Package 'OmnipathR'

March 28, 2021

Type Package

Title OmniPath web service client

Version 2.99.6

Description A client for the OmniPath web service (https://www.omnipathdb.org) and many other resources. It also includes functions to transform and pretty print some of the downloaded data, functions to access a number of other resources such as BioPlex, ConsensusPathDB, EVEX, Guide to Pharmacology (IUPHAR/BPS), Harmonizome, HTRIdb, InWeb InBioMap, Pathway Commons, Ramilowski et al. 2015, RegNetwork, ReMap, TF census, TRRUST and Vinayagam et al. 2011. Furthermore, OmnipathR features a close integration with the NicheNet method for ligand activity prediction from transcriptomics data, and its R implementation `nichenetr` (available only on github).

License MIT + file LICENSE

URL https://saezlab.github.io/OmnipathR/

BugReports https://github.com/saezlab/OmnipathR/issues

biocViews GraphAndNetwork, Network, Pathways, Software, ThirdPartyClient, DataImport, DataRepresentation, GeneSignaling, GeneRegulation, SystemsBiology, Transcriptomics, SingleCell, Annotation

Encoding UTF-8

VignetteBuilder knitr

Depends R(>= 4.1)

Imports utils, jsonlite, progress, magrittr, purrr, RCurl, readr, dplyr, tidyr, rappdirs, checkmate, yaml, digest, wand, tools, httr, stringr, readxl, rlang, tibble, igraph, stats, logger, tidyselect

Suggests dnet, gprofiler2, BiocStyle, testthat, knitr, rmarkdown, ggplot2, ggraph, mlrMBO, smoof, ParamHelpers

RoxygenNote 7.1.1

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

git_branch master

git_last_commit 3d75274

git_last_commit_date 2021-03-26

Date/Publication 2021-03-28

Author Alberto Valdeolivas [aut] (<https://orcid.org/0000-0001-5482-9023>),
Denes Turei [cre, aut] (<https://orcid.org/0000-0002-7249-9379>),
Attila Gabor [aut] (<https://orcid.org/0000-0002-0776-1182>)

Maintainer Denes Turei <turei.denes@gmail.com>

R topics documented:

`.omnipath_options_defaults` *Default values for the package options*

Description

These options describe the default settings for OmnipathR so you do not need to pass these parameters at each function call. Currently the only option useful for the public web service at omnipathdb.org is "omnipath.license". If you are a for-profit user set it to "'commercial'" to make sure all the data you download from OmniPath is legally allowed for commercial use. Otherwise just leave it as it is: "'academic'". If you don't use omnipathdb.org but within your organization you deployed your own pypath server and want to share data whith a limited availability to outside users, you may want to use a password. For this you can use the "omnipath.password" option. Also if you want the R package to work from another pypath server instead of omnipathdb.org, you can change the option "omnipath.url".

Usage

```
.omnipath_options_defaults
```

Format

An object of class `list` of length 34.

`all_uniprots` *A table with all UniProt IDs*

Description

Retrieves a table from UniProt with all proteins for a certain organism.

Usage

```
all_uniprots(fields = "id", reviewed = TRUE, organism = 9606)
```

Arguments

`fields`	Character vector of fields as defined by UniProt. For possible values please refer to https://www.uniprot.org/help/uniprotkb%5Fcolumn%5Fnames
`reviewed`	Retrieve only reviewed ('TRUE'), only unreviewed ('FALSE') or both ('NULL').
`organism`	Integer, NCBI Taxonomy ID of the organism (by default 9606 for human).

Value

Data frame (tibble) with the requested UniProt entries and fields.

Examples

```
human_swissprot_ac <- all_uniprots(fields = 'entry name')
human_swissprot_ac
# # A tibble: 20,396 x 1
#    `Entry name`
#    <chr>
#  1 OR4K3_HUMAN
#  2 O52A1_HUMAN
#  3 O2AG1_HUMAN
#  4 O10S1_HUMAN
#  5 O11G2_HUMAN
# # . with 20,386 more rows
```

---

annotated_network *Network interactions with annotations*

Description

Annotations are often useful in a network context, e.g. one might want to label the interacting partners by their pathway membership. This function takes a network data frame and joins an annotation data frame from both the left and the right side, so both the source and target molecular entities will be labeled by their annotations. If one entity has many annotations these will yield many rows, hence the interacting pairs won't be unique across the data frame any more. Also if one entity has really many annotations the resulting data frame might be huge, we recommend to be careful with that. Finally, if you want to do the same but with intercell annotations, there is the `import_intercell_network` function.

Usage

```
annotated_network(network = NULL, annot = NULL, ...)
```

Arguments

`network`	Behaviour depends on type: if list, will be passed as arguments to `import_omnipath_interactions` to obtain a network data frame; if a data frame or tibble, it will be used as a network data frame; if a character vector, will be assumed to be a set of resource names and interactions will be queried from these resources.
`annot`	Either the name of an annotation resource (for a list of available resources call `get_annotation_resources`), or an annotation data frame. If the data frame contains more than one resources, only the first one will be used.
`...`	Column names selected from the annotation data frame (passed to `dplyr::select`, if empty all columns will be selected.)

Value

A data frame of interactions with annotations for both interacting entities.

Examples

```
signalink_with_pathways <-
    annotated_network('SignaLink3', 'SignaLink_pathway')
```

`bioplex1` *Downloads the BioPlex version 1.0 interaction dataset*

Description

This dataset contains ~24,000 interactions detected in HEK293T cells using 2,594 baits. More details at https://bioplex.hms.harvard.edu/interactions.php.

Usage

```
bioplex1()
```

Value

Data frame (tibble) with interactions.

See Also

- bioplex2
- bioplex3
- bioplex_hct116_1
- bioplex_all

Examples

```
bioplex_interactions <- bioplex1()
nrow(bioplex_interactions)
# [1] 23744
colnames(bioplex_interactions)
# [1] "GeneA"         "GeneB"        "UniprotA"   "UniprotB"
# [5] "SymbolA"       "SymbolB"      "p_wrong"    "p_no_interaction"
# [9] "p_interaction"
```

bioplex2 *Downloads the BioPlex version 2.0 interaction dataset*

Description

This dataset contains ~56,000 interactions detected in HEK293T cells using 5,891 baits. More details at https://bioplex.hms.harvard.edu/interactions.php

Usage

```
bioplex2()
```

Value

Data frame (tibble) with interactions.

See Also

- bioplex1
- bioplex3
- bioplex_hct116_1
- bioplex_all

Examples

```
bioplex_interactions <- bioplex2()
nrow(bioplex_interactions)
# [1] 56553
colnames(bioplex_interactions)
# [1] "GeneA"         "GeneB"        "UniprotA"   "UniprotB"
# [5] "SymbolA"       "SymbolB"      "p_wrong"    "p_no_interaction"
# [9] "p_interaction"
```

bioplex3 *Downloads the BioPlex version 3.0 interaction dataset*

Description

This dataset contains ~120,000 interactions detected in HEK293T cells using 10,128 baits. More details at https://bioplex.hms.harvard.edu/interactions.php.

Usage

```
bioplex3()
```

Value

Data frame (tibble) with interactions.

See Also

- bioplex1
- bioplex2
- bioplex_hct116_1
- bioplex_all

Examples

```
bioplex_interactions <- bioplex3()
nrow(bioplex_interactions)
# [1] 118162
colnames(bioplex_interactions)
# [1] "GeneA"         "GeneB"        "UniprotA"   "UniprotB"
# [5] "SymbolA"       "SymbolB"      "p_wrong"    "p_no_interaction"
# [9] "p_interaction"
```

---

bioplex_all *Downloads all BioPlex interaction datasets*

Description

BioPlex provides four interaction datasets: version 1.0, 2.0, 3.0 and HCT116 version 1.0. This function downloads all of them, merges them to one data frame, removes the duplicates (based on unique pairs of UniProt IDs) and separates the isoform numbers from the UniProt IDs. More details at https://bioplex.hms.harvard.edu/interactions.php.

Usage

```
bioplex_all(unique = TRUE)
```

Arguments

unique	Logical. Collapse the duplicate interactions into single rows or keep them as they are. In case of merging duplicate records the maximum p value will be choosen for each record.

Value

Data frame (tibble) with interactions.

See Also

- bioplex1
- bioplex2
- bioplex3
- bioplex_hct116_1

Examples

```
bioplex_interactions <- bioplex_all()
bioplex_interactions
# # A tibble: 195,538 x 11
#    UniprotA IsoformA UniprotB IsoformB GeneA GeneB SymbolA SymbolB
#    <chr>       <int> <chr>       <int> <dbl> <dbl> <chr>   <chr>
#  1 A0AV02          2 Q5K4L6         NA 84561 11000 SLC12A8 SLC27A3
#  2 A0AV02          2 Q8N5V2         NA 84561 25791 SLC12A8 NGEF
#  3 A0AV02          2 Q9H6S3         NA 84561 64787 SLC12A8 EPS8L2
#  4 A0AV96          2 O00425          2 54502 10643 RBM47   IGF2BP3
#  5 A0AV96          2 O00443         NA 54502  5286 RBM47   PIK3C2A
#  6 A0AV96          2 O43426         NA 54502  8867 RBM47   SYNJ1
#  7 A0AV96          2 O75127         NA 54502 26024 RBM47   PTCD1
#  8 A0AV96          2 O95208          2 54502 22905 RBM47   EPN2
#  9 A0AV96          2 O95900         NA 54502 26995 RBM47   TRUB2
# 10 A0AV96          2 P07910          2 54502  3183 RBM47   HNRNPC
# # . with 195,528 more rows, and 3 more variables: p_wrong <dbl>,
# #   p_no_interaction <dbl>, p_interaction <dbl>
```

---

bioplex_hct116_1 *Downloads the BioPlex HCT116 version 1.0 interaction dataset*

Description

This dataset contains ~71,000 interactions detected in HCT116 cells using 5,522 baits. More details at https://bioplex.hms.harvard.edu/interactions.php.

Usage

```
bioplex_hct116_1()
```

Value

Data frame (tibble) with interactions.

See Also

- bioplex1
- bioplex2
- bioplex3
- bioplex_all

Examples

```
bioplex_interactions <- bioplex_hct116_1()
nrow(bioplex_interactions)
# [1] 70966
colnames(bioplex_interactions)
# [1] "GeneA"         "GeneB"        "UniprotA"   "UniprotB"
# [5] "SymbolA"       "SymbolB"      "p_wrong"    "p_no_interaction"
# [9] "p_interaction"
```

---

bma_motif_es *BMA motifs from a sequence of edges*

Description

These motifs can be added to a BMA canvas.

Usage

```
bma_motif_es(edge_seq, G, granularity = 2)
```

Arguments

edge_seq	An igraph edge sequence.
G	An igraph graph object.
granularity	Numeric: granularity value.

Value

Character: BMA motifs as a single string.

Examples

```
interactions <- import_omnipath_interactions(resources = 'ARN')
graph <- interaction_graph(interactions)
motifs <- bma_motif_es(igraph::E(graph)[1], graph)
```

bma_motif_vs *Prints a BMA motif to the screen from a sequence of nodes, which can be copy/pasted into the BMA canvas*

Description

Intended to parallel print_path_vs

Usage

```
bma_motif_vs(node_seq, G)
```

Arguments

node_seq	An igraph node sequence.
G	An igraph graph object.

Value

Character: BMA motifs as a single string.

Examples

```
interactions <- import_omnipath_interactions(resources = 'ARN')
graph <- interaction_graph(interactions)
bma_string <- bma_motif_vs(
    igraph::all_shortest_paths(
        graph,
        from = 'ULK1',
        to = 'ATG13'
    )$res,
    graph
)
```

consensuspathdb_download *Retrieves the ConsensusPathDB network*

Description

Compiles a table of binary interactions from ConsensusPathDB (http://cpdb.molgen.mpg.de/) and translates the UniProtKB ACs to Gene Symbols.

Usage

```
consensuspathdb_download(complex_max_size = 4, min_score = 0.9)
```

Arguments

`complex_max_size`
: Numeric: do not expand complexes with a higher number of elements than this. ConsensusPathDB does not contain conventional interactions but lists of participants, which might be members of complexes. Some records include dozens of participants and expanding them to binary interactions result thousands, sometimes hundreds of thousands of interactions from one single record. At the end, this process consumes >10GB of memory and results rather unusable data, hence it is recommended to limit the complex sizes at some low number.

`min_score`
: Numeric: each record in ConsensusPathDB comes with a confidence score, expressing the amount of evidences. The default value, a minimum score of 0.9 retains approx. the top 30 percent of the interactions.

Value

Data frame (tibble) with interactions.

Examples

```
cpdb_data <- consensuspathdb_download(
    complex_max_size = 1,
    min_score = .98
)
nrow(cpdb_data)
# [1] 252302
colnames(cpdb_data)
# [1] "databases"  "references" "uniprot_a"    "confidence"   "record_id"
# [6] "uniprot_b"  "in_complex" "genesymbol_a" "genesymbol_b"
cpdb_data
# # A tibble: 252,302 x 9
#    databases references uniprot_a confidence record_id uniprot_b in_com
#    <chr>     <chr>      <chr>          <dbl>     <int> <chr>     <lgl>
#  1 Reactome  NA         SUMF2_HU.      1             1 SUMF1_HU. TRUE
#  2 Reactome  NA         SUMF1_HU.      1             1 SUMF2_HU. TRUE
#  3 DIP,Reac. 22210847,. STIM1_HU.      0.998         2 TRPC1_HU. TRUE
#  4 DIP,Reac. 22210847,. TRPC1_HU.      0.998         2 STIM1_HU. TRUE
# # . with 252,292 more rows, and 2 more variables: genesymbol_a <chr>,
# #   genesymbol_b <chr
```

---

consensuspathdb_raw_table

Downloads interaction data from ConsensusPathDB

Description

Downloads interaction data from ConsensusPathDB

Usage

```
consensuspathdb_raw_table()
```

Value

Data frame (tibble) with interactions.

Examples

```
cpdb_raw <- consensuspathdb_raw_table()
```

---

enzsub_graph *Enzyme-substrate graph*

Description

Transforms the a data frame with enzyme-substrate relationships (obtained by import_omnipath_enzsub) to an igraph graph object.

Usage

```
enzsub_graph(enzsub)
```

Arguments

enzsub	Data frame created by import_omnipath_enzsub

Value

An igraph directed graph object.

See Also

- import_omnipath_enzsub
- giant_component
- find_all_paths

Examples

```
enzsub <- import_omnipath_enzsub(resources = c('PhosphoSite', 'SIGNOR'))
enzsub_g <- enzsub_graph(enzsub = enzsub)
```

evex_download *Interactions from the EVEX database*

Description

Downloads interactions from EVEX, a versatile text mining resource (http://evexdb.org). Translates the Entrez Gene IDs to Gene Symbols and combines the interactions and references into a single data frame.

Usage

```
evex_download(
    min_confidence = NULL,
    remove_negatives = TRUE,
    top_confidence = NULL
)
```

Arguments

min_confidence	Numeric: a threshold for confidence scores. EVEX confidence scores span roughly from -3 to 3. By providing a numeric value in this range the lower confidence interactions can be removed. If NULL no filtering performed.
remove_negatives	Logical: remove the records with the "negation" attribute set.
top_confidence	Confidence cutoff as quantile (a number between 0 and 1). If NULL no filtering performed.

Value

Data frame (tibble) with interactions.

Examples

```
evex_interactions <- evex_download()
evex_interactions
# # A tibble: 368,297 x 13
#   general_event_id source_entrezge. target_entrezge. confidence negation
#              <dbl> <chr>            <chr>                 <dbl>    <dbl>
# 1               98 8651             6774                 -1.45         0
# 2              100 8431             6774                 -1.45         0
# 3              205 6261             6263                  0.370        0
# 4              435 1044             1045                 -1.09         0
# . with 368,287 more rows, and 8 more variables: speculation <dbl>,
#   coarse_type <chr>, coarse_polarity <chr>, refined_type <chr>,
#   refined_polarity <chr>, source_genesymbol <chr>,
#   target_genesymbol <chr>, references <chr>
```

filter_by_resource *Filters OmniPath data by resources*

Description

Keeps only those records which are supported by any of the resources of interest.

Usage

```
filter_by_resource(data, resources = NULL)
```

Arguments

data	A data frame downloaded from the OmniPath web service (interactions, enzyme-substrate or complexes).
resources	Character vector with resource names to keep.

Value

The data frame filtered.

Examples

```
interactions <- import_omnipath_interactions()
signor <- filter_by_resource(interactions, resources = 'SIGNOR')
```

find_all_paths *All paths between two groups of vertices*

Description

Finds all paths up to length ‘maxlen‘ between specified groups of vertices. This function is needed only becaues igraph‘s ‘all_shortest_paths‘ finds only the shortest, not any path up to a defined length.

Usage

```
find_all_paths(
    graph,
    start,
    end,
    attr = NULL,
    mode = 'OUT',
    maxlen = 2,
    progress = TRUE
)
```

Arguments

graph	An igraph graph object.
start	Integer or character vector with the indices or names of one or more start vertices.
end	Integer or character vector with the indices or names of one or more end vertices.
attr	Character: name of the vertex attribute to identify the vertices by. Necessary if 'start' and 'end' are not igraph vertex ids but for example vertex names or labels.
mode	Character: IN, OUT or ALL. Default is OUT.
maxlen	Integer: maximum length of paths in steps, i.e. if maxlen = 3, then the longest path may consist of 3 edges and 4 nodes.
progress	Logical: show a progress bar. Default is FALSE.

Value

List of vertex paths, each path is a character or integer vector.

See Also

- interaction_graph
- enzsub_graph
- giant_component

Examples

```
interactions <- import_omnipath_interactions()
graph <- interaction_graph(interactions)
paths <- find_all_paths(
    graph = graph,
    start = c('EGFR', 'STAT3'),
    end = c('AKT1', 'ULK1'),
    attr = 'name'
)
```

---

get_annotation_resources

Retrieves a list of available resources in the annotations database of OmniPath

---

Description

Get the names of the resources from https://omnipath.org/annotations.

Usage

```
get_annotation_resources(dataset = NULL, ...)
```

Arguments

dataset	ignored for this query type
...	optional additional arguments

Value

character vector with the names of the annotation resources

See Also

- get_resources
- import_omnipath_annotations

Examples

```
get_annotation_resources()
```

---

get_complex_genes *Get all the molecular complexes for a given gene(s)*

Description

This function returns all the molecular complexes where an input set of genes participate. User can choose to retrieve every complex where any of the input genes participate or just retrieve these complexes where all the genes in input set participate together.

Usage

```
get_complex_genes(
  complexes = import_omnipath_complexes(),
  select_genes,
  total_match = FALSE
)
```

Arguments

complexes	complexes data frame (obtained using import_omnipath_complexes)
select_genes	vector containing the genes for whom complexes will be retrieved (hgnc format).
total_match	[default=FALSE] logical indicating if the user wants to get all the complexes where any of the input genes participate (FALSE) or to get only the complexes where all the input genes participate together (TRUE).

Value

Data frame of complexes

See Also

import_omnipath_complexes

Examples

```
complexes <- import_omnipath_complexes(
    filter_databases = c("CORUM", "hu.MAP")
)
query_genes <- c("LMNA", "BANF1")
complexes_query_genes <- get_complex_genes(complexes, query_genes)
```

---

get_complex_resources *Retrieve a list of complex resources available in Omnipath*

Description

get the names of the resources from https://omnipath.org/complexes

Usage

```
get_complex_resources(dataset = NULL)
```

Arguments

dataset	ignored for this query type

Value

character vector with the names of the databases

See Also

- get_resources
- import_omnipath_complexes

Examples

```
get_complex_resources()
```

get_enzsub_resources *Retrieves a list of enzyme-substrate resources available in OmniPath*

Description

Get the names of the enzyme-substrate relationship resources available in https://omnipath.org/enzsub

Usage

```
get_enzsub_resources(dataset = NULL)
```

Arguments

dataset	ignored for this query type

Value

character vector with the names of the enzyme-substrate resources

See Also

- get_resources
- import_omnipath_enzsub

Examples

```
get_enzsub_resources()
```

get_interaction_resources *Retrieve a list of interaction resources available in Omnipath*

Description

Gets the names of the resources from https://omnipath.org/interactions.

Usage

```
get_interaction_resources(dataset = NULL)
```

Arguments

dataset	a dataset within the interactions query type. Currently available datasets are 'omnipath', 'kinaseextra', 'pathwayextra', 'ligrecextra', 'dorothea', 'tf_target', 'tf_mirna', 'mirnatarget' and 'lncrna_mrna'

Value

character vector with the names of the interaction databases

See Also

- get_resources
- import_all_interactions
- import_omnipath_interactions
- import_pathwayextra_interactions
- import_kinaseextra_interactions
- import_ligrecextra_interactions
- import_mirnatarget_interactions
- import_dorothea_interactions

Examples

```
get_interaction_resources()
```

---

get_intercell_categories

Retrieves a list of categories from the intercell database of OmniPath

---

Description

Retrieves a list of categories from https://omnipath.org/intercell.

Usage

```
get_intercell_categories()
```

Value

character vector with the different intercell categories

See Also

- import_omnipath_intercell
- get_intercell_generic_categories

Examples

```
get_intercell_categories()
```

get_intercell_generic_categories

Retrieves a list of the generic categories in the intercell database of OmniPath

Description

Retrieves a list of the generic categories from https://omnipath.org/intercell.

Usage

```
get_intercell_generic_categories()
```

Value

character vector with the different intercell main classes

See Also

- import_omnipath_intercell
- get_intercell_categories

Examples

```
get_intercell_generic_categories()
```

get_intercell_resources

Retrieves a list of intercellular communication resources available in OmniPath

Description

Retrieves a list of the databases from https://omnipath.org/intercell.

Usage

```
get_intercell_resources(dataset = NULL)
```

Arguments

dataset	ignored at this query type

Value

character vector with the names of the databases

See Also

- get_resources
- import_omnipath_intercell

Examples

```
get_intercell_resources()
```

`get_resources` *Retrieve the available resources for a given query type*

Description

Collects the names of the resources available in OmniPath for a certain query type and optionally for a dataset within that.

Usage

```
get_resources(query_type, datasets = NULL, generic_categories = NULL)
```

Arguments

`query_type`	one of the query types ‘interactions‘, ‘enz_sub‘, ‘complexes‘, ‘annotations‘ or ‘intercell‘
`datasets`	currently within the ‘interactions‘ query type only, multiple datasets are available: ‘omnipath‘, ‘kinaseextra‘, ‘pathwayextra‘, ‘ligrecextra‘, ‘dorothea‘, ‘tf_target‘, ‘tf_mirna‘, ‘mirnatarget‘ and ‘lncrna_mrna‘.
`generic_categories`	for the ‘intercell‘ query type, restrict the search for some generic categories e.g. ‘ligand‘ or ‘receptor‘.

Value

a character vector with resource names

Examples

```
get_resources(query_type = 'interactions')
```

get_signed_ptms *Signs for enzyme-substrate interactions*

Description

Enzyme-substrate data does not contain sign (activation/inhibition), we generate this information based on the interaction network.

Usage

```
get_signed_ptms(
  enzsub = import_omnipath_enzsub(),
  interactions = import_omnipath_interactions()
)
```

Arguments

enzsub	Enzyme-substrate data frame generated by import_omnipath_enzsub
interactions	interaction data frame generated by import_omnipath_interactions

Value

Data frame of enzyme-substrate relationships with is_inhibition and is_stimulation columns.

See Also

- import_omnipath_enzsub
- import_omnipath_interactions

Examples

```
enzsub <- import_omnipath_enzsub(resources = c('PhosphoSite', 'SIGNOR'))
interactions <- import_omnipath_interactions()
enzsub <- get_signed_ptms(enzsub, interactions)
```

giant_component *Giant component of a graph*

Description

For an igraph graph object returns its giant component.

Usage

```
giant_component(graph)
```

Arguments

graph	An igraph graph object.

Value

An igraph graph object containing only the giant component.

Examples

```
interactions <- import_post_translational_interactions()
graph <- interaction_graph(interactions)
graph_gc <- giant_component(graph)
```

---

go_annot_download *Downloads gene annotations from Gene Ontology*

Description

Gene Ontology is an ontology of gene subcellular localizations, molecular functions and involvement in biological processes. Gene products across many organisms are annotated with the ontology terms. This function downloads the gene-ontology term associations for certain model organisms or all organisms. For a description of the columns see http://geneontology.org/docs/go-annotation-file-gaf-format-2.2/.

Usage

```
go_annot_download(organism = "human")
```

Arguments

organism	Character: either "chicken", "cow", "dog", "human", "pig" or "uniprot_all".

Value

A tibble (data frame) of annotations as it is provided by the database

Examples

```
goa_data <- go_annot_download()
goa_data
# # A tibble: 606,840 x 17
#    db       db_object_id db_object_symbol qualifier go_id   db_ref
#    <fct>    <chr>        <chr>            <fct>     <chr>   <chr>
#  1 UniProt. A0A024RBG1   NUDT4B           NA        GO:000. GO_REF:00.
#  2 UniProt. A0A024RBG1   NUDT4B           NA        GO:000. GO_REF:00.
#  3 UniProt. A0A024RBG1   NUDT4B           NA        GO:004. GO_REF:00.
#  4 UniProt. A0A024RBG1   NUDT4B           NA        GO:005. GO_REF:00.
#  5 UniProt. A0A024RBG1   NUDT4B           NA        GO:005. GO_REF:00.
```

```
# # . with 606,830 more rows, and 11 more variables:
# #   evidence_code <fct>, with_or_from <chr>, aspect <fct>,
# #   db_object_name <chr>, db_object_synonym <chr>,
# #   db_object_type <fct>, taxon <fct>, date <date>,
# #   assigned_by <fct>, annotation_extension <chr>,
# #   gene_product_from_id <chr>
```

`guide2pharma_download` *Downloads interactions from the Guide to Pharmacology database*

Description

Downloads ligand-receptor interactions from the Guide to Pharmacology (IUPHAR/BPS) database (https://www.guidetopharmacology.org/).

Usage

```
guide2pharma_download()
```

Value

A tibble (data frame) of interactions as it is provided by the database

Examples

```
g2p_data <- guide2pharma_download()
g2p_data
# # A tibble: 21,586 x 38
#    target target_id target_gene_sym. target_uniprot target_ensembl_.
#    <chr>      <dbl> <chr>            <chr>          <chr>
#  1 12S-L.      1387 ALOX12           P18054         ENSG00000108839
#  2 15-LO.      1388 ALOX15           P16050         ENSG00000161905
#  3 15-LO.      1388 ALOX15           P16050         ENSG00000161905
#  4 15-LO.      1388 ALOX15           P16050         ENSG00000161905
# # . with 21,576 more rows, and 33 more variables: target_ligand <chr>,
# #   target_ligand_id <chr>, target_ligand_gene_symbol <chr>,
# ... (truncated)
```

harmonizome_download *Downloads a Harmonizome network dataset*

Description

Downloads a single network dataset from Harmonizome https://maayanlab.cloud/Harmonizome.

Usage

```
harmonizome_download(dataset)
```

Arguments

dataset	The dataset part of the URL. Please refer to the download section of the Harmonizome webpage.

Value

Data frame (tibble) with interactions.

Examples

```
harmonizome_data <- harmonizome_download('phosphositeplus')
harmonizome_data
# # A tibble: 6,013 x 7
#    source   source_desc source_id target target_desc target_id weight
#    <chr>    <chr>           <dbl> <chr>  <chr>           <dbl>  <dbl>
#  1 TP53     na               7157 STK17A na               9263      1
#  2 TP53     na               7157 TP53RK na             112858      1
#  3 TP53     na               7157 SMG1   na              23049      1
#  4 UPF1     na               5976 SMG1   na              23049      1
# # . with 6,003 more rows
```

hpo_download *Downloads protein annotations from Human Phenotype Ontology*

Description

Human Phenotype Ontology (HPO) provides a standardized vocabulary of phenotypic abnormalities encountered in human disease. Each term in the HPO describes a phenotypic abnormality. HPO currently contains over 13,000 terms and over 156,000 annotations to hereditary diseases. See more at https://hpo.jax.org/app/.

Usage

```
hpo_download()
```

Value

A tibble (data frame) of annotations as it is provided by the database

Examples

```
hpo_data <- hpo_download()
hpo_data
# # A tibble: 231,738 x 9
#    entrez_gene_id entrez_gene_symb. hpo_term_id hpo_term_name
#             <dbl> <chr>             <chr>       <chr>
#  1           8192 CLPP              HP:0000013  Hypoplasia of the ute.
#  2           8192 CLPP              HP:0004322  Short stature
#  3           8192 CLPP              HP:0000786  Primary amenorrhea
#  4           8192 CLPP              HP:0000007  Autosomal recessive i.
#  5           8192 CLPP              HP:0000815  Hypergonadotropic hyp.
# # . with 231,733 more rows, and 5 more variables:
# #   frequency_raw <chr>, frequency_hpo <chr>, info_gd_source <chr>,
# #   gd_source <chr>, disease_id <chr>
```

`htridb_download` *Downloads TF-target interactions from HTRIdb*

Description

HTRIdb (https://www.lbbc.ibb.unesp.br/htri/) is a database of literature curated human TF-target interactions. As the database is recently offline, the data is distributed by the OmniPath rescued data repository (https://rescued.omnipathdb.org/).

Usage

```
htridb_download()
```

Value

Data frame (tibble) with interactions.

Examples

```
htridb_data <- htridb_download()
htridb_data
# # A tibble: 18,630 x 7
#      OID GENEID_TF SYMBOL_TF GENEID_TG SYMBOL_TG TECHNIQUE
#    <dbl>     <dbl> <chr>         <dbl> <chr>     <chr>
#  1 32399       142 PARP1           675 BRCA2     Electrophoretic Mobi.
#  2 32399       142 PARP1           675 BRCA2     Chromatin Immunoprec.
#  3 28907       196 AHR            1543 CYP1A1    Chromatin Immunoprec.
#  4 29466       196 AHR            1543 CYP1A1    Electrophoretic Mobi.
#  5 28911       196 AHR            1543 CYP1A1    Chromatin Immunoprec.
```

```
# # . with 18,620 more rows, and 1 more variable: PUBMED_ID <chr>
```

`import_all_interactions` *Imports all interaction datasets available in OmniPath*

Description

The interaction datasets currently available in OmniPath:

Usage

```
import_all_interactions(
  resources = NULL,
  organism = 9606,
  dorothea_levels = c("A", "B"),
  exclude = NULL,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)

import_AllInteractions(...)
```

Arguments

`resources`	interactions not reported in these databases are removed. See `get_interaction_resources` for more information.
`organism`	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
`dorothea_levels`	The confidence levels of the dorothea interactions (TF-target) which range from A to D. Set to A and B by default.
`exclude`	datasets to exclude
`fields`	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
`default_fields`	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
`references_by_resource`	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
`...`	Passed to `import_all_interactions`.

Details

omnipath: the OmniPath data as defined in the paper, an arbitrary optimum between coverage and quality pathwayextra: activity flow interactions without literature reference kinaseextra: enzyme-substrate interactions without literature reference ligrecextra: ligand-receptor interactions without literature reference dorothea: transcription factor (TF)-target interactions from DoRothEA tf_target: transcription factor (TF)-target interactions from other resources mirnatarget: miRNA-mRNA interactions tf_mirna: TF-miRNA interactions lncrna_mrna: lncRNA-mRNA interactions

Value

A dataframe containing all the datasets in the interactions query

See Also

- get_interaction_resources
- interaction_graph
- print_interactions

Examples

```
interactions <- import_all_interactions(
    resources = c('HPRD', 'BioGRID'),
    organism = 9606
)
```

---

import_dorothea_interactions

From the OmniPath webservice imports interactions from the DoRothEA dataset

---

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=dorothea which contains transcription factor (TF)-target interactions from DoRothEA https://github.com/saezlab/DoRothEA

Usage

```
import_dorothea_interactions(
  resources = NULL,
  organism = 9606,
  dorothea_levels = c("A", "B"),
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	interactions not reported in these databases are removed. See get_interaction_resources for more information.
organism	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
dorothea_levels	Vector detailing the confidence levels of the interactions to be downloaded. In dorothea, every TF-target interaction has a confidence score ranging from A to E, being A the most reliable interactions. By default we take A and B level interactions (c(A,B)). It is to note that E interactions are not available in OmnipathR.
fields	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
default_fields	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
references_by_resource	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
...	optional additional arguments

Value

A dataframe containing TF-target interactions from DoRothEA

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions <- import_dorothea_interactions(
    resources = c('DoRothEA', 'ARACNe-GTEx_DoRothEA'),
    organism = 9606,
    dorothea_levels = c('A', 'B', 'C')
)
```

`import_intercell_network` *Imports an intercellular network combining annotations and interactions*

Description

Imports an intercellular network by mapping intercellular annotations and protein interactions. First imports a network of protein-protein interactions. Then, it retrieves annotations about the proteins intercellular communication roles, once for the transmitter (delivering information from the expressing cell) and second, the receiver (receiving signal and relaying it towards the expressing cell) side. These 3 queries can be customized by providing parameters in lists which will be passed to the respective methods (`import_omnipath_interactions` for the network and `import_omnipath_intercell` for the annotations). Finally the 3 data frames combined in a way that the source proteins in each interaction annotated by the transmitter, and the target proteins by the receiver categories. If undirected interactions present (these are disabled by default) they will be duplicated, i.e. both partners can be both receiver and transmitter. If a cache file provided, its content will be returned without any further filtering.

Usage

```
import_intercell_network(
  interactions_param = list(),
  transmitter_param = list(),
  receiver_param = list(),
  resources = NULL,
  entity_types = NULL,
  ligand_receptor = FALSE,
  ...
)
```

Arguments

`interactions_param`
: a list with arguments for an interactions query: `import_omnipath_interactions`, `import_pathwayextra_interactions`, `import_kinaseextra_interactions`, `import_ligrecextra_interactions`

`transmitter_param`
: a list with arguments for `import_omnipath_intercell`, to define the transmitter side of intercellular connections

`receiver_param`
: a list with arguments for `import_omnipath_intercell`, to define the receiver side of intercellular connections

`resources`
: A character vector of resources to be applied to both the interactions and the annotations. For example, `resources = 'CellChatDB'` will download the transmitters and receivers defined by CellChatDB, connected by connections from CellChatDB.

entity_types	Character, possible values are "protein", "complex" or both.
ligand_receptor	Logical. If TRUE, only *ligand* and *receptor* annotations will be used instead of the more generic *transmitter* and *receiver* categories.
...	Ignored.

Value

A dataframe containing information about protein-protein interactions and the inter-cellular roles of the protiens involved in those interactions.

See Also

- get_intercell_categories
- get_intercell_generic_categories
- import_omnipath_intercell
- import_omnipath_interactions
- import_pathwayextra_interactions
- import_kinaseextra_interactions
- import_ligrecextra_interactions

Examples

```
intercellNetwork <- import_intercell_network(
   interactions_param = list(datasets = 'ligrecextra'),
   receiver_param = list(categories = c('receptor', 'transporter')),
   transmitter_param = list(categories = c('ligand', 'secreted_enzyme'))
)
```

---

import_kinaseextra_interactions

Imports interactions from the 'kinase extra' dataset of OmniPath

---

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=kinaseextra, which contains enzyme-substrate interactions without literature reference. The enzyme-substrate interactions supported by literature references are part of the 'omnipath' dataset.

Usage

```
import_kinaseextra_interactions(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	interactions not reported in these databases are removed. See get_interaction_resources for more information.
organism	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
fields	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
default_fields	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
references_by_resource	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
...	Optional additional arguments.

Value

A dataframe containing enzyme-substrate interactions without literature reference

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions <-
   import_kinaseextra_interactions(
       resources = c('PhosphoPoint', 'PhosphoSite'),
       organism = 9606
   )
```

`import_ligrecextra_interactions` *Imports interactions from the 'ligrec extra' dataset of OmniPath*

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=ligrecextra, which contains ligand-receptor interactions without literature reference. The ligand-receptor interactions supported by literature references are part of the 'omnipath' dataset.

Usage

```
import_ligrecextra_interactions(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

`resources`	interactions not reported in these databases are removed. See `get_interaction_resources` for more information.
`organism`	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
`fields`	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
`default_fields`	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
`references_by_resource`	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
`...`	optional additional arguments

Value

A dataframe containing ligand-receptor interactions including the ones without literature references

See Also

- `get_interaction_resources`
- `import_all_interactions`
- `interaction_graph`
- `print_interactions`

Examples

```
interactions <- import_ligrecextra_interactions(
    resources = c('HPRD', 'Guide2Pharma'),
    organism = 9606
)
```

`import_lncrna_mrna_interactions` *Imports interactions from the lncRNA-mRNA dataset of OmniPath*

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=lncrna_mrna, which contains lncRNA-mRNA interactions

Usage

```
import_lncrna_mrna_interactions(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

`resources`	interactions not reported in these databases are removed. See `get_interaction_resources` for more information.
`organism`	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
`fields`	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
`default_fields`	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
`references_by_resource`	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
`...`	optional additional arguments

Value

A dataframe containing lncRNA-mRNA interactions

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions <-
    import_lncrna_mrna_interactions(
        resources = c('ncRDeathDB')
    )
```

`import_mirnatarget_interactions` *Imports interactions from the miRNA-target dataset of OmniPath*

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=mirnatarget, which contains miRNA-mRNA interactions.

Usage

```
import_mirnatarget_interactions(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	interactions not reported in these databases are removed. See get_interaction_resources for more information.
organism	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
fields	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
default_fields	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.

references_by_resource
: if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.

... : optional additional arguments

Value

A dataframe containing miRNA-mRNA interactions

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions <-
    import_mirnatarget_interactions(
        resources = c('miRTarBase', 'miRecords')
    )
```

---

import_omnipath_annotations

Imports annotations from OmniPath

Description

Imports protein annotations about function, localization, expression, structure and other properties of proteins from OmniPath https://omnipathdb.org/annotations. Note: there might be also a few miRNAs annotated; a vast majority of protein complex annotations are inferred from the annotations of the members: if all members carry the same annotation the complex inherits.

Usage

```
import_omnipath_annotations(
  proteins = NULL,
  resources = NULL,
  wide = FALSE,
  ...
)
```

Arguments

proteins	Vector containing the genes or proteins for whom annotations will be retrieved (UniProt IDs or HGNC Gene Symbols or miRBase IDs). It is also possible to donwload annotations for protein complexes. To do so, write 'COMPLEX:' right before the genesymbols of the genes integrating the complex. Check the vignette for examples.
resources	Load the annotations only from these databases. See get_annotation_resources for possible values.
wide	Convert the annotation table to wide format, which corresponds more or less to the original resource. If the data comes from more than one resource a list of wide tables will be returned. See examples at pivot_annotations.
...	Additional arguments.

Details

Downloading the full annotations dataset is disabled by default because the size of this data is around 1GB. We recommend to retrieve the annotations for a set of proteins or only from a few resources, depending on your interest. You can always download the full database from https://archive.omnipathdb.org/omnipat using any standard R or readr method.

Value

A data.frame containing different gene/complex annotations

See Also

- get_annotation_databases
- pivot_annotations

Examples

```
annotations <- import_omnipath_annotations(
    proteins = c('TP53', 'LMNA'),
    resources = c('HPA_subcellular')
)
```

---

import_omnipath_complexes

Imports protein complexes from OmniPath

Description

Imports the complexes stored in Omnipath database from https://omnipathdb.org/complexes.

Usage

```
import_omnipath_complexes(resources = NULL, ...)
```

Arguments

resources	complexes not reported in these databases are removed. See get_complexes_databases for more information.
...	optional additional arguments

Value

A dataframe containing information about complexes

See Also

- get_complexes_databases

Examples

```
complexes = import_omnipath_complexes(
    resources = c('CORUM', 'hu.MAP')
)
```

---

import_omnipath_enzsub

Imports enzyme-substrate relationships from OmniPath

Description

Imports the enzyme-substrate (more exactly, enzyme-PTM) relationship database from https://omnipathdb.org/enzsub

Usage

```
import_omnipath_enzsub(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	PTMs not reported in these databases are removed. See get_ptms_databases for more information
organism	PTMs are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
fields	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
default_fields	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
references_by_resource	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
...	Optional additional arguments.

Value

A data frame containing the information about ptms

See Also

- get_enzsub_resources
- import_omnipath_interactions
- enzsub_graph
- print_interactions

Examples

```
enzsub <- import_omnipath_enzsub(
    resources = c('PhosphoSite', 'SIGNOR'),
    organism = 9606
)
```

---

import_omnipath_interactions

Imports interactions from the 'omnipath' dataset of Omnipath

Description

Imports the database from https://omnipathdb.org/interactions, which contains only interactions supported by literature references. This part of the interaction database compiled a similar way as it has been presented in the first paper describing OmniPath (Turei et al. 2016).

Usage

```
import_omnipath_interactions(
  resources = NULL,
  organism = 9606,
  datasets = "omnipath",
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	interactions not reported in these databases are removed. See get_interaction_resources for more information.
organism	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
datasets	Names of the interaction datasts to download: omnipath (by default). Other possiblites are: pathwayextra, kinaseextra, ligrecextra, dorothea,tf_target, mirnatarget, tf_mirna, lncrna_mrna. The user can select multiple datasets as for example: c('omnipath', 'pathwayextra', 'kinaseextra')
fields	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
default_fields	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
references_by_resource	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
...	optional additional arguments

Value

A dataframe of protein-protein interactions

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions = import_omnipath_interactions(
    resources = c('SignaLink3'),
    organism = 9606
)
```

`import_omnipath_intercell` *Imports OmniPath intercell annotations*

Description

Imports the OmniPath intercellular communication role annotation database from https://omnipathdb.org/intercell. It provides information on the roles in inter-cellular signaling. E.g. if a protein is a ligand, a receptor, an extracellular matrix (ECM) component, etc.

Usage

```
import_omnipath_intercell(
  categories = NULL,
  resources = NULL,
  parent = NULL,
  scope = NULL,
  aspect = NULL,
  source = NULL,
  transmitter = NULL,
  receiver = NULL,
  secreted = NULL,
  plasma_membrane_peripheral = NULL,
  plasma_membrane_transmembrane = NULL,
  proteins = NULL,
  topology = NULL,
  causality = NULL,
  ...
)
```

Arguments

`categories`	vector containing the categories to be retrieved. All the genes belonging to those categories will be returned. For further information about the categories see `get_intercell_categories`
`resources`	limit the query to certain resources; see the available resources by `get_intercell_resources`
`parent`	vector containing the parent classes to be retrieved. All the genes belonging to those classes will be returned. For furter information about the main classes see `get_intercell_categories`

scope	either ‘specific‘ or ‘generic‘
aspect	either ‘locational‘ or ‘functional‘
source	either ‘resource_specific‘ or ‘composite‘
transmitter	logical, include only transmitters i.e. proteins delivering signal from a cell to its environment
receiver	logical, include only receivers i.e. proteins delivering signal to the cell from its environment
secreted	logical, include only secreted proteins
plasma_membrane_peripheral	logical, include only plasma membrane peripheral membrane proteins
plasma_membrane_transmembrane	logical, include only plasma membrane transmembrane proteins
proteins	limit the query to certain proteins
topology	topology categories: one or more of ‘secreted‘ (sec), ‘plasma_membrane_peripheral‘ (pmp), ‘plasma_membrane_transmembrane‘ (pmtm) (both short or long notation can be used)
causality	‘transmitter‘ (trans), ‘receiver‘ (rec) or ‘both‘ (both short or long notation can be used)
...	Additional optional arguments

Value

A dataframe cotaining information about roles in intercellular signaling.

See Also

- get_intercell_categories
- get_intercell_generic_categories
- import_intercell_network

Examples

```
intercell <- import_omnipath_intercell(categories = 'ecm')
```

---

import_pathwayextra_interactions

Imports interactions from the ‘pathway extra‘ dataset of Omnipath

---

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=pathwayextra, which contains activity flow interactions without literature reference. The activity flow interactions supported by literature references are part of the ‘omnipath‘ dataset.

Usage

```
import_pathwayextra_interactions(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	interactions not reported in these databases are removed. See get_interaction_resources for more information.
organism	Interactions are available for human, mouse and rat. Choose one of those: 9606 human (default), 10116 rat or 10090 Mouse.
fields	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
default_fields	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
references_by_resource	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
...	optional additional arguments

Value

A dataframe containing activity flow interactions between proteins without literature reference

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions <-
    import_pathwayextra_interactions(
        resources = c('BioGRID', 'IntAct'),
        organism = 9606
    )
```

`import_post_translational_interactions` *Imports all post-translational interactions from OmniPath*

Description

Imports the dataset from all post-translational datasets of OmniPath.

Usage

```
import_post_translational_interactions(
  resources = NULL,
  organism = 9606,
  exclude = NULL,
  references_by_resource = TRUE,
  ...
)
```

Arguments

`resources`	interactions not reported in these databases are removed. See `get_interaction_resources` for more information.
`organism`	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
`exclude`	datasets to exclude
`references_by_resource`	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
`...`	optional additional arguments

Value

A dataframe containing post-translational interactions

See Also

- `get_interaction_resources`
- `import_all_interactions`
- `interaction_graph`
- `print_interactions`

Examples

```
interactions <-
    import_post_translational_interactions(
        resources = c('BioGRID')
    )
```

`import_tf_mirna_interactions` *Imports interactions from the TF-miRNA dataset of OmniPath*

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=tf_mirna, which contains transcription factor-miRNA gene interactions

Usage

```
import_tf_mirna_interactions(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

`resources`	interactions not reported in these databases are removed. See `get_interaction_resources` for more information.
`organism`	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
`fields`	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
`default_fields`	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.
`references_by_resource`	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
`...`	optional additional arguments

Value

A dataframe containing TF-miRNA interactions

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions <-
    import_tf_mirna_interactions(
        resources = c('TransmiR')
    )
```

`import_tf_target_interactions` *Imports interactions from the TF-target dataset of OmniPath*

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=tf_target, which contains transcription factor-target protein coding gene interactions. Note: this is not the only TF-target dataset in OmniPath, 'dorothea' is the other one and the 'tf_mirna' dataset provides TF-miRNA gene interactions.

Usage

```
import_tf_target_interactions(
  resources = NULL,
  organism = 9606,
  fields = NULL,
  default_fields = TRUE,
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	interactions not reported in these databases are removed. See get_interaction_resources for more information.
organism	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
fields	The user can define here the fields to be added. If used, set the next argument, 'default_fields', to FALSE.
default_fields	whether to include the default fields (columns) for the query type. If FALSE, only the fields defined by the user in the 'fields' argument will be added.

`references_by_resource`
: if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.

`...`
: Optional additional arguments

Value

A dataframe containing TF-target interactions

See Also

- `get_interaction_resources`
- `import_all_interactions`
- `interaction_graph`
- `print_interactions`

Examples

```
interactions <-
    import_tf_target_interactions(
        resources = c('DoRothEA', 'SIGNOR')
    )
```

---

import_transcriptional_interactions

Imports all TF-target interactions from OmniPath

Description

Imports the dataset from: https://omnipathdb.org/interactions?datasets=tf_target,dorothea, which contains transcription factor-target protein coding gene interactions.

Usage

```
import_transcriptional_interactions(
  resources = NULL,
  organism = 9606,
  dorothea_levels = c("A", "B"),
  references_by_resource = TRUE,
  ...
)
```

Arguments

resources	interactions not reported in these databases are removed. See get_interaction_resources for more information.
organism	Interactions are available for human, mouse and rat. Choose among: 9606 human (default), 10116 rat and 10090 Mouse
dorothea_levels	Vector detailing the confidence levels of the interactions to be downloaded. In dorothea, every TF-target interaction has a confidence score ranging from A to E, being A the most reliable interactions. By default we take A and B level interactions (c(A,B)). It is to note that E interactions are not available in OmnipathR.
references_by_resource	if FALSE, removes the resource name prefixes from the references (PubMed IDs); this way the information which reference comes from which resource will be lost and the PubMed IDs will be unique.
...	optional additional arguments

Value

A dataframe containing TF-target interactions

See Also

- get_interaction_resources
- import_all_interactions
- interaction_graph
- print_interactions

Examples

```
interactions <-
    import_transcriptional_interactions(
        resources = c('PAZAR', 'ORegAnno', 'DoRothEA')
    )
```

---

inbiomap_download *Downloads and preprocesses network data from InWeb InBioMap*

Description

Downloads the data by inbiomap_raw, extracts the UniProt IDs, Gene Symbols and scores and removes the irrelevant columns.

Usage

```
inbiomap_download(...)
```

Arguments

...	Passed to inbiomap_raw.

Value

A data frame (tibble) of interactions.

See Also

inbiomap_raw

Examples

```
inbiomap_interactions <- inbiomap_download()
inbiomap_interactions
# # A tibble: 625,641 x 7
#    uniprot_a uniprot_b genesymbol_a genesymbol_b inferred score1 score2
#    <chr>     <chr>     <chr>        <chr>        <lgl>     <dbl>  <dbl>
#  1 A0A5B9    P01892    TRBC2        HLA-A        FALSE     0.417 0.458
#  2 A0AUZ9    Q96CV9    KANSL1L      OPTN         FALSE     0.155 0.0761
#  3 A0AV02    P24941    SLC12A8      CDK2         TRUE      0.156 0.0783
#  4 A0AV02    Q00526    SLC12A8      CDK3         TRUE      0.157 0.0821
#  5 A0AV96    P0CG48    RBM47        UBC          FALSE     0.144 0.0494
# # . with 625,631 more rows
```

---

inbiomap_raw *Downloads network data from InWeb InBioMap*

Description

Downloads the data from https://inbio-discover.com/map.html#downloads in tar.gz format, extracts the PSI MITAB table and returns it as a data frame.

Usage

```
inbiomap_raw(curl_verbose = FALSE)
```

Arguments

curl_verbose	Logical. Perform CURL requests in verbose mode for debugging purposes.

Value

A data frame (tibble) with the extracted interaction table.

See Also

inbiomap_download

Examples

```
inbiomap_psimitab <- inbiomap_raw()
```

`interaction_graph` *Build Omnipath interaction graph*

Description

Transforms the interactions data frame to an igraph graph object.

Usage

```
interaction_graph(interactions = interactions)
```

Arguments

`interactions`	data.frame created by • `import_omnipath_enzsub` • `import_omnipath_interactions` • `import_pathwayextra_interactions` • `import_kinaseextra_interactions` • `import_ligrecextra_interactions` • `import_post_translational_interactions` • `import_dorothea_interactions` • `import_tf_target_interactions` • `import_transcriptional_interactions` • `import_mirnatarget_interactions` • `import_all_interactions`

Value

An igraph graph object.

See Also

- `import_omnipath_interactions`
- `import_pathwayextra_interactions`
- `import_kinaseextra_interactions`
- `import_ligrecextra_interactions`
- `import_dorothea_interactions`
- `import_mirnatarget_interactions`
- `import_all_interactions`
- `giant_component`
- `find_all_paths`

Examples

```
interactions <- import_omnipath_interactions(resources = c('SignaLink3'))
g <- interaction_graph(interactions)
```

`nichenet_build_model` *Construct a NicheNet ligand-target model*

Description

Construct a NicheNet ligand-target model

Usage

```
nichenet_build_model(optimization_results, networks, weighted = TRUE)
```

Arguments

`optimization_results`	The outcome of NicheNet parameter optimization as produced by `nichenet_optimization`.
`networks`	A list with NicheNet format signaling, ligand-receptor and gene regulatory networks as produced by `nichenet_networks`.
`weighted`	Logical: whether to use the optimized weights.

Value

A named list with two elements: ‘weighted_networks‘ and ‘optimized_parameters‘.

Examples

```
networks <- nichenet_networks()
expression <- nichenet_expression_data()

optimization_results <- nichenet_optimization(networks, expression)
nichenet_model <- nichenet_build_model(optimization_results, networks)
```

nichenet_expression_data *Expression data from ligand-receptor perturbation experiments used by NicheNet*

Description

NicheNet uses expression data from a collection of published ligand or receptor KO or perturbation experiments to build its model. This function retrieves the original expression data, deposited in Zenodo (https://zenodo.org/record/3260758).

Usage

```
nichenet_expression_data()
```

Value

Nested list, each element contains a data frame of processed expression data and key variables about the experiment.

Examples

```
exp_data <- nichenet_expression_data()
head(names(exp_data))
# [1] "bmp4_tgfb"     "tgfb_bmp4"     "nodal_Nodal"   "spectrum_Il4"
# [5] "spectrum_Tnf"  "spectrum_Ifng"
purrr::map_chr(head(exp_data), 'from')
#     bmp4_tgfb     tgfb_bmp4   nodal_Nodal  spectrum_Il4  spectrum_Tnf
#       "BMP4"       "TGFB1"       "NODAL"         "IL4"         "TNF"
# spectrum_Ifng
#       "IFNG"
```

nichenet_gr_network *Builds a NicheNet gene regulatory network*

Description

Builds gene regulatory network prior knowledge for NicheNet using multiple resources.

Usage

```
nichenet_gr_network(
  omnipath = list(),
  harmonizome = list(),
  regnetwork = list(),
  htridb = list(),
  remap = list(),
  evex = list(),
  pathwaycommons = list(),
  trrust = list(),
  only_omnipath = FALSE
)
```

Arguments

omnipath	List with paramaters to be passed to nichenet_gr_network_omnipath.
harmonizome	List with paramaters to be passed to nichenet_gr_network_harmonizome.
regnetwork	List with paramaters to be passed to nichenet_gr_network_regnetwork.
htridb	List with paramaters to be passed to nichenet_gr_network_htridb.
remap	List with paramaters to be passed to nichenet_gr_network_remap.
evex	List with paramaters to be passed to nichenet_gr_network_evex.
pathwaycommons	List with paramaters to be passed to nichenet_gr_network_pathwaycommons.
trrust	List with paramaters to be passed to nichenet_gr_network_trrust.
only_omnipath	Logical: a shortcut to use only OmniPath as network resource.

Value

A network data frame (tibble) with gene regulatory interactions suitable for use with NicheNet.

See Also

- nichenet_gr_network_evex
- nichenet_gr_network_htridb
- nichenet_gr_network_omnipath
- nichenet_gr_network_pathwaycommons
- nichenet_gr_network_regnetwork
- nichenet_gr_network_remap
- nichenet_gr_network_trrust

Examples

```
# load everything with the default parameters:
gr_network <- nichenet_gr_network()

# less targets from ReMap, not using RegNetwork:
```

```
gr_network <- nichenet_gr_network(
    remap = list(top_targets = 200),
    regnetwork = NULL,
)


# use only OmniPath:
gr_network_omnipath <- nichenet_gr_network(only_omnipath = TRUE)
```

`nichenet_gr_network_evex` *NicheNet gene regulatory network from EVEX*

Description

Builds a gene regulatory network using data from the EVEX database and converts it to a format suitable for NicheNet.

Usage

```
nichenet_gr_network_evex(
  top_confidence = 0.75,
  indirect = FALSE,
  regulation_of_expression = FALSE
)
```

Arguments

`top_confidence`	Double, between 0 and 1. Threshold based on the quantile of the confidence score.
`indirect`	Logical: whether to include indirect interactions.
`regulation_of_expression`	Logical: whether to include also the "regulation of expression" type interactions.

Value

Data frame of interactions in NicheNet format.

Data frame with gene regulatory interactions in NicheNet format.

See Also

- `nichenet_gr_network`
- `evex_download`

Examples

```
# use only the 10% with the highest confidence:
evex_gr_network <- nichenet_gr_network_evex(top_confidence = .9)
```

`nichenet_gr_network_harmonizome` *NicheNet gene regulatory network from Harmonizome*

Description

Builds gene regulatory network prior knowledge for NicheNet using Harmonizome

Usage

```
nichenet_gr_network_harmonizome(
  datasets = c("cheappi", "encodetfppi", "jasparpwm", "transfac", "transfacpwm",
    "motifmap", "geotf", "geokinase", "geogene"),
  ...
)
```

Arguments

`datasets`	The datasets to use. For possible values please refer to default value and the Harmonizome webpage.
`...`	Ignored.

Value

Data frame with gene regulatory interactions in NicheNet format.

See Also

- `nichenet_gr_network`
- `harmonizome_download`

Examples

```
# use only JASPAR and TRANSFAC:
hz_gr_network <- nichenet_gr_network_harmonizome(
    datasets = c('jasparpwm', 'transfac', 'transfacpwm')
)
```

nichenet_gr_network_htridb

NicheNet gene regulatory network from HTRIdb

Description

Builds a gene regulatory network using data from the HTRIdb database and converts it to a format suitable for NicheNet.

Usage

```
nichenet_gr_network_htridb()
```

Value

Data frame with gene regulatory interactions in NicheNet format.

See Also

htridb_download, nichenet_gr_network

Examples

```
htri_gr_network <- nichenet_gr_network_htridb()
```

nichenet_gr_network_omnipath

Builds gene regulatory network for NicheNet using OmniPath

Description

Retrieves network prior knowledge from OmniPath and provides it in a format suitable for NicheNet. This method never downloads the 'ligrecextra' dataset because the ligand-receptor interactions are supposed to come from nichenet_lr_network_omnipath.

Usage

```
nichenet_gr_network_omnipath(min_curation_effort = 0, ...)
```

Arguments

min_curation_effort	Lower threshold for curation effort
...	Passed to import_transcriptional_interactions

Value

A network data frame (tibble) with gene regulatory interactions suitable for use with NicheNet.

See Also

- nichenet_gr_network_evex
- nichenet_gr_network_harmonizome
- nichenet_gr_network_htridb
- nichenet_gr_network_omnipath
- nichenet_gr_network_pathwaycommons
- nichenet_gr_network_regnetwork
- nichenet_gr_network_remap
- nichenet_gr_network_trrust

Examples

```
# use interactions up to confidence level "C" from DoRothEA:
op_gr_network <- nichenet_gr_network_omnipath(
    dorothea_levels = c('A', 'B', 'C')
)
```

---

nichenet_gr_network_pathwaycommons

NicheNet gene regulatory network from PathwayCommons

Description

Builds gene regulation prior knowledge for NicheNet using PathwayCommons.

Usage

```
nichenet_gr_network_pathwaycommons(
  interaction_types = "controls-expression-of",
  ...
)
```

Arguments

interaction_types	Character vector with PathwayCommons interaction types. Please refer to the default value and the PathwayCommons webpage.
...	Ignored.

Value

Data frame with gene regulatory interactions in NicheNet format.

See Also

- nichenet_gr_network
- pathwaycommons_download

Examples

```
pc_gr_network <- nichenet_gr_network_pathwaycommons()
```

---

nichenet_gr_network_regnetwork

NicheNet gene regulatory network from RegNetwork

Description

Builds a gene regulatory network using data from the RegNetwork database and converts it to a format suitable for NicheNet.

Usage

```
nichenet_gr_network_regnetwork()
```

Value

Data frame with gene regulatory interactions in NicheNet format.

See Also

- regnetwork_download
- nichenet_gr_network

Examples

```
regn_gr_network <- nichenet_gr_network_regnetwork()
```

nichenet_gr_network_remap *NicheNet gene regulatory network from ReMap*

Description

Builds a gene regulatory network using data from the ReMap database and converts it to a format suitable for NicheNet.

Usage

```
nichenet_gr_network_remap(
  score = 100,
  top_targets = 500,
  only_known_tfs = TRUE
)
```

Arguments

score	Numeric: a minimum score between 0 and 1000, records with lower scores will be excluded. If NULL no filtering performed.
top_targets	Numeric: the number of top scoring targets for each TF. Essentially the maximum number of targets per TF. If NULL the number of targets is not restricted.
only_known_tfs	Logical: whether to exclude TFs which are not in TF census.

Value

Data frame with gene regulatory interactions in NicheNet format.

See Also

- remap_filtered
- nichenet_gr_network

Examples

```
# use only max. top 100 targets for each TF:
remap_gr_network <- nichenet_gr_network_remap(top_targets = 100)
```

nichenet_gr_network_trrust

NicheNet gene regulatory network from TRRUST

Description

Builds a gene regulatory network using data from the TRRUST database and converts it to a format suitable for NicheNet.

Usage

```
nichenet_gr_network_trrust()
```

Value

Data frame with gene regulatory interactions in NicheNet format.

See Also

- trrust_download
- nichenet_gr_network

Examples

```
trrust_gr_network <- nichenet_gr_network_trrust()
```

nichenet_ligand_activities

Calls the NicheNet ligand activity analysis

Description

Calls the NicheNet ligand activity analysis

Usage

```
nichenet_ligand_activities(
  ligand_target_matrix,
  lr_network,
  expressed_genes_transmitter,
  expressed_genes_receiver,
  genes_of_interest,
  background_genes = NULL,
  n_top_ligands = 42,
  n_top_targets = 250
)
```

Arguments

ligand_target_matrix	A matrix with rows and columns corresponding to ligands and targets, respectively. Produced by nichenet_ligand_target_matrix or nichenetr::construct_ligand_target_ma
lr_network	A data frame with ligand-receptor interactions, as produced by nichenet_lr_network.
expressed_genes_transmitter	Character vector with the gene symbols of the genes expressed in the cells transmitting the signal.
expressed_genes_receiver	Character vector with the gene symbols of the genes expressed in the cells receiving the signal.
genes_of_interest	Character vector with the gene symbols of the genes of interest. These are the genes in the receiver cell population that are potentially affected by ligands expressed by interacting cells (e.g. genes differentially expressed upon cell-cell interaction).
background_genes	Character vector with the gene symbols of the genes to be used as background.
n_top_ligands	How many of the top ligands to include in the ligand-target table.
n_top_targets	For each ligand, how many of the top targets to include in the ligand-target table.

Value

A named list with 'ligand_activities' (a tibble giving several ligand activity scores; following columns in the tibble: $test_ligand, $auroc, $aupr and $pearson) and 'ligand_target_links' (a tibble with columns ligand, target and weight (i.e. regulatory potential score)).

Examples

```
networks <- nichenet_networks()
expression <- nichenet_expression_data()
optimization_results <- nichenet_optimization(networks, expression)
nichenet_model <- nichenet_build_model(optimization_results, networks)
lt_matrix <- nichenet_ligand_target_matrix(
    nichenet_model$weighted_networks,
    networks$lr_network,
    nichenet_model$optimized_parameters
)
ligand_activities <- nichenet_ligand_activities(
    ligand_target_matrix = lt_matrix,
    lr_network = networks$lr_network,
    # the rest of the parameters should come
    # from your transcriptomics data:
    expressed_genes_transmitter = expressed_genes_transmitter,
    expressed_genes_receiver = expressed_genes_receiver,
    genes_of_interest = genes_of_interest
)
```

nichenet_ligand_target_links *Compiles a table with weighted ligand-target links*

Description

A wrapper around `nichenetr::get_weighted_ligand_target_links` to compile a data frame with weighted links from the top ligands to their top targets.

Usage

```
nichenet_ligand_target_links(
  ligand_activities,
  ligand_target_matrix,
  genes_of_interest,
  n_top_ligands = 42,
  n_top_targets = 250
)
```

Arguments

`ligand_activities`	Ligand activity table as produced by `nichenetr::predict_ligand_activities`.
`ligand_target_matrix`	Ligand-target matrix as produced by `nichenetr::construct_ligand_target_matrix` or the wrapper around it in the current package: `nichenet_ligand_target_matrix`.
`genes_of_interest`	Character vector with the gene symbols of the genes of interest. These are the genes in the receiver cell population that are potentially affected by ligands expressed by interacting cells (e.g. genes differentially expressed upon cell-cell interaction).
`n_top_ligands`	How many of the top ligands to include in the ligand-target table.
`n_top_targets`	For each ligand, how many of the top targets to include in the ligand-target table.

Value

A tibble with columns ligand, target and weight (i.e. regulatory potential score).

Examples

```
networks <- nichenet_networks()
expression <- nichenet_expression_data()
optimization_results <- nichenet_optimization(networks, expression)
nichenet_model <- nichenet_build_model(optimization_results, networks)
lt_matrix <- nichenet_ligand_target_matrix(
    nichenet_model$weighted_networks,
    networks$lr_network,
```

```
    nichenet_model$optimized_parameters
)
ligand_activities <- nichenet_ligand_activities(
    ligand_target_matrix = lt_matrix,
    lr_network = networks$lr_network,
    # the rest of the parameters should come
    # from your transcriptomics data:
    expressed_genes_transmitter = expressed_genes_transmitter,
    expressed_genes_receiver = expressed_genes_receiver,
    genes_of_interest = genes_of_interest
)
lt_links <- nichenet_ligand_target_links(
    ligand_activities = ligand_activities,
    ligand_target_matrix = lt_matrix,
    genes_of_interest = genes_of_interest,
    n_top_ligands = 20,
    n_top_targets = 100
)
```

`nichenet_ligand_target_matrix` *Creates a NicheNet ligand-target matrix*

Description

Creates a NicheNet ligand-target matrix

Usage

```
nichenet_ligand_target_matrix(
  weighted_networks,
  lr_network,
  optimized_parameters,
  weighted = TRUE,
  construct_ligand_target_matrix_param = list()
)
```

Arguments

`weighted_networks`	Weighted networks as pro
`lr_network`	A data frame with ligand-receptor interactions, as produced by `nichenet_lr_network`.vided by `nichenet_build_model`
`optimized_parameters`	The outcome of NicheNet parameter optimization as produced by `nichenet_build_model`.

weighted	Logical: wether the network sources are weighted. In this function it only affects the output file name.
construct_ligand_target_matrix_param	Override parameters for nichenetr::construct_ligand_target_matrix.

Value

A matrix containing ligand-target probability scores.

Examples

```
networks <- nichenet_networks()
expression <- nichenet_expression_data()
optimization_results <- nichenet_optimization(networks, expression)
nichenet_model <- nichenet_build_model(optimization_results, networks)
lt_matrix <- nichenet_ligand_target_matrix(
    nichenet_model$weighted_networks,
    networks$lr_network,
    nichenet_model$optimized_parameters
)
```

---

nichenet_lr_network *Builds a NicheNet ligand-receptor network*

Description

Builds ligand-receptor network prior knowledge for NicheNet using multiple resources.

Usage

```
nichenet_lr_network(
  omnipath = list(),
  guide2pharma = list(),
  ramilowski = list(),
  only_omnipath = FALSE
)
```

Arguments

omnipath	List with paramaters to be passed to nichenet_lr_network_omnipath.
guide2pharma	List with paramaters to be passed to nichenet_lr_network_guide2pharma.
ramilowski	List with paramaters to be passed to nichenet_lr_network_ramilowski.
only_omnipath	Logical: a shortcut to use only OmniPath as network resource.

Value

A network data frame (tibble) with ligand-receptor interactions suitable for use with NicheNet.

See Also

- nichenet_lr_network_omnipath
- nichenet_lr_network_guide2pharma
- nichenet_lr_network_ramilowski

Examples

```
# load everything with the default parameters:
lr_network <- nichenet_lr_network()

# don't use Ramilowski:
lr_network <- nichenet_lr_network(ramilowski = NULL)

# use only OmniPath:
lr_network_omnipath <- nichenet_lr_network(only_omnipath = TRUE)
```

---

nichenet_lr_network_guide2pharma

Ligand-receptor network from Guide to Pharmacology

Description

Downloads ligand-receptor interactions from the Guide to Pharmacology database and converts it to a format suitable for NicheNet.

Usage

```
nichenet_lr_network_guide2pharma()
```

Value

Data frame with ligand-receptor interactions in NicheNet format.

See Also

nichenet_lr_network,guide2pharma_download

Examples

```
g2p_lr_network <- nichenet_lr_network_guide2pharma()
```

`nichenet_lr_network_omnipath` *Builds ligand-receptor network for NicheNet using OmniPath*

Description

Retrieves network prior knowledge from OmniPath and provides it in a format suitable for NicheNet. This method never downloads the 'ligrecextra' dataset because the ligand-receptor interactions are supposed to come from `nichenet_lr_network_omnipath`.

Usage

```
nichenet_lr_network_omnipath(min_curation_effort = 0, ...)
```

Arguments

`min_curation_effort`	Lower threshold for curation effort
`...`	Passed to `import_intercell_network`

Value

A network data frame (tibble) with ligand-receptor interactions suitable for use with NicheNet.

See Also

- `nichenet_lr_network`
- `import_intercell_network`

Examples

```
# use only ligand-receptor interactions (not for example ECM-adhesion):
op_lr_network <- nichenet_lr_network_omnipath(ligand_receptor = TRUE)

# use only CellPhoneDB and Guide to Pharmacology:
op_lr_network <- nichenet_lr_network_omnipath(
    resources = c('CellPhoneDB', 'Guide2Pharma')
)

# only interactions where the receiver is a transporter:
op_lr_network <- nichenet_lr_network_omnipath(
    receiver_param = list(parent = 'transporter')
)
```

nichenet_lr_network_ramilowski *Ligand-receptor network from Ramilowski 2015*

Description

Downloads ligand-receptor interactions from Supplementary Table 2 of the paper 'A draft network of ligand–receptor-mediated multicellular signalling in human' (Ramilowski et al. 2015, https://www.nature.com/articles/ncomms8866). It converts the downloaded table to a format suitable for NicheNet.

Usage

```
nichenet_lr_network_ramilowski(
  evidences = c("literature supported", "putative")
)
```

Arguments

evidences	Character: evidence types, "literature supported", "putative" or both.

Value

Data frame with ligand-receptor interactions in NicheNet format.

See Also

- nichenet_lr_network
- ramilowski_download

Examples

```
# use only the literature supported data:
rami_lr_network <- nichenet_lr_network_ramilowski(
    evidences = 'literature supported'
)
```

nichenet_main *Executes the full NicheNet pipeline*

Description

Builds all prior knowledge data required by NicheNet. For this it calls a multitude of methods to download and combine data from various databases according to the settings. The content of the prior knowledge data is highly customizable, see the documentation of the related functions. After the prior knowledge is ready, it performs parameter optimization to build a NicheNet model. This results a weighted ligand- target matrix. Then, considering the expressed genes from user provided data, a gene set of interest and background genes, it executes the NicheNet ligand activity analysis.

Usage

```
nichenet_main(
  only_omnipath = FALSE,
  expressed_genes_transmitter = NULL,
  expressed_genes_receiver = NULL,
  genes_of_interest = NULL,
  background_genes = NULL,
  n_top_ligands = 42,
  n_top_targets = 250,
  signaling_network = list(),
  lr_network = list(),
  gr_network = list(),
  make_multi_objective_function_param = list(),
  objective_function_param = list(),
  mlrmbo_optimization_param = list(),
  construct_ligand_target_matrix_param = list(),
  results_dir = NULL
)
```

Arguments

only_omnipath	Logical: use only OmniPath for network knowledge. This is a simple switch for convenience, further options are available by the other arguments. By default we use all available resources. The networks can be customized on a resource by resource basis, as well as providing custom parameters for individual resources, using the parameters 'signaling_network', 'lr_network' and 'gr_network'.
expressed_genes_transmitter	Character vector with the gene symbols of the genes expressed in the cells transmitting the signal.
expressed_genes_receiver	Character vector with the gene symbols of the genes expressed in the cells receiving the signal.

genes_of_interest
: Character vector with the gene symbols of the genes of interest. These are the genes in the receiver cell population that are potentially affected by ligands expressed by interacting cells (e.g. genes differentially expressed upon cell-cell interaction).

background_genes
: Character vector with the gene symbols of the genes to be used as background.

n_top_ligands
: How many of the top ligands to include in the ligand-target table.

n_top_targets
: How many of the top targets (for each of the top ligands) to consider in the ligand-target table.

signaling_network
: A list of parameters for building the signaling network, passed to nichenet_signaling_network

lr_network
: A list of parameters for building the ligand-receptor network, passed to nichenet_lr_network

gr_network
: A list of parameters for building the gene regulatory network, passed to nichenet_gr_network

make_multi_objective_function_param
: Override parameters for smoof::makeMultiObjectiveFunction.

objective_function_param
: Override additional arguments passed to the objective function.

mlrmbo_optimization_param
: Override arguments for nichenetr::mlrmbo_optimization.

construct_ligand_target_matrix_param
: Override parameters for nichenetr::construct_ligand_target_matrix.

results_dir
: Character: path to the directory to save intermediate and final outputs from NicheNet methods.

Value

A named list with the intermediate and final outputs of the pipeline: 'networks', 'expression', 'optimized_parameters', 'weighted_networks' and 'ligand_target_matrix'.

See Also

- nichenet_networks
- nichenet_signaling_network
- nichenet_lr_network
- nichenet_gr_network

Examples

```
nichenet_results <- nichenet_main(
    # altering some network resource parameters, the rest
    # of the resources will be loaded according to the defaults
    signaling_network = list(
        cpdb = NULL, # this resource will be excluded
        evex = list(min_confidence = 1.0) # override some parameters
    ),
```

```
    gr_network = list(only_omnipath = TRUE),
    n_top_ligands = 20,
    # override the default number of CPU cores to use
    mlrmbo_optimization_param = list(ncores = 4)
)
```

`nichenet_networks` *Builds NicheNet network prior knowledge*

Description

Builds network knowledge required by NicheNet. For this it calls a multitude of methods to download and combine data from various databases according to the settings. The content of the prior knowledge data is highly customizable, see the documentation of the related functions.

Usage

```
nichenet_networks(
  signaling_network = list(),
  lr_network = list(),
  gr_network = list(),
  only_omnipath = FALSE
)
```

Arguments

`signaling_network`	A list of parameters for building the signaling network, passed to `nichenet_signaling_network`
`lr_network`	A list of parameters for building the ligand-receptor network, passed to `nichenet_lr_network`
`gr_network`	A list of parameters for building the gene regulatory network, passed to `nichenet_gr_network`
`only_omnipath`	Logical: a shortcut to use only OmniPath as network resource.

Value

A named list with three network data frames (tibbles): the signaling, the ligand-receptor (lr) and the gene regulatory (gr) networks.

See Also

- `nichenet_signaling_network`
- `nichenet_lr_network`
- `nichenet_gr_network`

Examples

```
networks <- nichenet_networks()
dplyr::sample_n(networks$gr_network, 10)
# # A tibble: 10 x 4
#    from    to       source               database
#    <chr>   <chr>    <chr>                <chr>
#  1 MAX     ALG3     harmonizome_ENCODE   harmonizome
#  2 MAX     IMPDH1   harmonizome_ENCODE   harmonizome
#  3 SMAD5   LCP1     Remap_5              Remap
#  4 HNF4A   TNFRSF19 harmonizome_CHEA     harmonizome
#  5 SMC3    FAP      harmonizome_ENCODE   harmonizome
#  6 E2F6    HIST1H1B harmonizome_ENCODE   harmonizome
#  7 TFAP2C  MAT2B    harmonizome_ENCODE   harmonizome
#  8 USF1    TBX4     harmonizome_TRANSFAC harmonizome
#  9 MIR133B FETUB    harmonizome_TRANSFAC harmonizome
# 10 SP4     HNRNPH2  harmonizome_ENCODE   harmonizome


# use only OmniPath:
omnipath_networks <- nichenet_networks(only_omnipath = TRUE)
```

---

nichenet_optimization *Optimizes NicheNet model parameters*

Description

Optimize NicheNet method parameters, i.e. PageRank parameters and source weights, basedon a collection of experiments where the effect of a ligand on gene expression was measured.

Usage

```
nichenet_optimization(
  networks,
  expression,
  make_multi_objective_function_param = list(),
  objective_function_param = list(),
  mlrmbo_optimization_param = list()
)
```

Arguments

networks	A list with NicheNet format signaling, ligand-receptor and gene regulatory networks as produced by nichenet_networks.
expression	A list with expression data from ligand perturbation experiments, as produced by nichenet_expression_data.
make_multi_objective_function_param	Override parameters for smoof::makeMultiObjectiveFunction.

objective_function_param
: Override additional arguments passed to the objective function.

mlrmbo_optimization_param
: Override arguments for nichenetr::mlrmbo_optimization.

Value

A result object from the function 'mlrMBO::mbo'. Among other things, this contains the optimal parameter settings, the output corresponding to every input etc.

Examples

```
networks <- nichenet_networks()
expression <- nichenet_expression_data()
optimization_results <- nichenet_optimization(networks, expression)
```

---

nichenet_remove_orphan_ligands

Removes experiments with orphan ligands

Description

Removes from the expression data the perturbation experiments involving ligands without connections.

Usage

```
nichenet_remove_orphan_ligands(expression, lr_network)
```

Arguments

expression	Expression data as returned by nichenet_expression_data.
lr_network	A NicheNet format ligand-recptor network data frame as produced by nichenet_lr_network.

Value

The same list as 'expression' with certain elements removed.

Examples

```
lr_network <- nichenet_lr_network()
expression <- nichenet_expression_data()
expression <- nichenet_remove_orphan_ligands(expression, lr_network)
```

nichenet_results_dir *Path to the current NicheNet results directory*

Description

Path to the directory to save intermediate and final outputs from NicheNet methods.

Usage

```
nichenet_results_dir()
```

Value

Character: path to the NicheNet results directory.

Examples

```
nichenet_results_dir()
# [1] "nichenet_results"
```

nichenet_signaling_network *Builds a NicheNet signaling network*

Description

Builds signaling network prior knowledge for NicheNet using multiple resources.

Usage

```
nichenet_signaling_network(
  omnipath = list(),
  pathwaycommons = list(),
  harmonizome = list(),
  vinayagam = list(),
  cpdb = list(),
  evex = list(),
  inbiomap = list(),
  only_omnipath = FALSE
)
```

Arguments

omnipath	List with paramaters to be passed to nichenet_signaling_network_omnipath.
pathwaycommons	List with paramaters to be passed to nichenet_signaling_network_pathwaycommons.
harmonizome	List with paramaters to be passed to nichenet_signaling_network_harmonizome.
vinayagam	List with paramaters to be passed to nichenet_signaling_network_vinayagam.
cpdb	List with paramaters to be passed to nichenet_signaling_network_cpdb.
evex	List with paramaters to be passed to nichenet_signaling_network_evex.
inbiomap	List with paramaters to be passed to nichenet_signaling_network_inbiomap.
only_omnipath	Logical: a shortcut to use only OmniPath as network resource.

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

See Also

- nichenet_signaling_network_omnipath
- nichenet_signaling_network_pathwaycommons
- nichenet_signaling_network_harmonizome
- nichenet_signaling_network_vinayagam
- nichenet_signaling_network_cpdb
- nichenet_signaling_network_evex
- nichenet_signaling_network_inbiomap

Examples

```
# load everything with the default parameters:
sig_network <- nichenet_signaling_network()

# override parameters for some resources:
sig_network <- nichenet_signaling_network(
    omnipath = list(resources = c('SIGNOR', 'SignaLink3', 'SPIKE')),
    pathwaycommons = NULL,
    harmonizome = list(datasets = c('phosphositeplus', 'depod')),
    cpdb = list(complex_max_size = 1, min_score = .98),
    evex = list(min_confidence = 1.5)
)

# use only OmniPath:
sig_network_omnipath <- nichenet_signaling_network(only_omnipath = TRUE)
```

nichenet_signaling_network_cpdb

Builds signaling network for NicheNet using ConsensusPathDB

Description

Builds signaling network prior knowledge using ConsensusPathDB (CPDB) data. Note, the interactions from CPDB are not directed and many of them comes from complex expansion. Find out more at http://cpdb.molgen.mpg.de/.

Usage

```
nichenet_signaling_network_cpdb(...)
```

Arguments

...	Passed to consensuspathdb_download.

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

See Also

- nichenet_signaling_network
- consensuspathdb_download

Examples

```
# use some parameters stricter than default:
cpdb_signaling_network <- nichenet_signaling_network_cpdb(
    complex_max_size = 2,
    min_score = .99
)
```

nichenet_signaling_network_evex

NicheNet signaling network from EVEX

Description

Builds signaling network prior knowledge for NicheNet from the EVEX database.

Usage

```
nichenet_signaling_network_evex(top_confidence = 0.75, indirect = FALSE, ...)
```

Arguments

top_confidence	Double, between 0 and 1. Threshold based on the quantile of the confidence score.
indirect	Logical: whether to include indirect interactions.
...	Ignored.

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

See Also

- evex_download
- nichenet_signaling_network

Examples

```
ev_signaling_network <- nichenet_signaling_network_evex(
    top_confidence = .9
)
```

---

nichenet_signaling_network_harmonizome

NicheNet signaling network from Harmonizome

Description

Builds signaling network prior knowledge for NicheNet using Harmonizome

Usage

```
nichenet_signaling_network_harmonizome(
  datasets = c("phosphositeplus", "kea", "depod"),
  ...
)
```

Arguments

datasets	The datasets to use. For possible values please refer to default value and the Harmonizome webpage.
...	Ignored.

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

Examples

```
# use only KEA and PhosphoSite:
hz_signaling_network <- nichenet_signaling_network_harmonizome(
    datasets = c('kea', 'phosphositeplus')
)
```

`nichenet_signaling_network_inbiomap` *NicheNet signaling network from InWeb InBioMap*

Description

Builds signaling network prior knowledge for NicheNet from the InWeb InBioMap database.

Usage

```
nichenet_signaling_network_inbiomap(...)
```

Arguments

`...`	Ignored.

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

See Also

`nichenet_signaling_network`, `inbiomap_download`

Examples

```
ib_signaling_network <- nichenet_signaling_network_inbiomap()
```

nichenet_signaling_network_omnipath

Builds signaling network for NicheNet using OmniPath

Description

Retrieves network prior knowledge from OmniPath and provides it in a format suitable for NicheNet. This method never downloads the 'ligrecextra' dataset because the ligand-receptor interactions are supposed to come from `nichenet_lr_network_omnipath`.

Usage

```
nichenet_signaling_network_omnipath(min_curation_effort = 0, ...)
```

Arguments

`min_curation_effort`	Lower threshold for curation effort
`...`	Passed to `import_post_translational_interactions`

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

See Also

- `nichenet_signaling_network`

Examples

```
# use interactions with at least 2 evidences (reference or database)
op_signaling_network <- nichenet_signaling_network_omnipath(
    min_curation_effort = 2
)
```

nichenet_signaling_network_pathwaycommons

NicheNet signaling network from PathwayCommons

Description

Builds signaling network prior knowledge for NicheNet using PathwayCommons.

Usage

```
nichenet_signaling_network_pathwaycommons(
  interaction_types = c("catalysis-precedes", "controls-phosphorylation-of",
    "controls-state-change-of", "controls-transport-of", "in-complex-with",
    "interacts-with"),
  ...
)
```

Arguments

interaction_types	Character vector with PathwayCommons interaction types. Please refer to the default value and the PathwayCommons webpage.
...	Ignored.

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

Examples

```
# use only the "controls-transport-of" interactions:
pc_signaling_network <- nichenet_signaling_network_pathwaycommons(
    interaction_types = 'controls-transport-of'
)
```

---

nichenet_signaling_network_vinayagam

NicheNet signaling network from Vinayagam

Description

Builds signaling network prior knowledge for NicheNet using Vinayagam 2011 Supplementary Table S6. Find out more at https://doi.org/10.1126/scisignal.2001699.

Usage

```
nichenet_signaling_network_vinayagam(...)
```

Arguments

...	Ignored.

Value

A network data frame (tibble) with signaling interactions suitable for use with NicheNet.

Examples

```
vi_signaling_network <- nichenet_signaling_network_vinayagam()
```

OmnipathR *The OmnipathR package*

Description

OmnipathR is an R package built to provide easy access to the data stored in the OmniPath web service:

https://omnipathdb.org/

And a number of other resources, such as BioPlex, ConsensusPathDB, EVEX, Guide to Pharmacology (IUPHAR/BPS), Harmonizome, HTRIdb, InWeb InBioMap, Pathway Commons, Ramilowski et al. 2015, RegNetwork, ReMap, TF census, TRRUST and Vinayagam et al. 2011.

The OmniPath web service implements a very simple REST style API. This package make requests by the HTTP protocol to retreive the data. Hence, fast Internet access is required for a propser use of OmnipathR.

The package also provides some utility functions to filter, analyse and visualize the data. Furthermore, OmnipathR features a close integration with the NicheNet method for ligand activity prediction from transcriptomics data, and its R implementation nichenetr (available in CRAN).

Author(s)

Alberto Valdeolivas <<alvaldeolivas@gmail>> and Denes Turei <<turei.denes@gmail.com>> and Attila Gabor <<gaborattila87@gmail.com>>

Examples

```
# Download post-translational modifications:
enzsub <- import_omnipath_enzsub(resources = c("PhosphoSite", "SIGNOR"))

# Download protein-protein interactions
interactions <- import_omnipath_interactions(resources = c("SignaLink3"))

# Convert to igraph objects:
enzsub_g <- enzsub_graph(enzsub = enzsub)
OPI_g <- interaction_graph(interactions = interactions )

# Print some interactions:
print_interactions(head(ptms))

# interactions with references:
print_interactions(tail(ptms),writeRefs=TRUE)

# find interactions between kinase and substrate:
print_interactions(dplyr::filter(ptms,enzyme_genesymbol=="MAP2K1",
```

```
   substrate_genesymbol=="MAPK3"))

# find shortest paths on the directed network between proteins
print_path_es(shortest_paths(OPI_g, from = "TYRO3", to = "STAT3",
   output = 'epath')$epath[[1]], OPI_g)

# find all shortest paths between proteins
print_path_vs(
    all_shortest_paths(
        enzsub_g,
        from = "SRC",
        to = "STAT1"
    )$res,
    enzsub_g
)
```

`omnipath_cache_autoclean` *Keeps only the latest versions of complete downloads*

Description

Removes the old versions, the failed downloads and the files in the cache directory which are missing from the database. For more flexible operations use `omnipath_cache_remove` and `omnipath_cache_clean`.

Usage

```
omnipath_cache_autoclean()
```

Value

Invisibl returns the cache database (list of cache records).

Examples

```
omnipath_cache_autoclean()
```

omnipath_cache_clean *Removes the items from the cache directory which are unknown by the cache database*

Description

Removes the items from the cache directory which are unknown by the cache database

Usage

```
omnipath_cache_clean()
```

Value

Returns 'NULL'.

Examples

```
omnipath_cache_clean()
```

omnipath_cache_clean_db *Removes the cache database entries without existing files*

Description

Removes the cache database entries without existing files

Usage

```
omnipath_cache_clean_db(...)
```

Arguments

...	Ignored.

Value

Returns 'NULL'.

Examples

```
omnipath_cache_clean_db()
```

omnipath_cache_download_ready *Sets the download status to ready for a cache item*

Description

Sets the download status to ready for a cache item

Usage

```
omnipath_cache_download_ready(version, key = NULL)
```

Arguments

version	Version of the cache item. If does not exist a new version item will be created
key	Key of the cache item

Value

Character: invisibly returns the version number of the cache version item.

Examples

```
bioc_url <- 'https://bioconductor.org/'
# request a new version item (or retrieve the latest)
new_version <- omnipath_cache_latest_or_new(url = bioc_url)
# check if the version item is not a finished download
new_version$status
# [1] "unknown"
# download the file
download.file(url = bioc_url, destfile = new_version$path)
# report to the cache database that the download is ready
omnipath_cache_download_ready(new_version)
# now the status is ready:
version <- omnipath_cache_latest_or_new(url = bioc_url)
version$status
# "ready"
version$dl_finished
# [1] "2021-03-09 16:48:38 CET"
omnipath_cache_remove(url = bioc_url) # cleaning up
```

`omnipath_cache_filter_versions` *Filters the versions from one cache record*

Description

Filters the versions based on multiple conditions: their age and status

Usage

```
omnipath_cache_filter_versions(
  record,
  latest = FALSE,
  max_age = NULL,
  min_age = NULL,
  status = CACHE_STATUS$READY
)
```

Arguments

`record`	A cache record
`latest`	Return the most recent version
`max_age`	The maximum age in days (e.g. 5: 5 days old or more recent)
`min_age`	The minimum age in days (e.g. 5: 5 days old or older)
`status`	Character vector with status codes. By default only the versions with 'ready' (completed download) status are selected

Value

Character vector with version IDs, NA if no version satisfies the conditions

Examples

```
# creating an example cache record
bioc_url <- 'https://bioconductor.org/'
version <- omnipath_cache_latest_or_new(url = bioc_url)
download.file(bioc_url, destfile = version$path)
omnipath_cache_download_ready(version)
record <- dplyr::first(omnipath_cache_search('biocond'))

# only the versions with status "ready"
version_numbers <- omnipath_cache_filter_versions(record, status = 'ready')
omnipath_cache_remove(url = bioc_url) # cleaning up
```

omnipath_cache_get *Retrieves one item from the cache directory*

Description

Retrieves one item from the cache directory

Usage

```
omnipath_cache_get(
  key = NULL,
  url = NULL,
  post = NULL,
  payload = NULL,
  create = TRUE,
  ...
)
```

Arguments

key	The key of the cache record
url	URL pointing to the resource
post	HTTP POST parameters as a list
payload	HTTP data payload
create	Create a new entry if doesn't exist yet
...	Passed to omnipath_cache_record (internal function)

Value

Cache record: an existing record if the entry already exists, otherwise a newly created and inserted record

Examples

```
# create an example cache record
bioc_url <- 'https://bioconductor.org/'
version <- omnipath_cache_latest_or_new(url = bioc_url)
omnipath_cache_remove(url = bioc_url) # cleaning up

# retrieve the cache record
record <- omnipath_cache_get(url = bioc_url)
record$key
# [1] "41346a00fb20d2a9df03aa70cf4d50bf88ab154a"
record$url
# [1] "https://bioconductor.org/"
```

omnipath_cache_key *Generates a hash which identifies an element in the cache database*

Description

Generates a hash which identifies an element in the cache database

Usage

```
omnipath_cache_key(url, post = NULL, payload = NULL)
```

Arguments

url	Character vector with URLs
post	List with the HTTP POST parameters or a list of lists if the url vector is longer than 1. NULL for queries without POST parameters.
payload	HTTP data payload. List with multiple items if the url vector is longer than 1. NULL for queries without data.

Value

Character vector of cache record keys.

Examples

```
bioc_url <- 'https://bioconductor.org/'
omnipath_cache_key(bioc_url)
# [1] "41346a00fb20d2a9df03aa70cf4d50bf88ab154a"
```

omnipath_cache_latest_or_new *The latest or a new version of a cache record*

Description

Looks up a record in the cache and returns its latest valid version. If the record doesn't exist or no valid version available, creates a new one.

Usage

```
omnipath_cache_latest_or_new(
  key = NULL,
  url = NULL,
  post = NULL,
  payload = NULL,
  create = TRUE,
  ...
)
```

Arguments

key	The key of the cache record
url	URL pointing to the resource
post	HTTP POST parameters as a list
payload	HTTP data payload
create	Logical: whether to create and return a new version. If FALSE only the latest existing valid version is returned, if available.
...	Passed to omnipath_cache_get

Value

A cache version item.

Examples

```
# retrieve the latest version of the first cache record
# found by the search keyword "bioplex"
latest_bioplex <-
    omnipath_cache_latest_or_new(
        names(omnipath_cache_search('bioplex'))[1]
    )

latest_bioplex$dl_finished
# [1] "2021-03-09 14:28:50 CET"
latest_bioplex$path
# [1] "/home/denes/.cache/OmnipathR/378e0def2ac97985f629-1.rds"


# create an example cache record
bioc_url <- 'https://bioconductor.org/'
version <- omnipath_cache_latest_or_new(url = bioc_url)
omnipath_cache_remove(url = bioc_url) # cleaning up
```

omnipath_cache_latest_version *Finds the most recent version in a cache record*

Description

Finds the most recent version in a cache record

Usage

```
omnipath_cache_latest_version(record)
```

Arguments

record	A cache record

Value

Character: the version ID with the most recent download finished time

omnipath_cache_load *Loads an R object from the cache*

Description

Loads the object from RDS format.

Usage

```
omnipath_cache_load(
  key = NULL,
  version = NULL,
  url = NULL,
  post = NULL,
  payload = NULL
)
```

Arguments

key	Key of the cache item
version	Version of the cache item. If does not exist or NULL, the latest version will be retrieved
url	URL of the downloaded resource
post	HTTP POST parameters as a list
payload	HTTP data payload

Value

Object loaded from the cache RDS file.

See Also

omnipath_cache_save

Examples

```
# works only if you have already this item in the cache
intercell_data <- omnipath_cache_load(url = paste0(
    'https://omnipathdb.org/intercell?resources=Adhesome,Almen2009,',
    'Baccin2019,CSPA,CellChatDB&license=academic'
))
class(intercell_data)
# [1] "data.frame"
nrow(intercell_data)
# [1] 16622
attr(intercell_data, 'origin')
# [1] "cache"


# basic example of saving and loading to and from the cache:
bioc_url <- 'https://bioconductor.org/'
bioc_html <- readChar(url(bioc_url), nchars = 99999)
omnipath_cache_save(bioc_html, url = bioc_url)
bioc_html <- omnipath_cache_load(url = bioc_url)
```

---

omnipath_cache_move_in

Moves an existing file into the cache

Description

Either the key or the URL (with POST and payload) must be provided.

Usage

```
omnipath_cache_move_in(
  path,
  key = NULL,
  version = NULL,
  url = NULL,
  post = NULL,
  payload = NULL,
  keep_original = FALSE
)
```

Arguments

path	Path to the source file
key	Key of the cache item
version	Version of the cache item. If does not exist a new version item will be created
url	URL of the downloaded resource
post	HTTP POST parameters as a list
payload	HTTP data payload
keep_original	Whether to keep or remove the original file

Value

Character: invisibly returns the version number of the cache version item.

See Also

omnipath_cache_save

Examples

```
omnipath_cache_move_in('some/file.zip', url = 'the_download_address')


# basic example of moving a file to the cache:

bioc_url <- 'https://bioconductor.org/'
html_file <- tempfile(fileext = '.html')
download.file(bioc_url, html_file)
omnipath_cache_move_in(path = html_file, url = bioc_url)
omnipath_cache_remove(url = bioc_url) # cleaning up
```

---

omnipath_cache_remove *Removes contents from the cache directory*

Description

According to the parameters, it can remove contents older than a certain age, or contents having a more recent version, one specific item, or wipe the entire cache.

Usage

```
omnipath_cache_remove(key = NULL, url = NULL, post = NULL,
    payload = NULL, max_age = NULL, min_age = NULL, status = NULL,
    only_latest = FALSE, wipe = FALSE, autoclean = TRUE)
```

Arguments

key	The key of the cache record
url	URL pointing to the resource
post	HTTP POST parameters as a list
payload	HTTP data payload
max_age	Age of cache items in days. Remove everything that is older than this age
min_age	Age of cache items in days. Remove everything more recent than this age
status	Remove items having any of the states listed here
only_latest	Keep only the latest version
wipe	Logical: if TRUE, removes all files from the cache and the cache database. Same as calling omnipath_cache_wipe.
autoclean	Remove the entries about failed downloads, the files in the cache directory which are missing from the cache database, and the entries without existing files in the cache directory

Value

Invisibly returns the cache database (list of cache records).

See Also

- omnipath_cache_wipe
- omnipath_cache_clean
- omnipath_cache_autoclean

Examples

```
# remove all cache data from the BioPlex database
cache_records <- omnipath_cache_search(
    'bioplex',
    ignore.case = TRUE
)
omnipath_cache_remove(names(cache_records))

# remove a record by its URL
regnetwork_url <- 'http://www.regnetworkweb.org/download/human.zip'
omnipath_cache_remove(url = regnetwork_url)

# remove all records older than 30 days
omnipath_cache_remove(max_age = 30)

# for each record, remove all versions except the latest
omnipath_cache_remove(only_latest = TRUE)


bioc_url <- 'https://bioconductor.org/'
version <- omnipath_cache_latest_or_new(url = bioc_url)
```

```
download.file(bioc_url, version$path)
omnipath_cache_download_ready(version)
key <- omnipath_cache_key(bioc_url)
omnipath_cache_remove(key = key)
```

`omnipath_cache_save` *Saves an R object to the cache*

Description

Exports the object in RDS format, creates new cache record if necessary.

Usage

```
omnipath_cache_save(
  data,
  key = NULL,
  version = NULL,
  url = NULL,
  post = NULL,
  payload = NULL
)
```

Arguments

`data`	An object
`key`	Key of the cache item
`version`	Version of the cache item. If does not exist a new version item will be created
`url`	URL of the downloaded resource
`post`	HTTP POST parameters as a list
`payload`	HTTP data payload

Value

Returns invisibly the data itself.

Invisibly returns the ‘data‘.

See Also

`omnipath_cache_move_in`

Examples

```
mydata <- data.frame(a = c(1, 2, 3), b = c('a', 'b', 'c'))
omnipath_cache_save(mydata, url = 'some_dummy_address')
from_cache <- omnipath_cache_load(url = 'some_dummy_address')
from_cache
#   a b
# 1 1 a
# 2 2 b
# 3 3 c
attr(from_cache, 'origin')
# [1] "cache"


# basic example of saving and loading to and from the cache:
bioc_url <- 'https://bioconductor.org/'
bioc_html <- readChar(url(bioc_url), nchars = 99999)
omnipath_cache_save(bioc_html, url = bioc_url)
bioc_html <- omnipath_cache_load(url = bioc_url)
```

---

omnipath_cache_search *Searches for cache items*

Description

Searches the cache records by matching the URL against a string or regexp.

Usage

```
omnipath_cache_search(pattern, ...)
```

Arguments

pattern	String or regular expression.
...	Passed to grep

Value

List of cache records matching the pattern.

Examples

```
# find all cache records from the BioPlex database
bioplex_cache_records <- omnipath_cache_search(
    'bioplex',
    ignore.case = TRUE
)
```

omnipath_cache_set_ext *Sets the file extension for a cache record*

Description

Sets the file extension for a cache record

Usage

```
omnipath_cache_set_ext(key, ext)
```

Arguments

key	Character: key for a cache item, alternatively a version entry.
ext	Character: the file extension, e.g. "zip".

Value

Returns ‘NULL‘.

Examples

```
bioc_url <- 'https://bioconductor.org/'
version <- omnipath_cache_latest_or_new(url = bioc_url)
version$path
# [1] "/home/denes/.cache/OmnipathR/41346a00fb20d2a9df03-1"
download.file(bioc_url, destfile = version$path)
key <- omnipath_cache_key(url = bioc_url)
omnipath_cache_set_ext(key = key, ext = 'html')
version <- omnipath_cache_latest_or_new(url = bioc_url)
version$path
# [1] "/home/denes/.cache/OmnipathR/41346a00fb20d2a9df03-1.html"
record <- omnipath_cache_get(url = bioc_url)
record$ext
# [1] "html"
omnipath_cache_remove(url = bioc_url) # cleaning up
```

`omnipath_cache_update_status` *Updates the status of an existing cache record*

Description

Updates the status of an existing cache record

Usage

```
omnipath_cache_update_status(key, version, status,
    dl_finished = NULL)
```

Arguments

`key`	Key of the cache item
`version`	Version of the cache item. If does not exist a new version item will be created
`status`	The updated status value
`dl_finished`	Timestamp for the time when download was finished, if 'NULL' the value remains unchanged

Value

Character: invisibly returns the version number of the cache version item.

Examples

```
bioc_url <- 'https://bioconductor.org/'
latest_version <- omnipath_cache_latest_or_new(url = bioc_url)
key <- omnipath_cache_key(bioc_url)
omnipath_cache_update_status(
    key = key,
    version = latest_version$number,
    status = 'ready',
    dl_finished = Sys.time()
)
omnipath_cache_remove(url = bioc_url) # cleaning up
```

omnipath_cache_wipe *Permanently removes all the cache contents*

Description

After this operation the cache directory will be completely empty, except an empty cache database file.

Usage

```
omnipath_cache_wipe(...)
```

Arguments

...	Ignored.

Value

Returns 'NULL'.

See Also

omnipath_cache_remove

Examples

```
omnipath_cache_wipe()
# the cache is completely empty:
print(omnipath.env$cache)
# list()
list.files(omnipath_get_cachedir())
# [1] "cache.json"
```

omnipath_load_config *Load the package configuration from a config file*

Description

Load the package configuration from a config file

Usage

```
omnipath_load_config(path = NULL, title = "default", user = FALSE, ...)
```

Arguments

path	Path to the config file.
title	Load the config under this title. One config file might contain multple configurations, each identified by a title. If the title is not available the first section of the config file will be used.
user	Force to use the user level config even if a config file exists in the current directory. By default, the local config files have prioroty over the user level config.
...	Passed to `yaml::yaml.load_file`.

Value

Invisibly returns the config as a list.

Examples

```
# load the config from a custom config file:
omnipath_load_config(path = 'my_custom_omnipath_config.yml')
```

---

omnipath_log *Browse the current OmnipathR log file*

Description

Browse the current OmnipathR log file

Usage

```
omnipath_log()
```

Value

Returns 'NULL'.

See Also

omnipath_logfile

Examples

```
omnipath_log()
# then you can browse the log file, and exit with `q`
```

omnipath_logfile *Returns the path to the current OmnipathR log file*

Description

Returns the path to the current OmnipathR log file

Usage

```
omnipath_logfile()
```

Value

Character: path to the current logfile.

See Also

omnipath_log

Examples

```
omnipath_logfile()
# [1] "/home/denes/omnipathr/omnipathr-log/omnipathr-20210309-1642.log"
```

omnipath_msg *Dispatch a message to the OmnipathR logger*

Description

Any package or script can easily send log messages and establish a logging facility with the fantastic 'logger' package. This function serves the only purpose if you want to inject messages into the logger of OmnipathR. Otherwise we recommend to use the 'logger' package directly.

Usage

```
omnipath_msg(level, ...)
```

Arguments

level	Character, numeric or class loglevel. A log level, if character one of the followings: "fatal", "error", "warn", "success", "info", "trace".
...	Arguments for string formatting, passed sprintf or str_glue.

Value

Returns 'NULL'.

Examples

```
omnipath_msg(
    level = 'success',
    'Talking to you in the name of OmnipathR, my favourite number is %d',
    round(runif(1, 1, 10))
)
```

---

omnipath_reset_config *Restores the built-in default values of all config parameters*

Description

Restores the built-in default values of all config parameters

Usage

```
omnipath_reset_config(save = NULL)
```

Arguments

save	If a path, the restored config will be also saved to this file. If TRUE, the config will be saved to the current default config path (see 'omnipath_get_config_path()').

Value

The config as a list.

See Also

omnipath_load_config, omnipath_save_config

Examples

```
# restore the defaults and write them to the default config file:
omnipath_reset_config()
omnipath_save_config()
```

omnipath_save_config *Save the current package configuration*

Description

Save the current package configuration

Usage

```
omnipath_save_config(path = NULL, title = "default", local = FALSE)
```

Arguments

path	Path to the config file. Directories and the file will be created if don't exist.
title	Save the config under this title. One config file might contain multple configurations, each identified by a title.
local	Save into a config file in the current directory instead of a user level config file. When loading, the config in the current directory has prioroty over the user level config.

Value

Returns 'NULL'.

Examples

```
# after this, all downloads will default to commercial licenses
# i.e. the resources that allow only academic use will be excluded:
options(omnipath.license = 'commercial')
omnipath_save_config()
```

omnipath_set_console_loglevel *Sets the log level for the console*

Description

Use this method to change during a session which messages you want to be printed on the console. Before loading the package, you can set it also by the config file, with the omnipath.console_loglevel key.

Usage

```
omnipath_set_console_loglevel(level)
```

Arguments

level	Character or class 'loglevel'. The desired log level.

Value

Returns 'NULL'.

See Also

omnipath_set_logfile_loglevel

Examples

```
omnipath_set_console_loglevel('warn')
# or:
omnipath_set_console_loglevel(logger::WARN)
```

---

omnipath_set_logfile_loglevel

Sets the log level for the logfile

Description

Use this method to change during a session which messages you want to be written into the logfile. Before loading the package, you can set it also by the config file, with the omnipath.loglevel key.

Usage

```
omnipath_set_logfile_loglevel(level)
```

Arguments

level	Character or class 'loglevel'. The desired log level.

Value

Returns 'NULL'.

See Also

omnipath_set_console_loglevel

Examples

```
omnipath_set_logfile_loglevel('info')
# or:
omnipath_set_logfile_loglevel(logger::INFO)
```

omnipath_set_loglevel *Sets the log level for the package logger*

Description

Sets the log level for the package logger

Usage

```
omnipath_set_loglevel(level, target = "logfile")
```

Arguments

level	Character or class 'loglevel'. The desired log level.
target	Character, either 'logfile' or 'console'

Value

Returns 'NULL'.

Examples

```
omnipath_set_loglevel(logger::FATAL, target = 'console')
```

omnipath_unlock_cache_db *Removes the lock file from the cache directory*

Description

A lock file in the cache directory avoids simulatneous write and read. It's supposed to be removed after each read and write operation. This might not happen if the process crashes during such an operation. In this case you can manually call this function.

Usage

```
omnipath_unlock_cache_db()
```

Value

Logical: returns TRUE if the cache was locked and now is unlocked; FALSE if it was not locked.

Examples

```
omnipath_unlock_cache_db()
```

pathwaycommons_download *Interactions from PathwayCommons*

Description

PathwayCommons (http://www.pathwaycommons.org/) provides molecular interactions from a number of databases, in either BioPAX or SIF (simple interaction format). This function retrieves all interactions in SIF format. The data is limited to the interacting pair and the type of the interaction.

Usage

```
pathwaycommons_download()
```

Value

A data frame (tibble) with interactions.

Examples

```
pc_interactions <- pathwaycommons_download()
 pc_interactions
# # A tibble: 1,884,849 x 3
#    from  type                       to
#    <chr> <chr>                      <chr>
#  1 A1BG  controls-expression-of     A2M
#  2 A1BG  interacts-with             ABCC6
#  3 A1BG  interacts-with             ACE2
#  4 A1BG  interacts-with             ADAM10
#  5 A1BG  interacts-with             ADAM17
# # . with 1,884,839 more rows
```

pivot_annotations *Converts annotation tables to a wide format*

Description

Use this method to reconstitute the annotation tables into the format of the original resources. With the 'wide=TRUE' option import_omnipath_annotations applies this function to the downloaded data.

Usage

```
pivot_annotations(annotations)
```

Arguments

annotations	A data frame of annotations downloaded from the OmniPath web service by import_omnipath_annotations.

Value

A wide format data frame (tibble) if the provided data contains annotations from one resource, otherwise a list of wide format tibbles.

See Also

import_omnipath_annotations

Examples

```
# single resource: the result is a data frame
disgenet <- import_omnipath_annotations(resources = 'DisGeNet')
disgenet <- pivot_annotations(disgenet)
disgenet
# # A tibble: 119,551 x 10
#    uniprot genesymbol entity_type disease score dsi   dpi   nof_pmids
#    <chr>   <chr>      <chr>       <chr>   <chr> <chr> <chr> <chr>
#  1 P04217  A1BG       protein     Schizo. . .  0.3   0.857 0.172 1
#  2 P04217  A1BG       protein     Hepato. . .  0.3   0.857 0.172 1
#  3 P01023  A2M        protein     alpha-. . .  0.31  0.564 0.724 0
#  4 P01023  A2M        protein     Fibros. . .  0.3   0.564 0.724 1
#  5 P01023  A2M        protein     Hepato. . .  0.3   0.564 0.724 1
# # . with 119,541 more rows, and 2 more variables: nof_snps <chr>,
# #   source <chr>

# multiple resources: the result is a list
annotations <- import_omnipath_annotations(
    resources = c('DisGeNet', 'SignaLink_function', 'DGIdb', 'kinase.com')
)
annotations <- pivot_annotations(annotations)
names(annotations)
# [1] "DGIdb"              "DisGeNet"           "kinase.com"
# [4] "SignaLink_function"
annotations$kinase.com
# # A tibble: 825 x 6
#    uniprot genesymbol entity_type group family subfamily
#    <chr>   <chr>      <chr>       <chr> <chr>  <chr>
#  1 P31749  AKT1       protein     AGC   Akt    NA
#  2 P31751  AKT2       protein     AGC   Akt    NA
#  3 Q9Y243  AKT3       protein     AGC   Akt    NA
#  4 O14578  CIT        protein     AGC   DMPK   CRIK
#  5 Q09013  DMPK       protein     AGC   DMPK   GEK
# # . with 815 more rows
```

print_bma_motif_es *Prints BMA motifs to the screen from a sequence of edges*

Description

The motifs can be copy-pasted into a BMA canvas.

Usage

```
print_bma_motif_es(edge_seq, G, granularity = 2)
```

Arguments

edge_seq	An igraph edge sequence.
G	An igraph graph object.
granularity	Numeric: granularity value.

Value

Returns 'NULL'.

Examples

```
interactions <- import_omnipath_interactions(resources = 'ARN')
graph <- interaction_graph(interactions)
print_bma_motif_es(igraph::E(graph)[1], graph)
# {"Model": {
#     "Name": "Omnipath motif",
#     "Variables":[{
#         "Name":"ULK1",
#         "Id":1,
#         "RangeFrom":0,
#         "RangeTo":2,
#         "Formula":""
#     },
#     {
#         "Name":"ATG13",
#         ...
#     }],
# ... (truncated)
# }}
```

print_bma_motif_vs *Prints BMA motifs to the screen from a sequence of nodes*

Description

The motifs can be copy-pasted into a BMA canvas.

Usage

```
print_bma_motif_vs(node_seq, G)
```

Arguments

node_seq	An igraph node sequence.
G	An igraph graph object.

Value

Returns 'NULL'.

Examples

```
interactions <- import_omnipath_interactions(resources = 'ARN')
graph <- interaction_graph(interactions)
print_bma_motif_vs(
    igraph::all_shortest_paths(
        graph,
        from = 'ULK1',
        to = 'ATG13'
    )$res,
    graph
)
```

print_interactions *Print OmniPath interactions*

Description

Prints the interactions or enzyme-substrate relationships in a nice format.

Usage

```
print_interactions(interDF, writeRefs = FALSE)
```

Arguments

interDF	data.frame with the interactions generated by any of the following functions: • import_omnipath_enzsub • import_omnipath_interactions • import_pathwayextra_interactions • import_kinaseextra_interactions • import_ligrecextra_interactions • import_post_translational_interactions • import_dorothea_interactions • import_tf_target_interactions • import_transcriptional_interactions • import_mirnatarget_interactions • import_all_interactions
writeRefs	[FALSE] writes also the PubMed IDs if available

Value

Returns 'NULL'.

Examples

```
enzsub <- import_omnipath_enzsub()
print_interactions(head(enzsub))
print_interactions(tail(enzsub), writeRefs = TRUE)
print_interactions(
    dplyr::filter(
        enzsub,
        enzyme_genesymbol == 'MAP2K1',
        substrate_genesymbol == 'MAPK3'
    )
)

signor <- import_omnipath_interactions(resources = 'SIGNOR')
print_interactions(head(signor))
#            source interaction            target n_resources
# 6 MAPK14 (Q16539)  ==( + )==> MAPKAPK2 (P49137)          23
# 4  TRPM7 (Q96QT4)  ==( + )==>    ANXA1 (P04083)          10
# 1  PRKG1 (Q13976)  ==( - )==>    TRPC3 (Q13507)           8
# 2  PTPN1 (P18031)  ==( - )==>    TRPV6 (Q9H1D0)           6
# 5 PRKACA (P17612)  ==( - )==>   MCOLN1 (Q9GZU1)           6
# 3  RACK1 (P63244)  ==( - )==>    TRPM6 (Q9BX84)           2
```

`print_path_es` *Prints network paths in an edge sequence*

Description

Pretty prints the interactions in a path.

Usage

```
print_path_es(edgeSeq, G)
```

Arguments

`edgeSeq`	edge sequence
`G`	igraph object (from ptms or any interaction dataset)

Value

Returns ‘NULL‘.

See Also

- `print_path_vs`

Examples

```
interactions <- import_omnipath_interactions(resources = c('SignaLink3'))
OPI_g <- interaction_graph(interactions = interactions)
print_path_es(
    suppressWarnings(igraph::shortest_paths(
        OPI_g,
        from = 'TYRO3',
        to = 'STAT3',
        output = 'epath'
    ))$epath[[1]],
    OPI_g
)
```

print_path_vs *Print networks paths given by node sequence*

Description

Prints the interactions in the path in a nice format.

Usage

```
print_path_vs(nodeSeq, G)
```

Arguments

nodeSeq	node sequence
G	igraph object (from ptms or interactions)

Value

Returns 'NULL'.

See Also

print_path_es

Examples

```
interactions <- import_omnipath_interactions(resources=c('SignaLink3'))
OPI_g <- interaction_graph(interactions = interactions)
print_path_vs(
    igraph::all_shortest_paths(
        OPI_g,
        from = 'TYRO3',
        to = 'STAT3'
    )$vpath,
    OPI_g
)
enzsub <- import_omnipath_enzsub(resources=c('PhosphoSite', 'SIGNOR'))
enzsub_g <- enzsub_graph(enzsub)
print_path_vs(
    igraph::all_shortest_paths(
        enzsub_g,
        from = 'SRC',
        to = 'STAT1'
    )$res,
    enzsub_g
)
```

ramilowski_download *Downloads ligand-receptor interactions from Ramilowski et al. 2015*

Description

Curated ligand-receptor pairs from Supplementary Table 2 of the article "A draft network of ligand-receptor mediated multicellular signaling in human" (https://www.nature.com/articles/ncomms8866).

Usage

```
ramilowski_download()
```

Value

A data frame (tibble) with interactions.

Examples

```
rami_interactions <- ramilowski_download()
rami_interactions
# # A tibble: 2,557 x 16
#    Pair.Name Ligand.Approved. Ligand.Name Receptor.Approv.
#    <chr>     <chr>            <chr>       <chr>
#  1 A2M_LRP1  A2M              alpha-2-ma. LRP1
#  2 AANAT_MT. AANAT            aralkylami. MTNR1A
#  3 AANAT_MT. AANAT            aralkylami. MTNR1B
#  4 ACE_AGTR2 ACE              angiotensi. AGTR2
#  5 ACE_BDKR. ACE              angiotensi. BDKRB2
# # . with 2,547 more rows, and 12 more variables: Receptor.Name <chr>,
# #   DLRP <chr>, HPMR <chr>, IUPHAR <chr>, HPRD <chr>,
# #   STRING.binding <chr>, STRING.experiment <chr>, HPMR.Ligand <chr>,
# #   HPMR.Receptor <chr>, PMID.Manual <chr>, Pair.Source <chr>,
# #   Pair.Evidence <chr>
```

regnetwork_directions *Transcription factor effects from RegNetwork*

Description

Transcription factor effects from RegNetwork

Usage

```
regnetwork_directions(organism = "human")
```

Arguments

organism	Character: either human or mouse.

Value

A data frame (tibble) of TF-target interactions with effect signs.

Examples

```
regn_dir <- regnetwork_directions()
regn_dir
# # A tibble: 3,954 x 5
#    source_genesymb. source_entrez target_genesymb. target_entrez
#    <chr>            <chr>         <chr>            <chr>
#  1 AHR              196           CDKN1B           1027
#  2 APLNR            187           PIK3C3           5289
#  3 APLNR            187           PIK3R4           30849
#  4 AR               367           KLK3             354
#  5 ARNT             405           ALDOA            226
# # . with 3,944 more rows, and 1 more variable: effect <dbl>
```

---

regnetwork_download *Interactions from RegNetwork*

Description

Downloads transcriptional and post-transcriptional regulatory interactions from the RegNetwork database (http://www.regnetworkweb.org/). The information about effect signs (stimulation or inhibition), provided by regnetwork_directions are included in the result.

Usage

```
regnetwork_download(organism = "human")
```

Arguments

organism	Character: either human or mouse.

Value

Data frame with interactions.

Examples

```
regn_interactions <- regnetwork_download()
regn_interactions
# # A tibble: 372,778 x 7
#    source_genesymb. source_entrez target_genesymb. target_entrez
#    <chr>            <chr>         <chr>            <chr>
#  1 USF1             7391          S100A6           6277
#  2 USF1             7391          DUSP1            1843
#  3 USF1             7391          C4A              720
#  4 USF1             7391          ABCA1            19
#  5 TP53             7157          TP73             7161
# # . with 372,768 more rows, and 3 more variables: effect <dbl>,
# #   source_type <chr>, target_type <chr>
```

---

remap_dorothea_download

Downloads TF-target interactions from ReMap

Description

ReMap (http://remap.univ-amu.fr/) is a database of ChIP-Seq experiments. It provides raw and merged peaks and CRMs (cis regulatory motifs) with their associations to regulators (TFs). TF-target relationships can be derived as it is written in Garcia-Alonso et al. 2019: "For ChIP-seq, we downloaded the binding peaks from ReMap and scored the interactions between each TF and each gene according to the distance between the TFBSs and the genes' transcription start sites. We evaluated different filtering strategies that consisted of selecting only the top-scoring 100, 200, 500, and 1000 target genes for each TF." (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6673718/#s1title). This function returns the top TF-target relationships as used in DoRothEA: https://github.com/saezlab/dorothea/blob/master/inst/scripts/02_chip_seq.R).

Usage

```
remap_dorothea_download()
```

Value

Data frame with TF-target relationships.

See Also

remap_tf_target_download

Examples

```
remap_interactions <- remap_dorothea_download()
remap_interactions
# # A tibble: 136,988 x 2
#    tf    target
#    <chr> <chr>
#  1 ADNP  ABCC1
#  2 ADNP  ABCC6
#  3 ADNP  ABHD5
#  4 ADNP  ABT1
#  5 ADNP  AC002066.1
# # . with 136,978 more rows
```

`remap_filtered` *Downloads TF-target interactions from ReMap*

Description

Downloads the ReMap TF-target interactions as processed by Garcia-Alonso et al. (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6673718/#s1title) and filters them based on a score threshold, the top targets and whether the TF is included in the TF census (Vaquerizas et al. 2009). The code for filtering is adapted from DoRothEA, written by Christian Holland.

Usage

```
remap_filtered(score = 100, top_targets = 500, only_known_tfs = TRUE)
```

Arguments

`score`	Numeric: a minimum score between 0 and 1000, records with lower scores will be excluded. If NULL no filtering performed.
`top_targets`	Numeric: the number of top scoring targets for each TF. Essentially the maximum number of targets per TF. If NULL the number of targets is not restricted.
`only_known_tfs`	Logical: whether to exclude TFs which are not in TF census.

Value

Data frame with TF-target relationships.

See Also

- `remap_tf_target_download`
- `remap_filtered`
- `tfcensus_download`

Examples

```
remap_interactions <- remap_filtered()
nrow(remap_interactions)
# [1] 145680

remap_interactions <- remap_filtered(top_targets = 100)
remap_interactions
# # A tibble: 30,330 x 2
#    source_genesymbol target_genesymbol
#    <chr>             <chr>
#  1 ADNP              ABCC1
#  2 ADNP              ABT1
#  3 ADNP              AC006076.1
#  4 ADNP              AC007792.1
#  5 ADNP              AC011288.2
# # . with 30,320 more rows
```

`remap_tf_target_download`

Downloads TF-target interactions from ReMap

Description

ReMap (http://remap.univ-amu.fr/) is a database of ChIP-Seq experiments. It provides raw and merged peaks and CRMs (cis regulatory motifs) with their associations to regulators (TFs). TF-target relationships can be derived as it is written in Garcia-Alonso et al. 2019: "For ChIP-seq, we downloaded the binding peaks from ReMap and scored the interactions between each TF and each gene according to the distance between the TFBSs and the genes' transcription start sites. We evaluated different filtering strategies that consisted of selecting only the top-scoring 100, 200, 500, and 1000 target genes for each TF." (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6673718/#s1title). This function retrieves the full processed TF-target list from the data deposited in https://zenodo.org/record/3713238.

Usage

```
remap_tf_target_download()
```

Value

Data frame with TF-target relationships.

See Also

- remap_dorothea_download
- remap_filtered

Examples

```
remap_interactions <- remap_tf_target_download()
remap_interactions
# # A tibble: 9,546,470 x 4
#    source_genesymbol target_genesymbol target_ensembl     score
#    <chr>             <chr>             <chr>              <dbl>
#  1 ADNP              PTPRS             ENSG00000105426.16  1000
#  2 AFF4              PRKCH             ENSG00000027075.14  1000
#  3 AHR               CTNND2            ENSG00000169862.18  1000
#  4 AR                PDE4D             ENSG00000113448.18  1000
#  5 ARID1A            PLEC              ENSG00000178209.14  1000
# # . with 9,546,460 more rows
```

---

tfcensus_download *Downloads the list of transcription factors from TF census*

Description

Vaquerizas et al. published in 2009 a list of transcription factors. This function retrieves Supplementary Table 2 from the article (http://www.nature.com/nrg/journal/v10/n4/index.html).

Usage

```
tfcensus_download()
```

Value

A data frame (tibble) listing transcription factors.

Examples

```
tfcensus <- tfcensus_download()
tfcensus
# # A tibble: 1,987 x 7
#    Class `Ensembl ID` `IPI ID` `Interpro DBD` `Interpro DNA-b.
#    <chr> <chr>        <chr>    <chr>          <chr>
#  1 a     ENSG0000000. IPI0021. NA             IPR001289
#  2 a     ENSG0000000. IPI0004. IPR000047;IPR. NA
#  3 a     ENSG0000000. IPI0001. IPR001356;IPR. NA
#  4 a     ENSG0000000. IPI0029. IPR000910;IPR. NA
#  5 a     ENSG0000000. IPI0001. IPR007087;IPR. IPR006794
# # . with 1,977 more rows, and 2 more variables: `HGNC symbol` <chr>,
# # `Tissue-specificity` <chr>
```

translate_ids *Translates a column of identifiers in a data frame by creating another column with the target identifiers*

Description

Translates a column of identifiers in a data frame by creating another column with the target identifiers

Usage

```
translate_ids(
  d,
  from_col,
  to_col,
  from,
  to,
  uploadlists = TRUE,
  keep_untranslated = TRUE,
  ...
)
```

Arguments

d	A data frame
from_col	Name of an existing column in the data frame containing the identifiers to be translated
to_col	Name for a new column which will contain the target identifiers
from	Identifier type for 'from_col'. See Details for possible values.
to	Identifier type for 'to_col'. See Details for possible values.
uploadlists	Whether to use the 'uploadlists' service from UniProt or the plain query interface (implemented in uniprot_full_id_mapping_table in this package).
keep_untranslated	Keep the records where the source identifier could not be translated. At these records the target identifier will be NA.
...	Passed to uniprot_full_id_mapping_table.

Details

This function uses the uploadlists service of UniProt to obtain identifier translation tables. The possible values for 'from' and 'to' are the identifier type abbreviations used in the UniProt API, please refer to the table here: https://www.uniprot.org/help/api_idmapping The mapping between identifiers can be ambiguous. In this case one row in the original data frame yields multiple rows in the returned data frame.

Value

The data frame ‘d‘ a new column added with the translated IDs.

See Also

uniprot_id_mapping_table

Examples

```
d <- data.frame(uniprot_id = c('P00533', 'Q9ULV1', 'P43897', 'Q9Y2P5'))
d <- translate_ids(d, uniprot_id, genesymbol, 'ID', 'GENENAME')
d
#   uniprot_id  genesymbol
# 1     P00533  EGFR_HUMAN
# 2     Q9ULV1  FZD4_HUMAN
# 3     P43897  EFTS_HUMAN
# 4     Q9Y2P5 S27A5_HUMAN
```

---

trrust_download *Downloads TF-target interactions from TRRUST*

Description

TRRUST v2 (https://www.grnpedia.org/trrust/) is a database of literature mined TF-target interactions for human and mouse.

Usage

```
trrust_download(organism = "human")
```

Arguments

organism	Character: either "human" or "mouse".

Value

A data frame of TF-target interactions.

Examples

```
trrust_interactions <- trrust_download()
trrust_interactions
# # A tibble: 11,698 x 4
#    source_genesymbol target_genesymbol effect reference
#    <chr>             <chr>              <dbl> <chr>
#  1 AATF              BAX                   -1 22909821
#  2 AATF              CDKN1A                 0 17157788
#  3 AATF              KLK3                   0 23146908
```

```
#  4 AATF              MYC                    1 20549547
#  5 AATF              TP53                   0 17157788
#  6 ABL1              BAX                    1 11753601
#  7 ABL1              BCL2                  -1 11753601
# # . with 11,688 more rows
```

`uniprot_full_id_mapping_table` *Creates an ID translation table from UniProt data*

Description

Creates an ID translation table from UniProt data

Usage

```
uniprot_full_id_mapping_table(
  to,
  from = "id",
  reviewed = TRUE,
  organism = 9606
)
```

Arguments

`to`	Target ID type. See Details for possible values.
`from`	Source ID type. See Details for possible values.
`reviewed`	Retrieve only reviewed (‘TRUE‘), only unreviewed (‘FALSE‘) or both (‘NULL‘).
`organism`	Integer, NCBI Taxonomy ID of the organism (by default 9606 for human).

Details

For both source and target ID type, this function accepts column codes used by UniProt and some simple shortcuts defined here. For the UniProt codes please refer to https://www.uniprot.org/help/uniprotkb The shortcuts are entrez, genesymbol, genesymbol_syn (synonym gene symbols), hgnc, embl, refseqp (RefSeq protein), enst (Ensembl transcript), uniprot_entry (UniProtKB AC, e.g. EGFR_HUMAN), protein_name (full name of the protein), uniprot (UniProtKB ID, e.g. P00533).

Value

A data frame (tibble) with columns ‘From‘ and ‘To‘, UniProt IDs and the corresponding foreign IDs, respectively.

Examples

```
uniprot_entrez <- uniprot_full_id_mapping_table(to = 'entrez')
uniprot_entrez
# # A tibble: 20,723 x 2
#    From   To
#    <chr>  <chr>
#  1 Q96R72 NA
#  2 Q9UKL2 23538
#  3 Q9H205 144125
#  4 Q8NGN2 219873
#  5 Q8NGC1 390439
# # . with 20,713 more rows
```

`uniprot_id_mapping_table` *Retrieves an identifier translation table from the UniProt uploadlists service*

Description

Retrieves an identifier translation table from the UniProt uploadlists service

Usage

```
uniprot_id_mapping_table(identifiers, from, to)
```

Arguments

`identifiers`	Character vector of identifiers
`from`	Type of the identifiers provided. See Details for possible values.
`to`	Identifier type to be retrieved from UniProt. See Details for possible values.

Details

This function uses the uploadlists service of UniProt to obtain identifier translation tables. The possible values for 'from' and 'to' are the identifier type abbreviations used in the UniProt API, please refer to the table here: https://www.uniprot.org/help/api_idmapping

Value

A data frame (tibble) with columns 'From' and 'To', the identifiers provided and the corresponding target IDs, respectively.

Examples

```
uniprot_genesymbol <- uniprot_id_mapping_table(
    c('P00533', 'P23771'), 'ID', 'GENENAME'
)
uniprot_genesymbol
# # A tibble: 2 x 2
#   From   To
#   <chr>  <chr>
# 1 P00533 EGFR
# 2 P23771 GATA3
```

`vinayagam_download` *Protein-protein interactions from Vinayagam 2011*

Description

Retrieves the Supplementary Table S6 from Vinayagam et al. 2011. Find out more at https://doi.org/10.1126/scisignal.2001699.

Usage

```
vinayagam_download()
```

Value

A data frame (tibble) with interactions.

Examples

```
vinayagam_interactions <- vinayagam_download()
vinayagam_interactions
# # A tibble: 34,814 x 5
#    `Input-node Gen. `Input-node Gen. `Output-node Ge. `Output-node Ge.
#    <chr>                       <dbl> <chr>                       <dbl>
#  1 C1orf103                    55791 MNAT1                        4331
#  2 MAST2                       23139 DYNLL1                       8655
#  3 RAB22A                      57403 APPL2                       55198
#  4 TRAP1                       10131 EXT2                         2132
#  5 STAT2                        6773 COPS4                       51138
# # . with 34,804 more rows, and 1 more variable:
# # `Edge direction score` <dbl>
```

zenodo_download *Retrieves data from Zenodo*

Description

Zenodo is a repository of large scientific datasets. Many projects and publications make their datasets available at Zenodo. This function downloads an archive from Zenodo and extracts the requested file.

Usage

```
zenodo_download(
  path,
  reader = NULL,
  reader_param = list(),
  url_key = NULL,
  zenodo_record = NULL,
  zenodo_fname = NULL,
  url_param = list(),
  url_key_param = list(),
  ...
)
```

Arguments

path	Character: path to the file within the archive.
reader	Optional, a function to read the connection.
reader_param	List: arguments for the reader function.
url_key	Character: name of the option containing the URL
zenodo_record	The Zenodo record ID, either integer or character.
zenodo_fname	The file name within the record.
url_param	List: variables to insert into the URL string (which is returned from the options).
url_key_param	List: variables to insert into the ‘url_key‘.
...	Passed to archive_extractor

Value

A connection

Examples

```
# an example from the OmnipathR::remap_tf_target_download function:
remap_dorothea <- zenodo_download(
    zenodo_record = 3713238,
    zenodo_fname = 'tf_target_sources.zip',
```

```
        path = (
            'tf_target_sources/chip_seq/remap/gene_tf_pairs_genesymbol.txt'
        ),
        reader = read_tsv,
        reader_param = list(
            col_names = c(
                'source_genesymbol',
                'target_genesymbol',
                'target_ensembl',
                'score'
            ),
            col_types = cols(),
            progress = FALSE
        ),
      resource = 'ReMap'
    )
```

Index