set `method="gprofiler"` above. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs. For more details, please see here.

Details

References homologeneData or gconvert.

Value

Table with all gene symbols from the given `species`.

Examples

```
genome_mouse <- all_genes(species = "mouse")
genome_human <- all_genes(species = "human")
```

`all_genes_babelgene` *Get all genes: babelgene*

Description

Get all genes for a given species using the method "babelgene".

Usage

```
all_genes_babelgene(
  species,
  run_map_species = TRUE,
  save_dir = cache_dir(),
  use_old = FALSE,
  min_support = 1,
  force = FALSE,
  verbose = TRUE
)
```

Arguments

`species`	Species to get all genes for. Will first be standardised with `map_species`.
`run_map_species`	Standardise `species` names with map_species first (Default: `TRUE`).
`save_dir`	Directory to save babelgene mapping files to.
`use_old`	Use an old version of `babelgene::orthologs_df` (stored on GitHub Releases) for consistency.
`force`	Force a new files to be generated even if a cached one exists.
`verbose`	Print messages.

Value

All genes.

Source

babelgene::orthologs_df version differences

`all_species` *All species*

Description

List all species currently supported by **orthogene**. Wrapper function for map_species. When `method=NULL`, all species from all available methods will be returned.

Usage

```
all_species(method = NULL, verbose = TRUE)
```

Arguments

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

Value

data.table of species names, provided in multiple formats.

Examples

```
species_dt <- all_species()
```

`cache_dir` *Get cache save dir*

Description

Create a package-specific cache dir.

Usage

```
cache_dir()
```

Value

Path string

`check_bool_args` *Check boolean args*

Description

Check that boolean args are indeed boolean

Usage

```
check_bool_args(
  standardise_genes,
  drop_nonorths,
  as_sparse,
  as_DelayedArray,
  sort_rows
)
```

Arguments

`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`.
`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.

`check_gene_df_type` *Check gene_df*

Description

Handles gene_df regardless of whether it's a data.frame, matrix, list, or vector

Usage

```
check_gene_df_type(gene_df, gene_input, verbose = TRUE)
```

Arguments

`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`. `list` 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"`.
`verbose`	Print messages.

Value

List of gene_df and gene_input

`convert_orthologs` *Map genes from one species to another*

Description

Currently supports ortholog mapping between any pair of 700+ species.
Use map_species to return a full list of available organisms.

Usage

```
convert_orthologs(
  gene_df,
  gene_input = "rownames",
  gene_output = "rownames",
  standardise_genes = FALSE,
  input_species,
  output_species = "human",
  method = c("gprofiler", "homologene", "babelgene"),
  drop_nonorths = TRUE,
  non121_strategy = "drop_both_species",
  agg_fun = NULL,
  mthreshold = Inf,
  as_sparse = FALSE,
  as_DelayedArray = FALSE,
  sort_rows = FALSE,
  gene_map = NULL,
  input_col = "input_gene",
  output_col = "ortholog_gene",
  verbose = TRUE,
  ...
)
```

Arguments

`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`. `list` 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.
`input_species`	Name of the input species (e.g., "mouse","fly"). Use map_species to return a full list of available species.
`output_species`	Name of the output species (e.g. "human","chicken"). Use map_species to return a full list of available species.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`drop_nonorths`	Drop genes that don't have an ortholog in the `output_species`.

`non121_strategy`	How to handle genes that don't have 1:1 mappings between `input_species`:`output_species`. Options include: `"drop_both_species"` or `"dbs"` or `1` Drop genes that have duplicate mappings in either the `input_species` or `output_species` (*DEFAULT*). `"drop_input_species"` or `"dis"` or `2` Only drop genes that have duplicate mappings in the `input_species`. `"drop_output_species"` or `"dos"` or `3` Only drop genes that have duplicate mappings in the `output_species`. `"keep_both_species"` or `"kbs"` or `4` Keep all genes regardless of whether they have duplicate mappings in either species. `"keep_popular"` or `"kp"` or `5` Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs. `"sum"`,`"mean"`,`"median"`,`"min"` or `"max"` When `gene_df` is a matrix and `gene_output="rownames"`, these options will aggregate many-to-one gene mappings (`input_species`-to-`output_species`) after dropping any duplicate genes in the `output_species`.
`agg_fun`	Aggregation function passed to aggregate_mapped_genes. Set to `NULL` to skip aggregation step (default).
`mthreshold`	Maximum number of ortholog names per gene to show. Passed to gorth. Only used when `method="gprofiler"` (*DEFAULT :* `Inf`).
`as_sparse`	Convert `gene_df` to a sparse matrix. Only works if `gene_df` is one of the following classes: • `matrix` • `Matrix` • `data.frame` • `data.table` • `tibble` If `gene_df` is a sparse matrix to begin with, it will be returned as a sparse matrix (so long as `gene_output=` `"rownames"` or `"colnames"`).
`as_DelayedArray`	Convert aggregated matrix to DelayedArray.
`sort_rows`	Sort `gene_df` rows alphanumerically.
`gene_map`	A data.frame that maps the current gene names to new gene names. This function's behaviour will adapt to different situations as follows: `gene_map=<data.frame>` When a data.frame containing the gene key:value columns (specified by `input_col` and `output_col`, respectively) is provided, this will be used to perform aggregation/expansion. `gene_map=NULL` **and** `input_species!=output_species` A `gene_map` is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion.

	`gene_map=NULL` **and** `input_species==output_species` A `gene_map` is automatically generated by map_genes to perform within-species gene gene symbol standardization and aggregation/expansion.
`input_col`	Column name within `gene_map` with gene names matching the row names of `X`.
`output_col`	Column name within `gene_map` with gene names that you wish you map the row names of `X` onto.
`verbose`	Print messages.
`...`	Additional arguments to be passed to gorth or homologene. *NOTE*: To return only the most "popular" interspecies ortholog mappings, supply `mthreshold=1` here AND set `method="gprofiler"` above. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs. For more details, please see here.

Value

`gene_df` with orthologs converted to the `output_species`.
Instead returned as a dictionary (named list) if `gene_output="dict"` or `"dict_rev"`.

Examples

```
data("exp_mouse")
gene_df <- convert_orthologs(
    gene_df = exp_mouse,
    input_species = "mouse"
)
```

`create_background` *Create gene background*

Description

Create a gene background as the union/intersect of all orthologs between input species (`species1` and `species2`), and the `output_species`. This can be useful when generating random lists of background genes to test against in analyses with data from multiple species (e.g. enrichment of mouse cell-type markers gene sets in human GWAS-derived gene sets).

Usage

```
create_background(
  species1,
  species2,
  output_species = "human",
  as_output_species = TRUE,
  use_intersect = TRUE,
```

```
    bg = NULL,
    gene_map = NULL,
    method = "homologene",
    non121_strategy = "drop_both_species",
    verbose = TRUE
  )
```

Arguments

`species1`	First species.
`species2`	Second species.
`output_species`	Species to convert all genes from `species1` and `species2` to first. `Default="human"`, but can be to either any species supported by **orthogene**, including `species1` or `species2`.
`as_output_species`	Return background gene list as `output_species` orthologs, instead of the gene names of the original input species.
`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`.
`bg`	User supplied background list that will be returned to the user after removing duplicate genes.
`gene_map`	User-supplied `gene_map` data table from map_orthologs or map_genes.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`non121_strategy`	How to handle genes that don't have 1:1 mappings between `input_species`:`output_species`. Options include: `"drop_both_species"` or `"dbs"` or `1` Drop genes that have duplicate mappings in either the `input_species` or `output_species` (*DEFAULT*). `"drop_input_species"` or `"dis"` or `2` Only drop genes that have duplicate mappings in the `input_species`. `"drop_output_species"` or `"dos"` or `3` Only drop genes that have duplicate mappings in the `output_species`. `"keep_both_species"` or `"kbs"` or `4` Keep all genes regardless of whether they have duplicate mappings in either species. `"keep_popular"` or `"kp"` or `5` Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.

`"sum"`,`"mean"`,`"median"`,`"min"` or `"max"`	When `gene_df` is a matrix and `gene_output="rownames"`, these options will aggregate many-to-one gene mappings (`input_species-to-output_species`) after dropping any duplicate genes in the `output_species`.
`verbose`	Print messages.

Value

Background gene list.

Examples

```
bg <- orthogene::create_background(species1 = "mouse",
                                   species2 = "rat",
                                   output_species = "human")
```

`dMcast` *dMcast*

Description

Reimplementation of function that originally part of the R package `Matrix.utils` before the package was deprecated. The only difference is that this version of `dMcast` does not include an aggregation feature at the end.

Usage

```
dMcast(
  data,
  formula,
  value.var = NULL,
  as.factors = FALSE,
  na.action = stats::na.pass,
  factor.nas = TRUE,
  drop.unused.levels = TRUE
)
```

Arguments

`data`	A data.frame.
`formula`	Casting formula, see details for specifics.
`value.var`	Name of column that stores values to be aggregated numerics.
`as.factors`	If `TRUE`, treat all columns as factors, including
`factor.nas`	If `TRUE`, treat factors with NAs as new levels. Otherwise, rows with NAs will receive zeroes in all columns for that factor.
`drop.unused.levels`	Should factors have unused levels dropped? Defaults to `TRUE`, in contrast to `model.matrix`

Value

matrix

Source

```
groupings <- data.frame(A = as.factor(sample(1e4,1e6,TRUE))) formula <- stats::as.formula("~0+.")
dm <- orthogene:::dMcast(data = groupings, formula = formula)
```

`earthworm2human_map` *Earthworm to human map*

Description

Orthologous gene mapping between earthworm (Eisenia andrei) and human (Homo sapiens) genes.

Usage

```
earthworm2human_map(evalue_threshold = NULL, save_dir = cache_dir())
```

Arguments

`evalue_threshold`	Only include mappings with an E-value below a set threshold. See here for further guidance.
`save_dir`	Directory to save mapping file to.

Details

These mappings were generated using BLAST (a protein sequence tool) implemented within SAMap. This mapping data was provided upon request by the authors of Wang et al. 2022. Column names were collected from Metagenomics Wiki.

Value

data.table containing earthworm-to-human gene orthologs.

`exp_mouse` *Gene expression data: mouse*

Description

Mean pseudobulk single-cell RNA-seq gene expression matrix.

Data originally comes from Zeisel et al., 2018 (Cell).

Usage

```
data("exp_mouse")
```

Format

sparse matrix

Source

Publication `ctd <- ewceData::ctd() exp_mouse <- as(ctd[[1]]$mean_exp, "sparseMatrix")`
`usethis::use_data(exp_mouse, overwrite = TRUE)`

`exp_mouse_enst` *Transcript expression data: mouse*

Description

Mean pseudobulk single-cell RNA-seq Transcript expression matrix.

Data originally comes from Zeisel et al., 2018 (Cell).

Usage

```
data("exp_mouse_enst")
```

Format

sparse matrix

Source

Publication `data("exp_mouse") mapped_genes <- map_genes(genes = rownames(exp_mouse)[seq(1,100)],`
`target = "ENST", species = "mouse", drop_na = FALSE) exp_mouse_enst <- exp_mouse[mapped_genes$input,]`
`rownames(exp_mouse_enst) <- mapped_genes$target all_nas <- orthogene:::find_all_nas(rownames(exp_mouse`
`exp_mouse_enst <- exp_mouse_enst[!all_nas,] exp_mouse_enst <- phenomix::add_noise(exp_mouse_enst)`
`usethis::use_data(exp_mouse_enst, overwrite = TRUE)`

`format_species` *Format species names*

Description

Format scientific species names into a standardised manner.

Usage

```
format_species(
  species,
  remove_parentheses = TRUE,
  abbrev = FALSE,
  remove_subspecies = FALSE,
  remove_subspecies_exceptions = c("Canis lupus familiaris"),
  split_char = " ",
  collapse = " ",
  remove_chars = c(" ", ".", "(", ")", "[", "]"),
  replace_char = "",
  lowercase = FALSE,
  trim = "'",
  standardise_scientific = FALSE
)
```

Arguments

`species`	Species query (e.g. "human", "homo sapiens", "hsapiens", or 9606). If given a list, will iterate queries for each item. Set to `NULL` to return all species.
`remove_parentheses`	Remove substring within parentheses: e.g. "Xenopus (Silurana) tropicalis" –> "Xenopus tropicalis"
`abbrev`	Abbreviate all taxonomic levels except the last one: e.g. "Canis lupus familiaris" ==> "C l familiaris"
`remove_subspecies`	Only keep the first two taxonomic levels: e.g. "Canis lupus familiaris" –> "Canis lupus"
`remove_subspecies_exceptions`	Selected species to ignore when `remove_subspecies=TRUE`. e.g. "Canis lupus familiaris" –> "Canis lupus familiaris"
`split_char`	Character to split species names by.
`collapse`	Character to re-collapse species names with after splitting with `split_char`.
`remove_chars`	Characters to remove.
`replace_char`	Character to replace `remove_chars` with.
`lowercase`	Make species names all lowercase.

`trim`	Characters to trim from the beginning/end of each species name.
`standardise_scientific`	Automatically sets multiple arguments at once to create standardised scientific names for each species. Assumes that `species` is provided in some version of scientific species names: e.g. "Xenopus (Silurana) tropicalis" –> "Xenopus tropicalis"

Value

A named vector where the values are the standardised species names and the names are the original input species names.

Examples

```
species <- c("Xenopus (Silurana) tropicalis","Canis lupus familiaris")
species2 <- format_species(species = species, abbrev=TRUE)
species3 <- format_species(species = species,
                           standardise_scientific=TRUE,
                           remove_subspecies_exceptions=NULL)
```

`get_cache_save_path` *Get cache save path*

Description

Create a save path for a cached file.

Usage

```
get_cache_save_path(
  fn,
  species,
  method,
  suffix = ".csv.gz",
  save_dir = cache_dir()
)
```

Arguments

`fn`	Function name.
`species`	input species name.
`method`	Method (gprofiler2, homologene, or babelgene).
`suffix`	Cache file suffix.
`save_dir`	Cache file directory.

Value

Path string

`get_orgdb_genomeinfodbdata`
Import organism database: GenomeInfoDbData

Description

Import and format organism ID table from **GenomeInfoDbData** to be comparable to `get_orgdb_gprofiler`.

Usage

```
get_orgdb_genomeinfodbdata(verbose = TRUE)
```

Value

Organisms `data.table`

Source

GenomeInfoDbData GitHub

`get_silhouettes` *Get silhouettes*

Description

Get silhouette images of each species from phylopic.

Usage

```
get_silhouettes(
  species,
  which = rep(1, length(species)),
  run_format_species = TRUE,
  include_image_data = FALSE,
  mc.cores = 1,
  add_png = FALSE,
  remove_bg = FALSE,
  verbose = TRUE
)
```

Arguments

`species`	A character vector of species names to query phylopic for.
`which`	An integer vector of the same length as `species`. Lets you choose which image you want to use for each species (1st, 2nd 3rd, etc.).
`run_format_species`	Standardise species names with format_species before querying phylopic (default: `TRUE`).
`include_image_data`	Include the image data itself (not just the image UID) in the results.
`mc.cores`	Accelerate multiple species queries by parallelising across multiple cores.
`add_png`	Return URLs for both the SVG and PNG versions of the image.
`remove_bg`	Remove image background.
`verbose`	Print messages.

Value

data.frame with:

input_species : Species name (input).

species : Species name (standardised).

uid : Species UID.

url : Image URL.

Source

Related function: `ggimage::geom_phylopic`

phylopic/rphylopic API changes

ggimage: Issue with finding valid PNGs

Examples

```
species <- c("Mus_musculus","Pan_troglodytes","Homo_sapiens")
uids <- get_silhouettes(species = species)
```

`ggtree_plot` *Plot a phylogenetic tree*

Description

Plot a phylogenetic tree with **ggtree** and metadata from report_orthologs.

Usage

```
ggtree_plot(
  tr,
  d,
  scaling_factor = 1,
  clades = NULL,
  clades_palette = NULL,
  reference_species = NULL,
  verbose = TRUE
)
```

Arguments

`tr`	Tree.
`d`	Metadata
`scaling_factor`	How much to scale y-axis parameters (e.g. offset) by.
`clades`	Clades metadata.
`clades_palette`	Palette to color highlighted `clades` with.
`verbose`	Print messages.

Value

ggplot object.

`gprofiler_namespace` *gconvert namespaces*

Description

Available namespaces used by gconvert.

Format

`data.frame`

Value

`data.frame`

Source

gProfiler site

`#### Manually-prepared CSV #### path <- "inst/extdata/gprofiler_namespace.csv.gz" gprofiler_namespace <- data.table::fread(path)`

`gprofiler_orgs` *Reference organisms*

Description

Organism for which gene references are available via gProfiler API. Used as a backup if API is not available.

Format

`data.frame`

Value

`data.frame`

Source

gProfiler site

`# NOTE!: Must run usethis::use_data for all internal data at once. # otherwise, the prior internal data will be overwritten. #### Internal data 1: gprofiler_namespace #### #### Manually-prepared CSV #### path <- "inst/extdata/gprofiler_namespace.csv.gz" gprofiler_namespace <- data.table::fread(path) #### Internal data 2: gprofiler_orgs gprofiler_orgs <- orthogene:::get_orgdb_ #### Save #### usethis::use_data(gprofiler_orgs,gprofiler_namespace, overwrite = TRUE, internal=TRUE)`

`infer_species` *Infer species from gene names*

Description

Infers which species the genes within `gene_df` is from. Iteratively test the percentage of `gene_df` genes that match with the genes from each `test_species`.

Usage

```
infer_species(
  gene_df,
  gene_input = "rownames",
  test_species = c("human", "monkey", "rat", "mouse", "zebrafish", "fly"),
  method = c("homologene", "gprofiler", "babelgene"),
  make_plot = TRUE,
  show_plot = TRUE,
  verbose = TRUE
)
```

Arguments

`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`. `list` 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"`.
`test_species`	Which species to test for matches with. If set to `NULL`, will default to a list of humans and 5 common model organisms. If `test_species` is set to one of the following options, it will automatically pull all species from that respective package and test against each of them: **"homologene"** 20+ species (default) **"gprofiler"** 700+ species **"babelgene"** 19 species
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`make_plot`	Make a plot of the results.
`show_plot`	Print the plot of the results.
`verbose`	Print messages.

Value

An ordered dataframe of `test_species` from best to worst matches.

Examples

```
data("exp_mouse")
matches <- orthogene::infer_species(gene_df = exp_mouse[1:200,])
```

`infer_species_plot` *infer_species_plot*

Description

Plot results from infer_species.

Usage

```
infer_species_plot(matches, show_plot = TRUE)
```

Value

ggplot object.

`invert_dictionary` *Invert dictionary*

Description

Switch the names/items in a named list.

Usage

```
invert_dictionary(dict)
```

Value

Named list

`many2many_rows` *Expand/aggregate rows of matrix for many:many mappings*

Description

Expand/aggregate rows of a matrix with any combination of many:many mappings. This method ensures that total counts per gene remain the same regardless of how many genes it has split/condensed into. This allows for many:many mappings that are otherwise not possible using standard aggregation functions, since they all require many:1 scenarios.

Internally, this is done as follows:

- Identify genes that appear more than once in `gene_map[[input_col]]`.

- For each gene identified, split its row into multiple rows, where the number of new rows is equal to the number of times that gene appears within `gene_map[[input_col]]`. In the new expanded matrix, each row will be equal to the column sums divided by the number of new rows. This means that averaged counts will be split equally amongst the new rows, in a column-specific manner.

  Thus, the column sums of the output matrix will be equal to the column sums in the input matrix. In the case of gene expression count matrices, this means that the total counts will remain equal between matrices, while avoiding being forced to drop genes with many:many mappings (as is the case with most other aggregation methods).
- Map rownames of the expanded matrix onto the orthologous gene names from `gene_map$ortholog_gene`.
- [Optional] When `aggregate_orthologs=TRUE`, aggregate rows of the expanded/mapped matrix such that there will only be 1 row per ortholog gene, using aggregate_rows. The arguments `FUN`, `method`, `as_sparse`, `as_DelayedArray`, and `dropNA` will all be passed to aggregate_rows if this step is selected.

Usage

```
many2many_rows(
  X,
  gene_map,
  input_col = "input_gene",
  output_col = "ortholog_gene",
  agg_fun = "sum",
  agg_method = c("monocle3", "stats"),
  as_sparse = TRUE,
  as_DelayedArray = FALSE,
  dropNA = TRUE,
  aggregate_orthologs = TRUE,
  verbose = TRUE
)
```

Arguments

`X`	Input matrix.
`gene_map`	A data.frame generated by map_orthologs, with columns mapping `input_col` to `output_col`.
`input_col`	Column name within `gene_map` with gene names matching the row names of `X`.
`output_col`	Column name within `gene_map` with gene names that you wish you map the row names of `X` onto.
`agg_fun`	Aggregation function.
`agg_method`	Aggregation method.
`as_sparse`	Convert aggregated matrix to sparse matrix.
`as_DelayedArray`	Convert aggregated matrix to DelayedArray.
`dropNA`	Drop genes assigned to `NA` in `groupings`.

`aggregate_orthologs`	[Optional] After performing an initial round of many:many aggregation/expansion with many2many_rows, ensure each orthologous gene only appears in one row by using the aggregate_rows function (default: `TRUE`).
`verbose`	Print messages.

Value

Expanded/aggregated matrix.

Source

`data("exp_mouse") X <- exp_mouse gene_map <- orthogene:::map_orthologs(genes = rownames(exp_mouse), input_species = "mouse", method="homologene") X_agg <- orthogene:::many2many_rows(X = X, gene_map = gene_map) sum(duplicated(rownames(exp_mouse))) # 0 sum(duplicated(gene_map$input_gene)) # 46 sum(duplicated(gene_map$ortholog_gene)) # 56 sum(duplicated(rownames(X_agg))) # 56`

`map_genes` *Map genes*

Description

Input a list of genes, transcripts, proteins, SNPs, or genomic ranges in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and return a table with standardised gene symbols (the "names" column).

Usage

```
map_genes(
  genes,
  species = "hsapiens",
  target = "ENSG",
  mthreshold = Inf,
  drop_na = FALSE,
  numeric_ns = "",
  run_map_species = TRUE,
  verbose = TRUE
)
```

Arguments

`genes`	Gene list.
`species`	Species to map against.
`target`	target namespace.
`mthreshold`	maximum number of results per initial alias to show. Shows all by default.
`drop_na`	Drop all genes without mappings. Sets `gprofiler2::gconvert(filter_na=)` as well an additional round of more comprehensive `NA` filtering by **orthogene**.

`numeric_ns`	namespace to use for fully numeric IDs (list of available namespaces).
`run_map_species`	Standardise `species` names with map_species first (Default: `TRUE`).
`verbose`	Print messages.

Details

Uses gconvert. The exact contents of the output table will depend on `target` parameter. See `?gprofiler2::gconvert` for more details.

Value

Table with standardised genes.

Examples

```
genes <- c(
    "Klf4", "Sox2", "TSPAN12", "NM_173007", "Q8BKT6",
    "ENSMUSG00000012396", "ENSMUSG00000074637"
)
mapped_genes <- map_genes(
    genes = genes,
    species = "mouse"
)
```

`map_genes_planosphere` *Map genes: SMED*

Description

Map planarian (Schmmidt mediterrani) genes to/from the SMED format using data from the planosphere database.

Usage

```
map_genes_planosphere(
  genes,
  output_format = "SMESG_dd_Smes_v2",
  drop_duplicates = TRUE,
  save_dir = cache_dir(),
  verbose = TRUE
)
```

Arguments

`genes`	Gene list.
`drop_duplicates`	Only output one row per input gene.
`verbose`	Print messages.

Value

data.table

Source

`genes <- c("dd_Smed_v6_10690_0","dd_Smed_v6_10691_0","dd_Smed_v6_10693_0") gene_map <- map_genes_planosphere(genes=genes)`

`map_orthologs` *Map orthologs*

Description

Map orthologs from one species to another.

Usage

```
map_orthologs(
  genes,
  standardise_genes = FALSE,
  input_species,
  output_species = "human",
  method = c("gprofiler", "homologene", "babelgene"),
  mthreshold = Inf,
  gene_map = NULL,
  input_col = "input_gene",
  output_col = "ortholog_gene",
  verbose = TRUE,
  ...
)
```

Arguments

`genes`	can be a mixture of any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to standardised HGNC symbol format.
`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.
`input_species`	Name of the input species (e.g., "mouse","fly"). Use map_species to return a full list of available species.
`output_species`	Name of the output species (e.g. "human","chicken"). Use map_species to return a full list of available species.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes.

	`"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`mthreshold`	Maximum number of ortholog names per gene to show. Passed to gorth. Only used when `method="gprofiler"` (*DEFAULT :* `Inf`).
`gene_map`	A data.frame that maps the current gene names to new gene names. This function's behaviour will adapt to different situations as follows: `gene_map=<data.frame>` When a data.frame containing the gene key:value columns (specified by `input_col` and `output_col`, respectively) is provided, this will be used to perform aggregation/expansion. `gene_map=NULL` **and** `input_species!=output_species` A `gene_map` is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion. `gene_map=NULL` **and** `input_species==output_species` A `gene_map` is automatically generated by map_genes to perform within-species gene gene symbol standardization and aggregation/expansion.
`input_col`	Column name within `gene_map` with gene names matching the row names of `X`.
`output_col`	Column name within `gene_map` with gene names that you wish you map the row names of `X` onto.
`verbose`	Print messages.
`...`	Additional arguments to be passed to gorth or homologene. *NOTE*: To return only the most "popular" interspecies ortholog mappings, supply `mthreshold=1` here AND set `method="gprofiler"` above. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs. For more details, please see here.

Details

`map_orthologs()` is a core function within `convert_orthologs()`, but does not have many of the extra checks, such as `non121_strategy`) and `drop_nonorths`.

Value

Ortholog map `data.frame` with at least the columns "input_gene" and "ortholog_gene".

Examples

```
data("exp_mouse")
gene_map <- map_orthologs(
    genes = rownames(exp_mouse),
    input_species = "mouse")
```

`map_orthologs_babelgene`

Map orthologs: babelgene

Description

Map orthologs from one species to another using orthologs.

Usage

```
map_orthologs_babelgene(
  genes,
  input_species,
  output_species = "human",
  min_support = 1,
  top = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`genes`	Gene list.
`input_species`	Name of the input species (e.g., "mouse","fly"). Use map_species to return a full list of available species.
`output_species`	Name of the output species (e.g. "human","chicken"). Use map_species to return a full list of available species.
`min_support`	Minimum number of supporting source databases. Gene pairs available in this package are supported by 2 to 12 databases (the maximum varies depending on the species).
`top`	For each gene, output only the match with the highest support level if there are multiple hits.
`verbose`	Print messages.
`...`	Additional arguments to be passed to gorth or homologene. *NOTE*: To return only the most "popular" interspecies ortholog mappings, supply `mthreshold=1` here AND set `method="gprofiler"` above. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs. For more details, please see here.

Value

Ortholog map `data.frame`

Source

babelgene tutorial

`map_orthologs_custom` *Map orthologs: gprofiler*

Description

Map orthologs from one species to another using a custom `gene_map` table.

Usage

```
map_orthologs_custom(
  gene_map,
  input_species,
  output_species,
  input_col,
  output_col,
  verbose = TRUE
)
```

Arguments

`gene_map`	A data.frame that maps the current gene names to new gene names. This function's behaviour will adapt to different situations as follows: `gene_map=<data.frame>` When a data.frame containing the gene key:value columns (specified by `input_col` and `output_col`, respectively) is provided, this will be used to perform aggregation/expansion. `gene_map=NULL` **and** `input_species!=output_species` A `gene_map` is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion. `gene_map=NULL` **and** `input_species==output_species` A `gene_map` is automatically generated by map_genes to perform within-species gene gene symbol standardization and aggregation/expansion.
`input_species`	Name of the input species (e.g., "mouse","fly"). Use map_species to return a full list of available species.
`output_species`	Name of the output species (e.g. "human","chicken"). Use map_species to return a full list of available species.
`input_col`	Column name within `gene_map` with gene names matching the row names of `X`.
`output_col`	Column name within `gene_map` with gene names that you wish you map the row names of `X` onto.
`verbose`	Print messages.

Value

Ortholog map `data.frame`

`map_orthologs_gprofiler`

Map orthologs: gprofiler

Description

Map orthologs from one species to another using gorth.

Usage

```
map_orthologs_gprofiler(
  genes,
  input_species,
  output_species = "human",
  filter_na = FALSE,
  mthreshold = Inf,
  verbose = TRUE,
  ...,
  chunked = TRUE,
  chunk_size = 20000,
  retry = 3,
  retry_backoff = 1,
  n_cores = 1
)
```

Arguments

`genes`	Gene list.
`input_species`	Input species.
`output_species`	Output species. Default "human".
`filter_na`	Logical indicating whether to filter out results without a corresponding target name.
`mthreshold`	Maximum number of ortholog names per gene to show.
`verbose`	Logical; print progress.
`...`	Additional arguments passed to gorth.
`chunked`	Logical; if TRUE, query g:Profiler in chunks and rbind results.
`chunk_size`	Integer; number of genes per chunk when chunked=TRUE.
`retry`	Integer; number of attempts per chunk (>=1). Only used when chunked=TRUE.
`retry_backoff`	Numeric; base backoff (seconds). Sleep = retry_backoff * attempt.

Value

Ortholog map `data.frame`

`map_orthologs_homologene` *Map orthologs: homologene*

Description

Map orthologs from one species to another using homologene.

Usage

```
map_orthologs_homologene(
  genes,
  input_species,
  output_species = "human",
  verbose = TRUE,
  ...
)
```

Arguments

`genes`	Gene list.
`input_species`	Name of the input species (e.g., "mouse","fly"). Use map_species to return a full list of available species.
`output_species`	Name of the output species (e.g. "human","chicken"). Use map_species to return a full list of available species.
`verbose`	Print messages.
`...`	Additional arguments to be passed to homologene.

Value

Ortholog map `data.frame`

`map_species` *Standardise species names*

Description

Search `gprofiler` database for species that match the input text string. Then translate to a standardised species ID.

Usage

```
map_species(
  species = NULL,
  search_cols = c("display_name", "id", "scientific_name", "taxonomy_id"),
 output_format = c("scientific_name", "id", "display_name", "taxonomy_id", "version",
    "scientific_name_formatted"),
  method = c("homologene", "gprofiler", "babelgene"),
  remove_subspecies = TRUE,
  remove_subspecies_exceptions = c("Canis lupus familiaris"),
  use_local = TRUE,
  verbose = TRUE
)
```

Arguments

`species`	Species query (e.g. "human", "homo sapiens", "hsapiens", or 9606). If given a list, will iterate queries for each item. Set to `NULL` to return all species.
`search_cols`	Which columns to search for `species` substring in metadata API.
`output_format`	Which column to return.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`remove_subspecies`	Only keep the first two taxonomic levels: e.g. "Canis lupus familiaris" –> "Canis lupus"
`remove_subspecies_exceptions`	Selected species to ignore when `remove_subspecies=TRUE`. e.g. "Canis lupus familiaris" –> "Canis lupus familiaris"
`use_local`	If `TRUE` *default*, map_species uses a locally stored version of the species metadata table instead of pulling directly from the gprofiler API. Local version may not be fully up to date, but should suffice for most use cases.
`verbose`	Print messages.

Value

Species ID of type `output_format`

Examples

```
ids <- map_species(species = c(
    "human", 9606, "mus musculus",
    "fly", "C elegans"
))
```

`message_parallel` *Send messages to console even from within parallel processes*

Description

Send messages to console even from within parallel processes

Usage

```
message_parallel(...)
```

Value

A message

`plot_benchmark_bar` *Plot benchmark: bar*

Description

Plot run time and # genes returned across species and function tests.

Usage

```
plot_benchmark_bar(
  bench_res,
  remove_failed_times = TRUE,
  point_size = 2,
  point_mult = 1,
  show_plot = TRUE
)
```

Arguments

`bench_res`	Results from
`remove_failed_times`	In instances where no genes were returned, set time to `NA`.
`point_size`	Size of the shape above each bar.
`point_mult`	Multiplier to place the point above the bar.
`show_plot`	Print plot.

Value

ggplot object

`plot_benchmark_scatter` *Plot benchmark: scatter*

Description

Plot run time vs. # genes returned across species and function tests.

Usage

```
plot_benchmark_scatter(
  bench_res,
  remove_failed_times = FALSE,
  show_plot = TRUE
)
```

Arguments

`bench_res`	Results from
`remove_failed_times`	In instances where no genes were returned, set time to `NA`.
`show_plot`	Print plot.

Value

ggplot object

`plot_orthotree` *Create a phylogenetic tree of shared orthologs*

Description

Automatically creates a phylogenetic tree plot annotated with metadata describing how many orthologous genes each species shares with the `reference_species` ("human" by default).

Usage

```
plot_orthotree(
  tree = NULL,
  orth_report = NULL,
  species = NULL,
  method = c("babelgene", "homologene", "gprofiler"),
  tree_source = "timetree",
  non121_strategy = "drop_both_species",
  reference_species = "human",
```

```
    clades = list(Primates = c("Homo sapiens", "Macaca mulatta"), Eutherians =
      c("Homo sapiens", "Mus musculus", "Bos taurus"), Mammals = c("Homo sapiens",
     "Mus musculus", "Bos taurus", "Ornithorhynchus anatinus", "Monodelphis domestica"),
     Tetrapods = c("Homo sapiens", "Mus musculus", "Gallus gallus", "Anolis carolinensis",
       "Xenopus tropicalis"), Vertebrates = c("Homo sapiens", "Mus musculus",
      "Gallus gallus", "Anolis carolinensis", "Xenopus tropicalis", "Danio rerio"),
       Invertebrates = c("Drosophila melanogaster",
        "Caenorhabditis elegans")),
    clades_rotate = list(),
    scaling_factor = NULL,
    show_plot = TRUE,
   save_paths = c(tempfile(fileext = ".ggtree.pdf"), tempfile(fileext = ".ggtree.png")),
    width = 15,
    height = width,
    mc.cores = 1,
    verbose = TRUE
  )
```

Arguments

`tree`	A phylogenetic tree of class phylo. If no tree is provided (`NULL`) a 100-way multiz tree will be imported from UCSC Genome Browser.
`orth_report`	An ortholog report from one or more species generated by report_orthologs.
`species`	Species to include in the final plot. If `NULL`, then all species from the given database (`method`) will be included (via map_species), so long as they also exist in the `tree`.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`tree_source`	Can be one of the following: **"timetree2022":** Import and prune the TimeTree >147k species phylogenetic tree. Can also simply type "timetree". **"timetree2015":** Import and prune the TimeTree >50k species phylogenetic tree. **"OmaDB":** Construct a tree from OMA (Orthologous Matrix browser) via the getTaxonomy function. *NOTE:* Does not contain branch lengths, and therefore may have limited utility. **"UCSC":** Import and prune the UCSC 100-way alignment phylogenetic tree (hg38 version). **"<path>":** Read a tree from a newick text file from a local or remote URL using read.tree.
`non121_strategy`	How to handle genes that don't have 1:1 mappings between `input_species`:`output_species`. Options include:

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

`reference_species`	Reference species.
`clades`	A named list of clades each containing a character vector of species used to define the respective clade using MRCA.
`clades_rotate`	A list of clades to rotate (via rotate), each containing a character vector of species used to define the respective clade using MRCA.
`scaling_factor`	How much to scale y-axis parameters (e.g. offset) by.
`show_plot`	Whether to print the final tree plot.
`save_paths`	Paths to save plot to.
`width`	Saved plot width.
`height`	Saved plot height.
`mc.cores`	Number of cores to parallelise different steps with.
`verbose`	Print messages.

Value

A list containing:

plot : Annotated ggtree object.

tree : The pruned, standardised phylogenetic tree used in the plot.

orth_report : Ortholog reports for each `species` against the `reference_species`.

metadata : Metadata used in the plot, including silhouette PNG ids from phylopic.

clades : Metadata used for highlighting clades.

method : `method` used.

reference_species : `reference_species` used.

save_paths : `save_paths` to plot.

Source

ggtree tutorial

Examples

```
if(require("ape")){
 suppressWarnings(
  orthotree <- plot_orthotree(species = c("human","monkey","mouse"))
 )
}
```

`prepare_tree` *Prepare a phylogenetic tree*

Description

Import a phylogenetic tree and then conduct a series of optional standardisation steps. Optionally, if `output_format` is not `NULL`, species names from both the tree and the `species` argument will first be standardised using map_species.

Usage

```
prepare_tree(
  tree_source = "timetree",
  species = NULL,
  output_format = "scientific_name_formatted",
  run_map_species = c(TRUE, TRUE),
  method = c("homologene", "gprofiler", "babelgene"),
  force_ultrametric = TRUE,
  age_max = NULL,
  show_plot = TRUE,
  save_dir = cache_dir(),
  verbose = TRUE,
  ...
)
```

Arguments

`tree_source`	Can be one of the following: **"timetree2022":** Import and prune the TimeTree >147k species phylogenetic tree. Can also simply type "timetree". **"timetree2015":** Import and prune the TimeTree >50k species phylogenetic tree. **"OmaDB":** Construct a tree from OMA (Orthologous Matrix browser) via the getTaxonomy function. *NOTE:* Does not contain branch lengths, and therefore may have limited utility.

	"UCSC": Import and prune the UCSC 100-way alignment phylogenetic tree (hg38 version). **"<path>":** Read a tree from a newick text file from a local or remote URL using read.tree.
`species`	Species names to subset the tree by (after `standardise_species` step).
`output_format`	Which column to return.
`run_map_species`	Whether to first standardise species names with map_species.
`method`	R package to use for gene mapping: `"gprofiler"` Slower but more species and genes. `"homologene"` Faster but fewer species and genes. `"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.
`force_ultrametric`	Whether to force the tree to be ultrametric (i.e. make all tips the same date) using force.ultrametric.
`age_max`	Rescale the edges of the tree into units of millions of years (MY) instead than evolutionary rates (e.g. dN/dS ratios). Only used if `age_max`, the max number , is numeric. Times are computed using makeChronosCalib and chronos.
`show_plot`	Show a basic plot of the resulting tree.
`save_dir`	Directory to cache full tree in. Set to `NULL` to avoid using cache.
`verbose`	Print messages.
`...`	Additional arguments passed to makeChronosCalib.

Value

A filtered tree of class "phylo" (with standardised species names).

Source

TimeTree 5: An Expanded Resource for Species Divergence Times

Examples

```
if(require("ape")){
 species <- c("human","chimp","mouse")
 tr <- orthogene::prepare_tree(species = species)
}
```

`remove_image_bg` *Remove image background*

Description

Import and image and remove the background using **magick**.

Usage

```
remove_image_bg(
  path,
  color = "white",
  fuzz = 0,
 save_path = file.path(tempdir(), "phylopic_processed", paste0(basename(dirname(path)),
    ".png"))
)
```

Arguments

`path`	a file, url, or raster object or bitmap array
`color`	a valid color string such as `"navyblue"` or `"#000080"`. Use `"none"` for transparency.
`fuzz`	relative color distance (value between 0 and 100) to be considered similar in the filling algorithm

Value

Named list containing the modified image itself and the saved path of the modified image.

Source

```
 path <- paste0("https://images.phylopic.org/images/","2de1c95c-7e1f-429b-9c08-17f0a27d176f/vector.sv
img_res <- remove_image_bg(path=path)
```

`report_orthologs` *Report orthologs*

Description

Identify the number of orthologous genes between two species.

Usage

```
report_orthologs(
  target_species = "mouse",
  reference_species = "human",
  standardise_genes = FALSE,
  method_all_genes = c("homologene", "gprofiler", "babelgene"),
  method_convert_orthologs = method_all_genes,
  drop_nonorths = TRUE,
  non121_strategy = "drop_both_species",
  round_digits = 2,
  return_report = TRUE,
  ref_genes = NULL,
  mc.cores = 1,
  verbose = TRUE,
  ...
)
```

Arguments

`target_species`	Target species.
`reference_species`	Reference species.
`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.
`method_all_genes`	Package to use in the all_genes step: `"gprofiler"` Slower, but covers more species and genes. `"homologene"` Faster, but covers fewer species and genes. `"babelgene"` Faster, fewer species/genes; also provides consensus scores for each mapping from multiple data sources.
`method_convert_orthologs`	Package to use in the convert_orthologs step: `"gprofiler"` Slower, but covers more species and genes. `"homologene"` Faster, but covers fewer species and genes. `"babelgene"` Faster, fewer species/genes; also provides consensus scores for each mapping from multiple data sources.
`drop_nonorths`	Drop genes that don't have an ortholog in the `output_species`.
`non121_strategy`	How to handle genes that don't have 1:1 mappings between `input_species:output_species`. Options include: `"drop_both_species"` or `"dbs"` or `1` Drop genes that have duplicate mappings in either the `input_species` or `output_species` (*DEFAULT*).

	`"drop_input_species"` or `"dis"` or `2` Only drop genes that have duplicate mappings in the `input_species`. `"drop_output_species"` or `"dos"` or `3` Only drop genes that have duplicate mappings in the `output_species`. `"keep_both_species"` or `"kbs"` or `4` Keep all genes regardless of whether they have duplicate mappings in either species. `"keep_popular"` or `"kp"` or `5` Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs. `"sum"`,`"mean"`,`"median"`,`"min"` or `"max"` When `gene_df` is a matrix and `gene_output="rownames"`, these options will aggregate many-to-one gene mappings (`input_species`-to-`output_species`) after dropping any duplicate genes in the `output_species`.
`round_digits`	Number of digits to round to when printing percentages.
`return_report`	If `FALSE`, return only the ortholog mapping between the two species. If `TRUE`, return both the ortholog mapping and a `data.frame` of report statistics.
`ref_genes`	A table of all genes for `reference_species`. If `NULL` (default), this is created via all_genes.
`mc.cores`	Number of cores to parallelize across `target_species`.
`verbose`	Print messages.
`...`	Arguments passed on to `convert_orthologs` `gene_df` Data object containing the genes (see `gene_input` for options on how the genes can be stored within the object). Can be one of the following formats: `matrix` A sparse or dense matrix. `data.frame` A `data.frame`, `data.table`. or `tibble`. `list` 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.

`input_species` Name of the input species (e.g., "mouse","fly"). Use map_species to return a full list of available species.

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

`method` R package to use for gene mapping:

`"gprofiler"` Slower but more species and genes.

`"homologene"` Faster but fewer species and genes.

`"babelgene"` Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.

`as_sparse` Convert `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"`).

`sort_rows` Sort `gene_df` rows alphanumerically.

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

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

`gene_map=NULL` **and** `input_species!=output_species` A `gene_map` is automatically generated by map_orthologs to perform inter-species gene aggregation/expansion.

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

`as_DelayedArray` Convert aggregated matrix to DelayedArray.

`input_col` Column name within `gene_map` with gene names matching the row names of `X`.

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

Value

A list containing:

map A table of inter-species gene mappings.

report A list of aggregate orthology report statistics.

If more than one `target_species` is provided, the function returns a table of aggregated `report` statistics concatenated across species.

Examples

```
orth_fly <- report_orthologs(
  target_species = "fly",
  reference_species = "human"
)
```

`run_benchmark` *Run benchmark tests*

Description

Runs benchmarks tests on all_genes and convert_orthologs across multiple species, using multiple methods ("homologene", and "gprofiler").

Usage

```
run_benchmark(
  species,
  method_list = c("homologene", "gprofiler", "babelgene"),
  run_convert_orthologs = TRUE,
  remove_failed_times = FALSE,
  save_path = tempfile(fileext = ".csv"),
  mc.cores = 1,
  force = FALSE,
  verbose = TRUE
)
```

Arguments

`species`	Species names.
`method_list`	A list of ortholog conversion methods to benchmark.
`run_convert_orthologs`	Benchmark convert_orthologs function.
`remove_failed_times`	In instances where no genes were returned, set time to `NA`.
`save_path`	Path to save results to.

`mc.cores`	Number of cores to parallelise species across.
`force`	Force regeneration of files even if cached files exist. Set to TRUE or 1 to just force regenerating the `convert_orthologs` step. Set to 2 to force regenerating both the `convert_orthologs` and `all_genes` steps.
`verbose`	Print messages.

Value

`data.table` with benchmark results

`taxa_id_dict` *Taxa ID dictionary*

Description

Dictionary of NCBI taxonomy IDs mapped to Latin and common names of 20+ organisms.

Usage

```
taxa_id_dict(
  species = c("human", "chimp", "monkey", "mouse", "rat", "dog", "cow", "chicken",
    "zebrafish", "frog", "fly", "worm", "rice"),
   include_common_names = TRUE
)
```

Arguments

`species`	Species to get dictionary for. Can supply either Latin names (e.g. "Homo sapiens") or common names (e.g, "human").

Value

Named list of taxa IDs to organism names.

Index