

# Package ‘orthogene’

April 8, 2025

**Type** Package

**Title** Interspecies gene mapping

**Version** 1.13.0

**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 (>= 4.1)

**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, grr, repmis, ggtree, tools

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

**RoxygenNote** 7.2.3

**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** cd1c983

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2025-04-07

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

**Maintainer** Brian Schilder <brian\_schilder@alumni.brown.edu>

## Contents

orthogene-package . . . . .	3
add_synonyms . . . . .	4
aggregate_mapped_genes . . . . .	4
aggregate_rows . . . . .	6
aggregate_rows_monocle3 . . . . .	7
all_genes . . . . .	8
all_genes_babelgene . . . . .	9
all_species . . . . .	10
check_gene_df_type . . . . .	11
convert_orthologs . . . . .	12
create_background . . . . .	15
dMcast . . . . .	17
earthworm2human_map . . . . .	18
exp_mouse . . . . .	19
exp_mouse_enst . . . . .	19
format_species . . . . .	20
get_orgdb_genomeinfodbdata . . . . .	21
get_silhouettes . . . . .	22
ggtree_plot . . . . .	23
gprofiler_namespace . . . . .	24
gprofiler_orgs . . . . .	24
infer_species . . . . .	25
infer_species_plot . . . . .	26
invert_dictionary . . . . .	27
many2many_rows . . . . .	27
map_genes . . . . .	29
map_genes_planosphere . . . . .	30
map_orthologs . . . . .	31
map_orthologs_babelgene . . . . .	32
map_orthologs_custom . . . . .	34
map_orthologs_gprofiler . . . . .	35
map_orthologs_homologene . . . . .	36
map_species . . . . .	36
message_parallel . . . . .	38
plot_benchmark_bar . . . . .	38
plot_benchmark_scatter . . . . .	39
plot_orthotree . . . . .	39
prepare_tree . . . . .	42

<i>orthogene-package</i>	3
remove_image_bg . . . . .	44
report_orthologs . . . . .	44
run_benchmark . . . . .	48
set_gprofiler . . . . .	49
taxa_id_dict . . . . .	50
<b>Index</b>	<b>51</b>

---

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](#))

## 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	<i>Add gene synonyms</i>
--------------	--------------------------

---

**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 <a href="#">data.frame</a> that maps the current gene names to new gene names. This function's behaviour will adapt to different situations as follows: <ul style="list-style-type: none"> <li>• <code>gene_map=&lt;data.frame&gt;</code> : When a <code>data.frame</code> containing the gene key:value columns (specified by <code>input_col</code> and <code>output_col</code>, respectively) is provided, this will be used to perform aggregation/expansion.</li> <li>• <code>gene_map=NULL</code> and <code>input_species!=output_species</code> : A <code>gene_map</code> is automatically generated by <a href="#">map_orthologs</a> to perform inter-species gene aggregation/expansion.</li> <li>• <code>gene_map=NULL</code> and <code>input_species==output_species</code> : A <code>gene_map</code> is automatically generated by <a href="#">map_genes</a> to perform within-species gene symbol standardization and aggregation/expansion.</li> </ul>
input_col	Column name within <code>gene_map</code> with gene names matching the row names of X.
output_col	Column name within <code>gene_map</code> 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 <a href="#">map_species</a> to return a full list of available species.
output_species	Name of the output species (e.g. "human","chicken"). Use <a href="#">map_species</a> to return a full list of available species.
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
agg_fun	Aggregation function.
agg_method	Aggregation method.

aggregate_orthologs	[Optional] After performing an initial round of many:many aggregation/expansion with <code>many2many_rows</code> , ensure each orthologous gene only appears in one row by using the <code>aggregate_rows</code> function (default: TRUE).
transpose	Transpose <code>gene_df</code> before mapping genes.
mtreehreshold	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 ( <a href="#">list of available namespaces</a> ).
as_integers	Force all values in the matrix to become integers, by applying <code>floor</code> (default: FALSE).
as_sparse	Convert aggregated matrix to sparse matrix.
as_DelayedArray	Convert aggregated matrix to <code>DelayedArray</code> .
dropNA	Drop genes assigned to NA in groupings.
sort_rows	Sort <code>gene_df</code> 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	<i>Aggregate rows of matrix</i>
----------------	---------------------------------

---

**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 <a href="#">DelayedArray</a> .
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,
  verbose = TRUE,
  ...
)
```



**Arguments**

species	Species to get all genes for. Will first be standardised with <code>map_species</code> .
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
ensure_filter_nas	Perform an extra check to remove genes that are NAs of any kind.
run_map_species	Standardise species names with <code>map_species</code> first (Default: TRUE).
verbose	Print messages.
...	Additional arguments to be passed to <code>gorth</code> or <code>homologene</code> .

*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 = tools::R_user_dir("orthogene", which = "cache"),
  use_old = FALSE,
  min_support = 1,
  verbose = TRUE
)
```

**Arguments**

species	Species to get all genes for. Will first be standardised with <code>map_species</code> .
run_map_species	Standardise species names with <code>map_species</code> first (Default: TRUE).
save_dir	Directory to save babelgene mapping files to.
use_old	Use an old version of <code>babelgene::orthologs_df</code> (stored on GitHub Releases) for consistency.
verbose	Print messages.

**Value**

All genes.

**Source**

[babelgene::orthologs\\_df version differences](#)

---

all_species	<i>All species</i>
-------------	--------------------

---

**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: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
verbose	Print messages.

**Value**

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

**Examples**

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

---

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**

<code>gene_df</code>	<p>Data object containing the genes (see <code>gene_input</code> for options on how the genes can be stored within the object). Can be one of the following formats:</p> <ul style="list-style-type: none"> <li>• <code>matrix</code> : A sparse or dense matrix.</li> <li>• <code>data.frame</code> : A <code>data.frame</code>, <code>data.table</code>. or <code>tibble</code>.</li> <li>• <code>codelist</code> : A list or character vector.</li> </ul> <p>Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the <code>...</code> arguments. <i>Note:</i> If you set <code>method="homologene"</code>, you must either supply genes in gene symbol format (e.g. "Sox2") OR set <code>standardise_genes=TRUE</code>.</p>
<code>gene_input</code>	<p>Which aspect of <code>gene_df</code> to get gene names from:</p> <ul style="list-style-type: none"> <li>• <code>"rownames"</code> : From row names of <code>data.frame/matrix</code>.</li> <li>• <code>"colnames"</code> : From column names of <code>data.frame/matrix</code>.</li> <li>• <code>&lt;column name&gt;</code> : From a column in <code>gene_df</code>, e.g. <code>"gene_names"</code>.</li> </ul>
<code>verbose</code>	<p>Print messages.</p>

**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.

	<ul style="list-style-type: none"> <li>• <code>codelist</code> : A list or character vector.</li> </ul> <p>Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the <code>...</code> arguments. <i>Note:</i> If you set <code>method="homologene"</code>, you must either supply genes in gene symbol format (e.g. "Sox2") OR set <code>standardise_genes=TRUE</code>.</p>
<code>gene_input</code>	<p>Which aspect of <code>gene_df</code> to get gene names from:</p> <ul style="list-style-type: none"> <li>• <code>"rownames"</code> : From row names of data.frame/matrix.</li> <li>• <code>"colnames"</code> : From column names of data.frame/matrix.</li> <li>• <code>&lt;column name&gt;</code> : From a column in <code>gene_df</code>, e.g. <code>"gene_names"</code>.</li> </ul>
<code>gene_output</code>	<p>How to return genes. Options include:</p> <ul style="list-style-type: none"> <li>• <code>"rownames"</code> : As row names of <code>gene_df</code>.</li> <li>• <code>"colnames"</code> : As column names of <code>gene_df</code>.</li> <li>• <code>"columns"</code> : As new columns <code>"input_gene"</code>, <code>"ortholog_gene"</code> (and <code>"input_gene_standard"</code> if <code>standardise_genes=TRUE</code>) in <code>gene_df</code>.</li> <li>• <code>"dict"</code> : As a dictionary (named list) where the names are <code>input_gene</code> and the values are <code>ortholog_gene</code>.</li> <li>• <code>"dict_rev"</code> : As a reversed dictionary (named list) where the names are <code>ortholog_gene</code> and the values are <code>input_gene</code>.</li> </ul>
<code>standardise_genes</code>	<p>If TRUE AND <code>gene_output="columns"</code>, a new column <code>"input_gene_standard"</code> will be added to <code>gene_df</code> containing standardised HGNC symbols identified by <a href="#">gorth</a>.</p>
<code>input_species</code>	<p>Name of the input species (e.g., "mouse","fly"). Use <a href="#">map_species</a> to return a full list of available species.</p>
<code>output_species</code>	<p>Name of the output species (e.g. "human","chicken"). Use <a href="#">map_species</a> to return a full list of available species.</p>
<code>method</code>	<p>R package to use for gene mapping:</p> <ul style="list-style-type: none"> <li>• <code>"gprofiler"</code> : Slower but more species and genes.</li> <li>• <code>"homologene"</code> : Faster but fewer species and genes.</li> <li>• <code>"babelgene"</code> : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
<code>drop_nonorths</code>	<p>Drop genes that don't have an ortholog in the <code>output_species</code>.</p>

## 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 symbol standardization and aggregation/expansion.

<code>input_col</code>	Column name within <code>gene_map</code> with gene names matching the row names of <code>X</code> .
<code>output_col</code>	Column name within <code>gene_map</code> with gene names that you wish you map the row names of <code>X</code> onto.
<code>verbose</code>	Print messages.
<code>...</code>	Additional arguments to be passed to <a href="#">gorth</a> or <a href="#">homologene</a> .

*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 <b>orthogene</b> , 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 <a href="#">map_orthologs</a> or <a href="#">map_genes</a> .
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
non121_strategy	How to handle genes that don't have 1:1 mappings between input_species:output_species. Options include: <ul style="list-style-type: none"> <li>• "drop_both_species" or "dbs" or 1 : Drop genes that have duplicate mappings in either the input_species or output_species (<i>DEFAULT</i>).</li> <li>• "drop_input_species" or "dis" or 2 : Only drop genes that have duplicate mappings in the input_species.</li> </ul>



- "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 <a href="#">data.frame</a> .
formula	Casting <a href="#">formula</a> , 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 <code>model.matrix</code>

**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(
  evaluate_threshold = NULL,
  save_dir = tools::R_user_dir("orthogene", which = "cache")
)
```

**Arguments**

evaluate_threshold	Only include mappings with an E-value below a set threshold. See <a href="#">here</a> 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	<i>Gene expression data: mouse</i>
-----------	------------------------------------

---

**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	<i>Transcript expression data: mouse</i>
----------------	--

---

**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	<i>Format species names</i>
----------------	-----------------------------

---

## 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\_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 <a href="#">phylopic</a> 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 <a href="#">format_species</a> before querying <a href="#">phylopic</a> (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)
```

---

<code>ggtree_plot</code>	<i>Plot a phylogenetic tree</i>
--------------------------	---------------------------------

---

## 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

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

## Value

[ggplot](#) object.

---

gprofiler\_namespace     *gconvert namespaces*

---

### Description

Available namespaces used by link[gprofiler2]gconvert.

### Format

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

### 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_oradb_
#### Save #### usethis::use_data(gprofiler_orgs,gprofiler_namespace, overwrite = TRUE,
internal=TRUE)
```



---

infer_species	<i>Infer species from gene names</i>
---------------	--------------------------------------

---

## 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.
- codelist :  
A list or character vector.

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

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

gene\_input      Which aspect of gene\_df to get gene names from:

- "rownames" :  
From row names of data.frame/matrix.
- "colnames" :  
From column names of data.frame/matrix.
- <column name> :  
From a column in gene\_df, e.g. "gene\_names".

<code>test_species</code>	Which species to test for matches with. If set to NULL, will default to a list of humans and 5 common model organisms. If <code>test_species</code> is set to one of the following options, it will automatically pull all species from that respective package and test against each of them: <ul style="list-style-type: none"> <li>• "homologene" : 20+ species (default)</li> <li>• "gprofiler" : 700+ species</li> <li>• "babelgene" : 19 species</li> </ul>
<code>method</code>	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
<code>make_plot</code>	Make a plot of the results.
<code>show_plot</code>	Print the plot of the results.
<code>verbose</code>	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	<i>Invert dictionary</i>
-------------------	--------------------------

---

**Description**

Switch the names/items in a named list.

**Usage**

```
invert_dictionary(dict)
```

**Value**

Named list

---

many2many_rows	<i>Expand/aggregate rows of matrix for many:many mappings</i>
----------------	---

---

**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:

1. Identify genes that appear more than once in `gene_map[[input_col]]`.
2. 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).
3. Map rownames of the expanded matrix onto the orthologous gene names from `gene_map$ortholog_gene`.
4. [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 <a href="#">data.frame</a> generated by <a href="#">map_orthologs</a> , 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 <a href="#">DelayedArray</a> .
dropNA	Drop genes assigned to NA in groupings.
aggregate_orthologs	[Optional] After performing an initial round of many:many aggregation/expansion with <a href="#">many2many_rows</a> , ensure each orthologous gene only appears in one row by using the <a href="#">aggregate_rows</a> 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

<code>genes</code>	Gene list.
<code>species</code>	Species to map against.
<code>target</code>	target namespace.
<code>mthreshold</code>	maximum number of results per initial alias to show. Shows all by default.
<code>drop_na</code>	Drop all genes without mappings. Sets <code>gprofiler2::gconvert(filter_na=)</code> as well an additional round of more comprehensive NA filtering by <b>orthogene</b> .
<code>numeric_ns</code>	namespace to use for fully numeric IDs ( <a href="#">list of available namespaces</a> ).
<code>run_map_species</code>	Standardise species names with <a href="#">map_species</a> first (Default: TRUE).
<code>verbose</code>	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 (Schmidtd 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 = tools::R_user_dir("orthogene", which = "cache"),
  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	<i>Map orthologs</i>
---------------	----------------------

---

## 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 <a href="#">gorth</a> .
input_species	Name of the input species (e.g., "mouse","fly"). Use <a href="#">map_species</a> to return a full list of available species.
output_species	Name of the output species (e.g. "human","chicken"). Use <a href="#">map_species</a> to return a full list of available species.
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
mthreshold	Maximum number of ortholog names per gene to show. Passed to <a href="#">gorth</a> . Only used when method="gprofiler" ( <i>DEFAULT</i> : Inf).
gene_map	A <a href="#">data.frame</a> that maps the current gene names to new gene names. This function's behaviour will adapt to different situations as follows:

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

<code>input_col</code>	Column name within <code>gene_map</code> with gene names matching the row names of X.
<code>output_col</code>	Column name within <code>gene_map</code> with gene names that you wish you map the row names of X onto.
<code>verbose</code>	Print messages.
<code>...</code>	Additional arguments to be passed to <a href="#">gorth</a> or <a href="#">homologene</a> .

*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 <a href="#">map_species</a> to return a full list of available species.
output_species	Name of the output species (e.g. "human","chicken"). Use <a href="#">map_species</a> 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 <a href="#">gorth</a> or <a href="#">homologene</a> .

*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	<p>A <a href="#">data.frame</a> that maps the current gene names to new gene names. This function's behaviour will adapt to different situations as follows:</p> <ul style="list-style-type: none"> <li>• <code>gene_map=&lt;data.frame&gt;</code> : When a <code>data.frame</code> containing the gene key:value columns (specified by <code>input_col</code> and <code>output_col</code>, respectively) is provided, this will be used to perform aggregation/expansion.</li> <li>• <code>gene_map=NULL</code> and <code>input_species!=output_species</code> : A <code>gene_map</code> is automatically generated by <a href="#">map_orthologs</a> to perform inter-species gene aggregation/expansion.</li> <li>• <code>gene_map=NULL</code> and <code>input_species==output_species</code> : A <code>gene_map</code> is automatically generated by <a href="#">map_genes</a> to perform within-species gene symbol standardization and aggregation/expansion.</li> </ul>
input_species	Name of the input species (e.g., "mouse", "fly"). Use <a href="#">map_species</a> to return a full list of available species.
output_species	Name of the output species (e.g. "human", "chicken"). Use <a href="#">map_species</a> to return a full list of available species.
input_col	Column name within <code>gene_map</code> with gene names matching the row names of X.
output_col	Column name within <code>gene_map</code> 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,
  ...
)
```

## Arguments

genes	Gene list.
input_species	Name of the input species (e.g., "mouse","fly"). Use <a href="#">map_species</a> to return a full list of available species.
output_species	Name of the output species (e.g. "human","chicken"). Use <a href="#">map_species</a> to return a full list of available species.
filter_na	Logical indicating whether to filter out results without a corresponding target name. ( <i>DEFAULT</i> is FALSE, so that NAs can be handled by <b>orthogene</b> ).
mthreshold	Maximum number of ortholog names per gene to show. Passed to <a href="#">gorth</a> . Only used when method="gprofiler" ( <i>DEFAULT</i> : Inf).
verbose	Print messages.
...	Additional arguments to be passed to <a href="#">gorth</a> .

## Details

"mthreshold is used to set the maximum number of ortholog names per gene to show. This is useful to handle the problem of having many orthologs per gene (most of them uninformative). The function tries to find the most informative by selecting the most popular ones."

~ From [gprofiler2 vignette](#)

Available namespaces for the numeric\_ns argument can be found [here](#).

## 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 <a href="#">map_species</a> to return a full list of available species.
output_species	Name of the output species (e.g. "human","chicken"). Use <a href="#">map_species</a> to return a full list of available species.
verbose	Print messages.
...	Additional arguments to be passed to <a href="#">homologene</a> .

### 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 <a href="#">API</a> .
output_format	Which column to return.
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
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 <i>default</i> , <a href="#">map_species</a> 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	<i>Send messages to console even from within parallel processes</i>
------------------	---

---

**Description**

Send messages to console even from within parallel processes

**Usage**

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

**Value**

A message

---

plot_benchmark_bar	<i>Plot benchmark: bar</i>
--------------------	----------------------------

---

**Description**

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

**Usage**

```
plot_benchmark_bar(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\_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 <a href="#">phylo</a> . If no tree is provided (NULL) a 100-way multiz tree will be imported from <a href="#">UCSC Genome Browser</a> .
orth_report	An ortholog report from one or more species generated by <a href="#">report_orthologs</a> .
species	Species to include in the final plot. If NULL, then all species from the given database (method) will be included (via <a href="#">map_species</a> ), so long as they also exist in the tree.
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>"gprofiler" : Slower but more species and genes.</li> <li>"homologene" : Faster but fewer species and genes.</li> <li>"babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
tree_source	Can be one of the following: <ul style="list-style-type: none"> <li>"timetree2022": Import and prune the <a href="#">TimeTree &gt;147k species</a> phylogenetic tree. Can also simply type "timetree".</li> <li>"timetree2015": Import and prune the <a href="#">TimeTree &gt;50k species</a> phylogenetic tree.</li> <li>"OmaDB": Construct a tree from <a href="#">OMA</a> (Orthologous Matrix browser) via the <a href="#">getTaxonomy</a> function. <i>NOTE:</i> Does not contain branch lengths, and therefore may have limited utility.</li> <li>"UCSC": Import and prune the <a href="#">UCSC 100-way alignment</a> phylogenetic tree (hg38 version).</li> <li>"&lt;path&gt;": Read a tree from a newick text file from a local or remote URL using <a href="#">read.tree</a>.</li> </ul>



## 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

```
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 = tools::R_user_dir("orthogene", which = "cache"),
  verbose = TRUE,
  ...
)
```

### Arguments

<code>tree_source</code>	Can be one of the following: <ul style="list-style-type: none"> <li>• "timetree2022": Import and prune the <a href="#">TimeTree &gt;147k species</a> phylogenetic tree. Can also simply type "timetree".</li> </ul>
--------------------------	---

- "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 <a href="#">map_species</a> .
method	R package to use for gene mapping: <ul style="list-style-type: none"> <li>• "gprofiler" : Slower but more species and genes.</li> <li>• "homologene" : Faster but fewer species and genes.</li> <li>• "babelgene" : Faster but fewer species and genes. Also gives consensus scores for each gene mapping based on a several different data sources.</li> </ul>
force_ultrametric	Whether to force the tree to be ultrametric (i.e. make all tips the same date) using <a href="#">force.ultrametric</a> .
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 <a href="#">makeChronosCalib</a> and <a href="#">chronos</a> .
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 <a href="#">makeChronosCalib</a> .

**Value**

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

**Source**

[TimeTree 5: An Expanded Resource for Species Divergence Times](#)

**Examples**

```
species <- c("human", "chimp", "mouse")
tr <- orthogene::prepare_tree(species = species)
```

---

remove_image_bg	<i>Remove image background</i>
-----------------	--------------------------------

---

### Description

Import an 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 <b>color string</b> 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.svg")
img_res <- remove_image_bg(path=path)
```

---

report_orthologs	<i>Report orthologs</i>
------------------	-------------------------

---

### 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`  
R package to to use in [all\\_genes](#) step:

- "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.

`method_convert_orthologs`  
R package to to use in [convert\\_orthologs](#) step:

- "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

	<p>output_species (<i>DEFAULT</i>).</p> <ul style="list-style-type: none"> <li>• "drop_input_species" or "dis" or 2 : Only drop genes that have duplicate mappings in the input_species.</li> <li>• "drop_output_species" or "dos" or 3 : Only drop genes that have duplicate mappings in the output_species.</li> <li>• "keep_both_species" or "kbs" or 4 : Keep all genes regardless of whether they have duplicate mappings in either species.</li> <li>• "keep_popular" or "kp" or 5 : Return only the most "popular" interspecies ortholog mappings. This procedure tends to yield a greater number of returned genes but at the cost of many of them not being true biological 1:1 orthologs.</li> <li>• "sum", "mean", "median", "min" or "max" : When gene_df is a matrix and gene_output="rownames", these options will aggregate many-to-one gene mappings (input_species-to-output_species) after dropping any duplicate genes in the output_species.</li> </ul>
round_digits	Number of digits to round to when printing percentages.
return_report	Return just the ortholog mapping between two species (FALSE) or return both the ortholog mapping as well a data.frame of the report statistics (TRUE).
ref_genes	A table of all genes for the reference_species. If NULL (default), this will automatically be created using <a href="#">all_genes</a> .
mc.cores	Number of cores to parallelise each target_species with.
verbose	Print messages.
...	Arguments passed on to <a href="#">convert_orthologs</a>
gene_df	<p>Data object containing the genes (see gene_input for options on how the genes can be stored within the object). Can be one of the following formats:</p> <ul style="list-style-type: none"> <li>• matrix : A sparse or dense matrix.</li> <li>• data.frame : A data.frame, data.table. or tibble.</li> <li>• codelist : A list or character vector.</li> </ul> <p>Genes, transcripts, proteins, SNPs, or genomic ranges can be provided in any format (HGNC, Ensembl, RefSeq, UniProt, etc.) and will be automatically converted to gene symbols unless specified otherwise with the ... arguments.</p> <p><i>Note:</i> If you set method="homologene", you must either supply genes in gene symbol format (e.g. "Sox2") OR set standardise_genes=TRUE.</p>
gene_input	<p>Which aspect of gene_df to get gene names from:</p> <ul style="list-style-type: none"> <li>• "rownames" : From row names of data.frame/matrix.</li> </ul>

- "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 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 `>1 target_species` are provided, then a table of aggregated report statistics concatenated across species will be returned instead.

## Examples

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

---

run\_benchmark

*Run benchmark tests*

---

## Description

Runs benchmark 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,
  verbose = TRUE
)
```

**Arguments**

species            Species names.

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.

verbose           Print messages.

benchmark\_homologene  
                  Benchmark method "homologene".

benchmark\_gprofiler  
                  Benchmark method "gprofiler".

benchmark\_babelgene  
                  Benchmark method "babelgene".

**Value**

data.table with benchmark results

---

set_gprofiler	<i>Set gprofiler</i>
---------------	----------------------

---

**Description**

Set the default URL for gprofiler API queries.

- default: <http://biit.cs.ut.ee/gprofiler>
- bea: [http://biit.cs.ut.ee/gprofiler\\_beta](http://biit.cs.ut.ee/gprofiler_beta)

**Usage**

```
set_gprofiler(url = "http://biit.cs.ut.ee/gprofiler_beta")
```

**Arguments**

url                    the base URL.

**Value**

Null

---

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

- \* **datasets**
  - exp\_mouse, 19
  - exp\_mouse\_enst, 19
- \* **internal**
  - add\_synonyms, 4
  - aggregate\_rows, 6
  - aggregate\_rows\_monocle3, 7
  - all\_genes\_babelgene, 9
  - check\_gene\_df\_type, 11
  - dMcast, 17
  - earthworm2human\_map, 18
  - get\_orgdb\_genomeinfodbdata, 21
  - ggtree\_plot, 23
  - infer\_species\_plot, 26
  - invert\_dictionary, 27
  - many2many\_rows, 27
  - map\_genes\_planosphere, 30
  - map\_orthologs\_babelgene, 32
  - map\_orthologs\_custom, 34
  - map\_orthologs\_gprofiler, 35
  - map\_orthologs\_homologene, 36
  - message\_parallel, 38
  - plot\_benchmark\_bar, 38
  - plot\_benchmark\_scatter, 39
  - remove\_image\_bg, 44
  - run\_benchmark, 48
  - set\_gprofiler, 49
  - taxa\_id\_dict, 50
- add\_synonyms, 4
- aggregate\_mapped\_genes, 4, 14, 47
- aggregate\_rows, 6, 6, 27, 28
- aggregate\_rows\_monocle3, 7
- all\_genes, 8, 45, 46, 48
- all\_genes\_babelgene, 9
- all\_species, 10
- check\_gene\_df\_type, 11
- chronos, 43
- convert\_orthologs, 4, 12, 45, 46, 48, 49
- create\_background, 15
- data.frame, 4, 5, 14, 18, 28, 31, 34, 48
- data.table, 11, 19, 30
- DelayedArray, 6, 7, 14, 28, 48
- dMcast, 17
- earthworm2human\_map, 18
- exp\_mouse, 19
- exp\_mouse\_enst, 19
- floor, 6
- force.ultrametric, 43
- format\_species, 20, 22
- formula, 18
- gconvert, 9, 24, 29
- get\_orgdb\_genomeinfodbdata, 21
- get\_silhouettes, 22
- getTaxonomy, 40, 43
- ggplot, 23
- ggtree\_plot, 23
- gorth, 9, 13–15, 31–33, 35, 45, 47
- gprofiler\_namespace, 24
- gprofiler\_orgs, 24
- homologene, 9, 15, 32, 33, 36
- homologeneData, 9
- infer\_species, 25, 26
- infer\_species\_plot, 26
- invert\_dictionary, 27
- makeChronosCalib, 43
- many2many\_rows, 4, 6, 27, 28
- map\_genes, 5, 15, 16, 29, 32, 34, 48
- map\_genes\_planosphere, 30
- map\_orthologs, 5, 15, 16, 28, 31, 32, 34, 48
- map\_orthologs\_babelgene, 32
- map\_orthologs\_custom, 34
- map\_orthologs\_gprofiler, 35

map\_orthologs\_homologene, 36  
map\_species, 5, 9, 10, 12, 13, 29, 31, 33–36,  
36, 37, 40, 42, 43, 47  
message\_parallel, 38  
MRCA, 41  
  
orthogene (orthogene-package), 3  
orthogene-package, 3  
orthologs, 32  
  
phylo, 40  
plot\_benchmark\_bar, 38  
plot\_benchmark\_scatter, 39  
plot\_orthotree, 39  
prepare\_tree, 42  
  
read.tree, 40, 43  
remove\_image\_bg, 44  
report\_orthologs, 23, 40, 44  
rotate, 41  
run\_benchmark, 48  
  
set\_gprofiler, 49  
  
taxa\_id\_dict, 50