Package 'fenr'

October 30, 2026

Title Fast functional enrichment for interactive applications

Version 1.11.3

Description Perform fast functional enrichment on feature lists (like genes or proteins) using the hypergeometric distribution. Tailored for speed, this package is ideal for interactive platforms such as Shiny. It supports the retrieval of functional data from sources like GO, KEGG, Reactome, Bioplanet and WikiPathways. By downloading and preparing data first, it allows for rapid successive tests on various feature selections without the need for repetitive, time-consuming preparatory steps typical of other packages.

URL https://github.com/bartongroup/fenr

BugReports https://github.com/bartongroup/fenr/issues

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

biocViews FunctionalPrediction, DifferentialExpression, GeneSetEnrichment, GO, KEGG, Reactome, Proteomics

Imports tools, methods, assertthat, rlang, dplyr, tidyr, tidyselect, tibble, purrr, readr, stringr, httr2, progress, BiocFileCache, shiny, ggplot2

Suggests BiocStyle, testthat, knitr, rmarkdown, topGO

VignetteBuilder knitr

LazyData false

Config/roxygen2/version 8.0.0

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

git_branch devel

git_last_commit 78a5a12

git_last_commit_date 2026-09-23

Repository Bioconductor 3.24

Date/Publication 2026-10-29

Author Marek Gierlinski [aut, cre] (ORCID: <https://orcid.org/0000-0001-9149-3514>)

Maintainer Marek Gierlinski <M.Gierlinski@dundee.ac.uk>

Contents

fenr-package . . . 2
enrichment_interactive . . . 3
exmpl_all . . . 3
exmpl_sel . . . 4
fetch_bp . . . 4
fetch_go . . . 5
fetch_go_species . . . 6
fetch_kegg . . . 6
fetch_kegg_species . . . 7
fetch_reactome . . . 8
fetch_reactome_species . . . 9
fetch_terms_for_example . . . 9
fetch_wiki . . . 10
fetch_wiki_species . . . 11
functional_enrichment . . . 11
get_feature_terms . . . 12
get_go_legacy_mapping . . . 13
get_term_features . . . 13
go . . . 14
go_species . . . 14
prepare_for_enrichment . . . 15
remove_cache . . . 16
yeast_de . . . 16

Index **17**

`fenr-package` *fenr: Fast functional enrichment for interactive applications*

Description

This R package provides a fast and efficient method for functional enrichment analysis, optimized for speed and designed for use in interactive applications, such as *Shiny* apps.

To learn more about fenr, start with the vignette: vignette("fenr").

Value

Package fenr.

Author(s)

Maintainer: Marek Gierlinski <M.Gierlinski@dundee.ac.uk> (ORCID)

Authors:

- Marek Gierlinski <M.Gierlinski@dundee.ac.uk> (ORCID)

See Also

Useful links:

- https://github.com/bartongroup/fenr
- Report bugs at https://github.com/bartongroup/fenr/issues

`enrichment_interactive` *Small Shiny app serving as example for fast enrichment*

Description

Small Shiny app serving as example for fast enrichment

Usage

```
enrichment_interactive(de, term_data)
```

Arguments

`de`	Differential expression results, `yeast_de` attached to this package can be used.
`term_data`	A list of `fenr_terms` objects containing functional data for various ontologies. `fetch_terms_for_example` can be used to create this object.

Value

An interactive Shiny app

Examples

```
## Not run:
data(yeast_de)
term_data <- fetch_terms_for_example(yeast_de)
  enrichment_interactive(yeast_de, term_data)

## End(Not run)
```

`exmpl_all` *Example set of background genes.*

Description

A set of gene names for proteins that could be detected in a typical proteomics experiment on yeast samples.

Usage

```
data(exmpl_all)
```

Format

A character vector with 6985 elements.

Value

A vector with background gene names.

---

exmpl_sel *Example set of selected genes.*

Description

A set of gene names manually selected to illustrate functional enrichment.

Usage

```
data(exmpl_sel)
```

Format

A character vector with 21 elements.

Value

A vector with selected gene names.

---

fetch_bp *Get functional term data from BioPlanet*

Description

Download term information (term ID and name) and gene-pathway mapping (NCBI gene ID, gene symbol and pathway ID) from BioPlanet.

Usage

```
fetch_bp(use_cache = TRUE, on_error = c("stop", "warn", "ignore"))
```

Arguments

use_cache	Logical, if TRUE, the remote file will be cached locally.
on_error	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return ‘NULL‘ or "ignore" to return ‘NULL‘ without warnings. Defaults to "stop".

Value

A list with `terms` and `mapping` tibbles.

Examples

```
bioplanet_data <- fetch_bp(on_error = "warn")
```

fetch_go *Get Gene Ontology (GO) data*

Description

This function downloads term information (GO term ID and name) and gene-term mapping (gene ID, symbol, and GO term ID) from either the Ensembl database (using BioMart) or the Gene Ontology database (using GAF files), depending on the provided argument.

Usage

```
fetch_go(
  species = NULL,
  dataset = NULL,
  use_cache = TRUE,
  on_error = c("stop", "warn", "ignore")
)
```

Arguments

species	(Optional) Species designation. Examples are HUMAN-uniprot for human, MOUSE-mod for mouse, or YEAST-mod for yeast. Legacy names such as goa_human, mgi, and sgd are also accepted when they map to a single current designation. Current species designations can be obtained using fetch_go_species; legacy mappings can be inspected using get_go_legacy_mapping. This argument is used when fetching data from the Gene Ontology database.
dataset	(Optional) A string representing the dataset passed to Ensembl's Biomart, e.g. 'scerevisiae_gene_ensembl'. To see the different datasets available within a biomaRt you can e.g. do: mart <- biomaRt::useEnsembl(biomart = "ensembl"), followed by biomaRt::listDatasets(mart).
use_cache	Logical, if TRUE, the remote data will be cached locally.
on_error	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return 'NULL' or "ignore" to return 'NULL' without warnings. Defaults to "stop".

Details

If species is provided, mapping from a Gene Ontology GAF file will be downloaded. GAF files contain more generic information than gene symbols. In this function, the third column of the GAF file (DB Object Symbol) is returned as gene_symbol, but, depending on the species argument it can contain other entities, e.g. RNA or protein complex names. Similarly, the eleventh column of the GAF file (DB Object Synonym) is returned as gene_id. It is up to the user to select the appropriate database.

Alternatively, if dataset is provided, mapping will be downloaded from Ensembl database. It will contain gene symbols and Ensembl gene IDs.

Value

A list with terms and mapping tibbles.

Examples

```
# Fetch GO data from Ensembl
go_data_ensembl <- fetch_go(dataset = "scerevisiae_gene_ensembl", on_error = "warn")
# Fetch GO data from Gene Ontology
go_data_go <- fetch_go(species = "YEAST-mod", on_error = "warn")
```

`fetch_go_species` *Find all species available from geneontology.org*

Description

This function downloads and parses Gene Ontology's organism metadata table to return the current annotation file species designations. By default, one preferred designation is returned for each species. Model Organism Database (MOD) annotations are preferred when they are available; otherwise UniProt annotations are returned. Set `all_sources = TRUE` to return both MOD and UniProt designations.

Usage

```
fetch_go_species(on_error = c("stop", "warn", "ignore"), all_sources = FALSE)
```

Arguments

`on_error`	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return 'NULL' or "ignore" to return 'NULL' without warnings. Defaults to "stop".
`all_sources`	Logical, if TRUE, return all available annotation sources for each species. If FALSE, return one preferred designation per species.

Value

A tibble with Gene Ontology species metadata. Column `designation` contains species designations used in function `fetch_go`.

Examples

```
go_species <- fetch_go_species(on_error = "warn")
go_species_all <- fetch_go_species(on_error = "warn", all_sources = TRUE)
```

`fetch_kegg` *Get functional term data from KEGG*

Description

Download information (pathway ID and name) and gene-pathway mapping (entrez gene ID, gene symbol and pathway ID) from KEGG. Gene symbols are extracted from gene descriptions. For some species (e.g. yeast), gene symbols are returned instead of entrez IDs and not in gene description.

Usage

```
fetch_kegg(species, batch_size = 10, on_error = c("stop", "warn", "ignore"))
```

Arguments

species	KEGG species code, for example "hsa" for human. The full list of available KEGG species can be found by using fetch_kegg_species.
batch_size	Number of pathways sent to KEGG database in one query. The maximum allowed is 10.
on_error	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return 'NULL' or "ignore" to return 'NULL' without warnings. Defaults to "stop".

Value

A list with terms and mapping tibbles.

Examples

```
kegg_data <- fetch_kegg("mge", on_error = "warn")
```

---

fetch_kegg_species *Find all species available from KEGG*

Description

Find all species available from KEGG

Usage

```
fetch_kegg_species(on_error = c("stop", "warn", "ignore"))
```

Arguments

on_error	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return 'NULL' or "ignore" to return 'NULL' without warnings. Defaults to "stop".

Value

A tibble, in which column designation contains species designations used in function fetch_kegg.

Examples

```
spe <- fetch_kegg_species(on_error = "warn")
```

`fetch_reactome` *Get functional term data from Reactome*

Description

Download term information (pathway ID and name) and gene-pathway mapping (Ensembl gene ID or gene symbol and pathway ID) from Reactome.

Usage

```
fetch_reactome(
  species,
  source = c("ensembl", "api", "gene_association"),
  use_cache = TRUE,
  on_error = c("stop", "warn", "ignore")
)
```

Arguments

`species`	Reactome species designation, for example "Homo sapiens" for human. Full list of available species can be found using `fetch_reactome_species()`.
`source`	How to download the mapping. If 'ensembl' or 'gene_association', one mapping file provided by Reactome will be downloaded, if 'api', then Reactome API will be used. See details.
`use_cache`	Logical, if TRUE, the remote file will be cached locally.
`on_error`	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return 'NULL' or "ignore" to return 'NULL' without warnings. Defaults to "stop".

Details

Reactome makes mapping between Ensembl ID and pathway ID available in form of one downloadable file. This mapping contains gene symbols as well. Also, a gene association file with mapping between UniProt accession number, gene symbol and Reactome term is available. If `source = "ensembl"` or `source = "gene_association"` is set, one large file will be downloaded and parsed. If `source = "api"` is set, then Reactome APIs will be interrogated for each pathway available. This method is considerably slower, especially for large genomes. However, gene association file contains far fewer mappings than can be extracted using API. If gene symbols are needed, we recommend using `source = "ensembl"`.

Value

A list with `terms` and `mapping` tibbles

Examples

```
reactome_data <- fetch_reactome("Saccharomyces cerevisiae", on_error = "warn")
```

`fetch_reactome_species` *List of available Reactome species*

Description

List of available Reactome species

Usage

```
fetch_reactome_species(on_error = c("stop", "warn", "ignore"))
```

Arguments

on_error	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return 'NULL' or "ignore" to return 'NULL' without warnings. Defaults to "stop".

Value

A tibble with species names used by Reactome.

Examples

```
re <- fetch_reactome_species(on_error = "warn")
```

`fetch_terms_for_example` *Create term data for interactive example*

Description

Create term data for interactive example

Usage

```
fetch_terms_for_example(de)
```

Arguments

de	Differential expression results, use `yeast_de` data attached to this package.

Value

A list of objects containing functional terms for GO and Reactome.

Examples

```
## Not run:
data(yeast_de)
term_data <- fetch_terms_for_example(yeast_de)

## End(Not run)
```

`fetch_wiki` *Get functional term data from WikiPathways*

Description

Download term information (pathway ID and name) and gene-pathway mapping (gene symbol and pathway ID) from WikiPathways.

Usage

```
fetch_wiki(
  species,
  databases = c("Ensembl", "Entrez Gene", "HGNC", "HGNC Accession number",
    "Uniprot-TrEMBL"),
  types = c("GeneProduct", "Protein", "Rna", "RNA"),
  on_error = c("stop", "warn", "ignore")
)
```

Arguments

`species`	WikiPathways species designation, for example "Homo sapiens" for human. Full list of available species can be found using `fetch_wiki_species()`.
`databases`	A character vector with database names to pre-filter mapping data. See details. Full result will be returned if NULL.
`types`	A character vector with types of entities to pre-filter mapping data. See details. Full result will be returned if NULL.
`on_error`	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return ‘NULL‘ or "ignore" to return ‘NULL‘ without warnings. Defaults to "stop".

Details

WikiPathways contain mapping between pathways and a variety of entities from various databases. Typically a gene symbol is returned in column `text_label` and some sort of ID in column `id`, but this depends on the species and databases used. For gene/protein enrichment, these should be filtered to contain gene symbols only. This can be done by selecting a desired databases and types. The default values for parameters `databases` and `types` attempt to select information from generic databases, but there are organism-specific databases not included in the selection. We suggest to run this function with `databases = NULL, types = NULL` to see what types and databases are available before making selection.

Value

A list with `terms` and `mapping` tibbles.

Examples

```
wiki_data <- fetch_wiki("Bacillus subtilis", on_error = "warn")
```

fetch_wiki_species *List of available WikiPathways species*

Description

List of available WikiPathways species

Usage

```
fetch_wiki_species(on_error = c("stop", "warn", "ignore"))
```

Arguments

on_error	A character string indicating the error handling strategy: either "stop" to halt execution, "warn" to issue a warning and return ‘NULL‘ or "ignore" to return ‘NULL‘ without warnings. Defaults to "stop".

Value

A character vector with species names used by WikiPathways.

Examples

```
spec <- fetch_wiki_species(on_error = "warn")
```

functional_enrichment *Fast Functional Enrichment*

Description

Perform fast functional enrichment analysis based on the hypergeometric distribution. Designed for use in interactive applications.

Usage

```
functional_enrichment(feat_all, feat_sel, term_data, feat2name = NULL)
```

Arguments

feat_all	A character vector with all feature identifiers, serving as the background for enrichment.
feat_sel	A character vector with feature identifiers in the selection.
term_data	An object of class fenr_terms, created by prepare_for_enrichment.
feat2name	An optional named list to convert feature IDs into feature names.

Details

This function carries out functional enrichment analysis on a selection of features (e.g., differentially expressed genes) using the hypergeometric probability distribution (Fisher's exact test). Features can be genes, proteins, etc. The term_data object contains functional term information and feature-term mapping.

Value

A tibble with enrichment results, providing the following information for each term:

- N_with - number of features with this term among all features
- n_with_sel - number of features with this term in the selection
- n_expect - expected number of features with this term in the selection, under the null hypothesis that terms are mapped to features randomly
- enrichment - ratio of n_with_sel / n_expect
- odds_ratio - odds ratio for enrichment; is infinite when all features with the given term are in the selection
- p_value - p-value from a single hypergeometric test
- p_adjust - p-value adjusted for multiple tests using the Benjamini-Hochberg approach

.

Examples

```
## Not run:
data(exmpl_all, exmpl_sel)
go <- fetch_go(species = "YEAST-mod")
go_terms <- prepare_for_enrichment(go$terms, go$mapping, exmpl_all, feature_name = "gene_symbol")
enr <- functional_enrichment(exmpl_all, exmpl_sel, go_terms)

## End(Not run)
```

---

get_feature_terms *Get terms annotating a given feature.*

Description

Get terms annotating a given feature.

Usage

```
get_feature_terms(term_data, feature_id)
```

Arguments

term_data	An object class fenr_terms, created by prepare_for_enrichment.
feature_id	A string with a feature ID

Value

A character vector containing functional term IDs annotating given feature.

Examples

```
## Not run:
go_data <- fetch_go(species = "YEAST-mod")
go_terms <- prepare_for_enrichment(go_data$terms, go_data$mapping, feature = "gene_symbol")
trms <- get_feature_terms(go_terms, "GEM1")

## End(Not run)
```

get_go_legacy_mapping *Get Gene Ontology legacy species mapping*

Description

Returns the packaged mapping between legacy Gene Ontology annotation species names and current annotation file designations. This table is used internally to keep legacy `species` values working in `fetch_go()`, but is also exposed so users can inspect or update their own code. Some legacy names map to more than one current designation.

Usage

```
get_go_legacy_mapping()
```

Value

A tibble with columns `legacy_name` and `designation`.

Examples

```
go_legacy_mapping <- get_go_legacy_mapping()
```

get_term_features *Get features annotated with a given term.*

Description

Get features annotated with a given term.

Usage

```
get_term_features(term_data, term_id)
```

Arguments

`term_data`	An object class `fenr_terms`, created by `prepare_for_enrichment`.
`term_id`	A string with a functional term ID.

Value

A character vector containing feature IDs annotated with the term ID.

Examples

```
## Not run:
go_data <- fetch_go(species = "YEAST-mod")
go_terms <- prepare_for_enrichment(go_data$terms, go_data$mapping, feature = "gene_symbol")
feats <- get_term_features(go_terms, "GO:0000001")

## End(Not run)
```

go *GO-terms data downloaded for the vignette.*

Description

Downloaded using `go <- fetch_go(species = "YEAST-mod")`

Usage

```
data(go)
```

Format

A list of two tibbles

Value

Contains GO-term descriptions and gene mapping.

go_species *GO species*

Description

Downloaded using `go_species <- fetch_go_species()`

Usage

```
data(go_species)
```

Format

A tibble

Value

Contains species available through Gene Ontology

prepare_for_enrichment *Prepare Term Data for Enrichment Analysis*

Description

Process term data downloaded with the `fetch_*` functions, preparing it for fast enrichment analysis using `functional_enrichment`.

Usage

```
prepare_for_enrichment(
  terms,
  mapping,
  all_features = NULL,
  feature_name = "gene_id"
)
```

Arguments

`terms`	A tibble with at least two columns: `term_id` and `term_name`. This tibble contains information about functional term names and descriptions.
`mapping`	A tibble with at least two columns, containing the mapping between functional terms and features. One column must be named `term_id`, while the other column should have a name specified by the `feature_name` argument. For example, if `mapping` contains columns `term_id`, `accession_number`, and `gene_symbol`, setting `feature_name = "gene_symbol"` indicates that gene symbols will be used for enrichment analysis.
`all_features`	A vector with all feature IDs used as the background for enrichment. If not specified, all features found in `mapping` will be used, resulting in a larger object size.
`feature_name`	The name of the column in the `mapping` tibble to be used as the feature identifier. For example, if `mapping` contains columns `term_id`, `accession_number`, and `gene_symbol`, setting `feature_name = "gene_symbol"` indicates that gene symbols will be used for enrichment analysis.

Details

This function takes two tibbles containing functional term information (`terms`) and feature mapping (`mapping`), and converts them into an object required by `functional_enrichment` for efficient analysis. Terms and mapping can be generated with the database access functions included in this package, such as `fetch_reactome` or `fetch_go_from_go`.

Value

An object of class `fenr_terms` required by `functional_enrichment`.

Examples

```
## Not run:
data(exmpl_all)
go <- fetch_go(species = "YEAST-mod")
go_terms <- prepare_for_enrichment(go$terms, go$mapping, exmpl_all,
                                   feature_name = "gene_symbol")

## End(Not run)
```

`remove_cache` *Remove all cache*

Description

This function will remove all cached data used by 'fenr'. The user will be prompted for confirmation. Use only when you suspect the cache was corrupted. Use with caution!

Usage

```
remove_cache(ask = TRUE)
```

Arguments

`ask`	Logical, whether to ask user for confirmation.

Value

TRUE if successfully removed.

Examples

```
## Not run:
remove_cache()

## End(Not run)
```

`yeast_de` *Differential expression results for yeast RNA-seq.*

Description

A subset of 6 + 6 replicates was selected from data set reported in https://doi.org/10.1093/bioinformatics/btv425

Usage

```
data(yeast_de)
```

Format

A tibble with 5 columns

Value

Results for differential expression for yeast RNA-seq.

Index

∗ **datasets**
    exmpl_all, 3
    exmpl_sel, 4
    go, 14
    go_species, 14
    yeast_de, 16
∗ **internal**
    fenr-package, 2

enrichment_interactive, 3
exmpl_all, 3
exmpl_sel, 4

fenr *(*fenr-package*)*, 2
fenr-package, 2
fetch_bp, 4
fetch_go, 5
fetch_go_species, 6
fetch_kegg, 6
fetch_kegg_species, 7
fetch_reactome, 8
fetch_reactome_species, 9
fetch_terms_for_example, 9
fetch_wiki, 10
fetch_wiki_species, 11
functional_enrichment, 11

get_feature_terms, 12
get_go_legacy_mapping, 13
get_term_features, 13
go, 14
go_species, 14

prepare_for_enrichment, 15

remove_cache, 16

yeast_de, 16