Package 'puremoe'

October 20, 2026

Type Package

Title Integrated Retrieval and Analysis of 'PubMed', 'NIH', and 'NLM' Literature Data

Version 1.1.1

Author Jason Timm [aut, cre]

Maintainer Jason Timm <JaTimm@salud.unm.edu>

Description Retrieve and analyze biomedical literature from 'PubMed' and the wider 'NIH'/'NLM' data stack through a single, PMID-centered interface. A PubMed search resolves to a set of PMIDs, which can be used to retrieve article metadata and abstracts, author affiliations, 'iCite' citation data and links, 'PubTator3' entity and relation annotations, and open-access full text from 'PMC'. A local analysis layer operates on the retrieved tables, supporting corpus expansion through citation links, citation network construction, sentence-level entity co-occurrence, inspection of relation evidence, and 'MeSH' descriptor keyness.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

Depends R (>= 3.5)

Imports rentrez, textshape, xml2, data.table, httr, pbapply, jsonlite, rappdirs, textpress, parallel, tools, utils

Suggests knitr, rmarkdown, DT, dplyr, testthat (>= 3.0.0)

URL https://github.com/jaytimm/puremoe, https://jaytimm.github.io/puremoe/

BugReports https://github.com/jaytimm/puremoe/issues

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-10-20 16:30:02 UTC

Contents

citation_network . . . 2
citation_snowball . . . 3
data_mesh_frequencies . . . 5
data_mesh_thesaurus . . . 5
data_mesh_trees . . . 6
endpoint_info . . . 7
get_records . . . 8
mesh_keyness . . . 9
pmid_to_ftp . . . 10
pmid_to_pmc . . . 11
pubtator_context . . . 12
pubtator_cooccurrence . . . 13
pubtator_network . . . 14
search_pubmed . . . 15

Index **17**

`citation_network` *Build a Citation Network from an iCite Corpus*

Description

Converts an `icites` data.table into a tidy graph representation (nodes + edges) suitable for **igraph** or **tidygraph**. Only edges where *both* endpoints are present in the corpus are retained, so the graph is bounded to the papers you already have metadata for.

Usage

```
citation_network(icites)
```

Arguments

`icites`	A `data.table` returned by `get_records(endpoint = "icites")`. Must contain `pmid` and `citation_net` columns.

Details

RCR and `is_clinical` are carried as node attributes, making the resulting graph immediately weighted by field-normalized impact and enabling bench-to-bedside edge filtering without any additional API calls.

Value

A named list with two `data.tables`:

`nodes` One row per PMID. Contains all iCite metadata columns except `citation_net`. Key columns: `pmid`, `relative_citation_ratio`, `nih_percentile`, `is_clinical`.

`edges` One row per within-corpus directed citation. Columns: `from_pmid` (the citing paper), `to_pmid` (the cited paper).

Examples

```
## Not run:
# network from a seed corpus
pmids |>
  get_records(endpoint = "icites") |>
  citation_network()

# expand first, then fetch iCite metadata for the full network
snowball <- pmids |>
  get_records(endpoint = "icites") |>
  citation_snowball()

snowball$pmid |>
  get_records(endpoint = "icites") |>
  citation_network()

# filter to clinical citation targets
snowball <- pmids |>
  get_records(endpoint = "icites") |>
  citation_snowball()

net <- snowball$pmid |>
  get_records(endpoint = "icites") |>
  citation_network()

clinical_edges <- net$edges |>
  merge(net$nodes[, .(pmid, is_clinical)],
        by.x = "to_pmid", by.y = "pmid") |>
  subset(is_clinical == TRUE)

## End(Not run)
```

`citation_snowball` *Expand a PMID Corpus via One-Hop iCite Citation Snowballing*

Description

Starting from an `icites` data.table returned by `get_records(endpoint = "icites")`, follows the citation links already present in the `citation_net` column and returns a candidate table. The function does not call iCite again; use `get_records(endpoint = "icites")` explicitly on the returned PMIDs if metadata is needed for the expanded corpus.

Usage

```
citation_snowball(
  icites,
  max_nodes = 2000,
  direction = c("both", "citing", "cited"),
  min_links = 2
)
```

Arguments

icites	A data.table returned by get_records(endpoint = "icites"). Must contain pmid and citation_net columns.
max_nodes	Hard ceiling on the total number of PMIDs in the returned corpus (seed + discovered). Candidates are filtered by min_links, ranked by citation-link evidence, and then truncated to the remaining slots after all seed PMIDs are retained. Publication year is not used for this cap because citation_snowball() does not fetch metadata for newly discovered PMIDs. Default 2000.
direction	One of "both" (default), "citing", or "cited". "cited" expands to papers referenced by the seeds; "citing" expands to papers that cite the seeds; "both" combines both directions.
min_links	Minimum number of seed papers a candidate must be linked to in order to be included. Default 2. Higher values yield a smaller, more focused expansion.

Value

A data.table with one row per seed or candidate PMID. Columns are pmid, seed, cited_links, citing_links, and link_count. cited_links counts seed papers that cite the candidate; citing_links counts seed papers cited by the candidate.

Examples

```
## Not run:
pmids <- search_pubmed("metformin AND PCOS [TiAb]")

snowball <- pmids |>
  get_records(endpoint = "icites") |>
  citation_snowball(direction = "cited", min_links = 2)

snowball$pmid |> get_records(endpoint = "pubmed_abstracts")

## End(Not run)
```

data_mesh_frequencies *MeSH Descriptor Frequencies Across PubMed*

Description

Baseline frequencies for MeSH descriptors computed from a local PostgreSQL mirror of PubMed (April 2026). For each descriptor, counts reflect the number of distinct PMIDs indexed with that term; proportions use the full PubMed corpus of 39,703,112 PMIDs as denominator. Descriptor UI and canonical name are joined from the NLM MeSH thesaurus. Intended as a baseline for `mesh_keyness` against arbitrary PubMed subsets.

Usage

```
data_mesh_frequencies
```

Format

A data.table with 30,521 rows and 4 columns:

DescriptorUI MeSH descriptor unique identifier (e.g., `D000001`)

DescriptorName Canonical MeSH descriptor name

n_pmids Number of distinct PubMed records indexed with this descriptor

prop_total Proportion of all 39,703,112 PubMed PMIDs indexed with this descriptor

Source

Computed from `mesh_descriptor` table in a local PubMed PostgreSQL mirror; descriptor metadata from the NLM MeSH Thesaurus (April 2026).

data_mesh_thesaurus *Download and Combine 'MeSH' and Supplemental Thesauruses*

Description

This function downloads and combines the 'MeSH' (Medical Subject Headings) Thesaurus and a supplemental concept thesaurus. The data is sourced from specified URLs and stored locally for subsequent use. By default, the data is stored in a temporary directory. Users can opt into persistent storage by setting ‘use_persistent_storage‘ to TRUE and optionally specifying a path.

Usage

```
data_mesh_thesaurus(
  path = NULL,
  use_persistent_storage = FALSE,
  force_install = FALSE
)
```

Arguments

path	A character string specifying the directory path where data should be stored. If not provided and persistent storage is requested, it defaults to a system-appropriate persistent location managed by 'rappdirs'.
use_persistent_storage	A logical value indicating whether to use persistent storage. If TRUE and no path is provided, data will be stored in a system-appropriate location. Defaults to FALSE, using a temporary directory.
force_install	A logical value indicating whether to force re-downloading of the data even if it already exists locally.

Value

A data.table containing the combined MeSH and supplemental thesaurus data.

Examples

```
if (interactive()) {
  data <- data_mesh_thesaurus()
}
```

---

data_mesh_trees *Download and Load 'MeSH' Trees Data*

Description

This function downloads and loads the 'MeSH' (Medical Subject Headings) Trees data.

Usage

```
data_mesh_trees(
  path = NULL,
  use_persistent_storage = FALSE,
  force_install = FALSE
)
```

Arguments

path	A character string specifying the directory path where data should be stored. If not provided and persistent storage is requested, it defaults to a system-appropriate persistent location managed by 'rappdirs'.
use_persistent_storage	A logical value indicating whether to use persistent storage. If TRUE and no path is provided, data will be stored in a system-appropriate location. Defaults to FALSE, using a temporary directory.
force_install	A logical value indicating whether to force re-downloading of the data even if it already exists locally.

Details

The data is sourced from specified URLs and stored locally for subsequent use. By default, the data is stored in a temporary directory. Users can opt into persistent storage by setting ‘use_persistent_storage‘ to TRUE and optionally specifying a path.

Value

A data frame containing the MeSH Trees data.

Examples

```
if (interactive()) {
  data <- data_mesh_trees()
}
```

---

endpoint_info *Get Information About Available Endpoints*

Description

This function provides detailed information about the available endpoints in the package, including column descriptions, parameters, rate limits, and usage notes.

Usage

```
endpoint_info(endpoint = NULL, format = c("list", "json"))
```

Arguments

endpoint	Character string specifying which endpoint to get information about. If NULL (default), returns a list of all available endpoints.
format	Character string specifying the output format. Either "list" (default) or "json" for JSON-formatted output.

Value

If endpoint is NULL, returns a character vector of available endpoint names. If endpoint is specified, returns a list (or JSON string) with detailed information about that endpoint including description, columns, parameters, rate limits, and notes.

Examples

```
if (interactive()) {
  # List all available endpoints
  endpoint_info()

  # Get information about a specific endpoint
  endpoint_info("pubmed_abstracts")

  # Get information in JSON format
  endpoint_info("icites", format = "json")
}
```

`get_records` *Retrieve Data from 'NLM'/'PubMed' databases Based on PMIDs*

Description

This function retrieves different types of data (like 'PubMed' records, affiliations, 'iCites 'data, etc.) from 'PubMed' based on provided PMIDs. It supports parallel processing for efficiency.

Usage

```
get_records(
  pmids,
  endpoint = c("pubtator", "pubtations", "icites", "pubmed_affiliations",
    "pubmed_abstracts", "pmc_fulltext"),
  cores = 3,
  sleep = 1,
  ncbi_key = NULL,
  icite_timeout = getOption("puremoe.icite_timeout", 15)
)
```

Arguments

`pmids`	A vector of PMIDs for which data is to be retrieved. For 'pmc_fulltext' endpoint, provide full URLs instead (e.g., from `pmid_to_ftp()$url`).
`endpoint`	A character vector specifying the type of data to retrieve ('pubtator', 'pubtations', 'icites', 'pubmed_affiliations', 'pubmed_abstracts', 'pmc_fulltext').
`cores`	Number of cores to use for parallel processing (default is 3).
`sleep`	Duration (in seconds) to pause after each batch
`ncbi_key`	(Optional) NCBI API key for authenticated access.
`icite_timeout`	Maximum elapsed seconds to allow each iCite batch before skipping it and returning PMID-only rows. Defaults to the `puremoe.icite_timeout` option, or 15 seconds if unset.

Details

For the 'pmc_fulltext' endpoint, provide full URLs to PMC Cloud Service XML files. Use pmid_to_ftp to convert PMIDs to PMC IDs and full-text URLs first.

Value

A data.table containing combined results from the specified endpoint, except for the PubTator endpoint, which returns a list with entities and relations data.tables.

Examples

```
if (interactive()) {
  pmids <- c("38136652")
  results <- get_records(pmids, endpoint = "pubmed_abstracts", cores = 1)
}
```

---

mesh_keyness *MeSH Descriptor Keyness for a Retrieved Corpus*

Description

Scores the MeSH descriptors of a retrieved corpus against PubMed-wide descriptor frequencies, identifying the terms that are over- or under-represented relative to PubMed as a whole. This is a local transform of the pubmed_abstracts output – it makes no API calls – and is intended to characterise a corpus and to guide search refinement and expansion.

Usage

```
mesh_keyness(
  records,
  frequencies = NULL,
  measure = c("log_odds", "g2"),
  smoothing = 0.5,
  min_count = 1L
)
```

Arguments

records	A pubmed_abstracts table from get_records(endpoint = "pubmed_abstracts") (with its annotations list-column), or a long data.frame already exposing pmid and DescriptorUI (optionally DescriptorName and a type column, in which case only type == "MeSH" rows are used).
frequencies	Baseline descriptor frequencies. Defaults to the bundled data_mesh_frequencies; must contain DescriptorUI, n_pmids, and prop_total.

measure	Keyness statistic: "log_odds" (default) for a Haldane-corrected log odds ratio with standard error and z-score, or "g2" for the signed Dunning log-likelihood ratio.
smoothing	Positive continuity correction added to each cell of the 2x2 incidence table for measure = "log_odds" (default 0.5, the Haldane-Anscombe correction).
min_count	Drop descriptors indexed in fewer than min_count corpus PMIDs before scoring (default 1).

Details

Keyness is computed on document incidence: for each descriptor, the number of distinct corpus PMIDs indexed with it is compared against the number of distinct PubMed PMIDs indexed with it (data_mesh_frequencies).

Value

A data.table, one row per scored descriptor, ordered by keyness (descending). Common columns: DescriptorUI, DescriptorName, corpus_count, corpus_total, corpus_prop, baseline_count, baseline_total, baseline_prop, and direction ("over"/"under"). With measure = "log_odds": log_odds, std_error, z. With measure = "g2": g2.

Examples

```
## Not run:
pmids   <- search_pubmed('"doxorubicin"[TiAb] AND "cardiotoxicity"[TiAb]')
records <- get_records(pmids, endpoint = "pubmed_abstracts")

mesh_keyness(records)                       # most over-represented descriptors
mesh_keyness(records, measure = "g2")

## End(Not run)
```

---

pmid_to_ftp *Convert PubMed IDs (PMIDs) to PMC IDs and Full-Text URLs*

Description

This function converts PMIDs to PMC IDs, then fetches the full-text file URLs from the PMC Open Access service. It combines both steps into a single workflow.

Usage

```
pmid_to_ftp(
  pmids,
  batch_size = 200L,
  sleep = 0.5,
  verbose = FALSE,
  ncbi_key = NULL
)
```

Arguments

pmids	A character or numeric vector of PubMed IDs (PMIDs) to convert.
batch_size	An integer specifying the number of PMIDs to process per batch for ID conversion. Defaults to 200L. The NCBI API has limitations on batch sizes.
sleep	A numeric value specifying the number of seconds to pause between API requests for ID conversion (Step 1). Defaults to 0.5 seconds. For OA API calls (Step 2), sleep time is automatically adjusted based on rate limits: 0.11s with API key (10 req/sec), 0.34s without (3 req/sec).
verbose	Logical, whether to print progress messages. Defaults to FALSE.
ncbi_key	(Optional) NCBI API key for authenticated access.

Value

A data.table with columns:

- pmid: The input PubMed ID
- pmcid: The corresponding PMC ID
- doi: The corresponding DOI (NA if not available)
- url: The full HTTPS URL for downloading PMC full text

Results are filtered to only include rows with valid URLs (open access articles), ordered by PMID. Returns NULL with a message if the API is unavailable or returns invalid data.

Examples

```
if (interactive()) {
  # Convert PMIDs to PMC IDs and get full-text URLs
  result <- pmid_to_ftp(c("11250746", "11573492"))
}
```

---

pmid_to_pmc *Convert PubMed IDs (PMIDs) to PMC IDs*

Description

This function converts a vector of PubMed IDs (PMIDs) to their corresponding PubMed Central (PMC) IDs and DOIs using the NCBI ID Converter API.

Usage

```
pmid_to_pmc(pmids, batch_size = 200L, sleep = 0.5)
```

Arguments

pmids	A character or numeric vector of PubMed IDs (PMIDs) to convert.
batch_size	An integer specifying the number of PMIDs to process per batch. Defaults to 200L. The NCBI API has limitations on batch sizes.
sleep	A numeric value specifying the number of seconds to pause between API requests. Defaults to 0.5 seconds to respect API rate limits.

Value

A data.table with columns:

- pmid: The input PubMed ID
- pmcid: The corresponding PMC ID (NA if not available in PMC)
- doi: The corresponding DOI (NA if not available)

Results are ordered by PMID. Returns NULL with a message if the API is unavailable or returns invalid data.

Examples

```
if (interactive()) {
  # Convert a single PMID to PMC ID
  result <- pmid_to_pmc("12345678")

  # Convert multiple PMIDs
  pmids <- c("12345678", "23456789", "34567890")
  result <- pmid_to_pmc(pmids, batch_size = 10, sleep = 1)
}
```

---

pubtator_context *Add Sentence Context to PubTator Entities and Relations*

Description

Adds sentence identifiers and sentence-relative spans to PubTator entity mentions, then carries compact sentence anchors onto relation rows.

Usage

```
pubtator_context(pubtator)
```

Arguments

pubtator	A list returned by get_records(endpoint = "pubtator"), with entities and relations data.tables.

Value

A list with `entities`, `relations`, and `sentences` data.tables. Entity rows preserve their original `start`/`end` spans and gain `sentence_id`, `sentence_start`, and `sentence_end`. Relation rows gain role-specific entity labels and sentence anchors, plus `same_sentence` and `sentence_distance`.

`pubtator_cooccurrence` *Count PubTator Entity Co-occurrence from Sentence Context*

Description

Counts pairs of biomedical entities that co-occur in the same sentence (`window = 0`) or within `window` sentences of each other, using the contextualized entity table returned by `pubtator_context`. Co-occurrence is computed within each `pmid`/`tiab` passage; title and abstract sentence IDs are not compared to one another.

Usage

```
pubtator_cooccurrence(x, window = 0L, by = c("type", "entity"))
```

Arguments

`x`	A PubTator context list returned by `pubtator_context`, or a contextualized entity data.frame with `pmid`, `tiab`, `type`, `identifier`, `text`, and `sentence_id`.
`window`	Non-negative integer sentence distance. `0` counts entities in the same sentence; `n` counts entities whose sentences are at most `n` apart within the same `pmid`/`tiab` passage.
`by`	One of `"type"` (default) or `"entity"`. `"type"` aggregates counts by entity-type pair; `"entity"` aggregates by the specific `(type, identifier, text)` pair.

Details

Entities are de-duplicated to one mention per sentence before pairing, and pairs of the same entity (identical `type`, `identifier`, and `text`) are dropped.

Value

A data.table. With `by = "type"`: `type_x`, `type_y`, `n` (co-occurrence instances), and `n_pmids` (distinct documents), ordered by `n`. With `by = "entity"`: the same plus `identifier_x`/`text_x`/`identifier_y`/`text_y`.

Examples

```
## Not run:
pmids <- search_pubmed('"biomarker"[TiAb] AND "cancer"[TiAb]')

ctx <- pmids |>
  get_records(endpoint = "pubtator") |>
  pubtator_context()

ctx |> pubtator_cooccurrence(window = 0, by = "type")
ctx |> pubtator_cooccurrence(window = 1, by = "entity")

## End(Not run)
```

`pubtator_network` *Build a PubTator Relation Network with Evidence*

Description

Converts a `pubtator_context` result into a relation network: graph-ready `nodes` and `edges`, plus a lean `evidence` table that maps each edge back to the PubTator relation row and, when the endpoint mentions share a sentence, the supporting sentence.

Usage

```
pubtator_network(x)
```

Arguments

`x`	A list returned by `pubtator_context`, with `entities`, `relations`, and `sentences` data.tables.

Value

A named list with three `data.tables`:

`nodes` One row per normalized relation endpoint. Columns: `id`, `type`, `label`, `n_mentions`, and `n_pmids`. Entity identifiers are used when present; otherwise, including PubTator's `"-"` placeholder for unnormalized mentions, nodes fall back to `type:text`.

`edges` One row per directed PubTator relation edge. Columns: `from`, `to`, `relation_type`, `weight`, `n_pmids`, and `n_sentences`.

`evidence` One row per PubTator relation row. Columns: `from`, `to`, `relation_type`, `pmid`, `relation_id`, `same_sentence`, `sentence_distance`, and `sentence`. The sentence is populated only when the relation endpoints share a sentence.

See Also

`pubtator_context`, `pubtator_cooccurrence`

Examples

```
## Not run:
pmids <- search_pubmed('"doxorubicin"[TiAb] AND "cardiotoxicity"[TiAb]')

ctx <- pmids |>
  get_records(endpoint = "pubtator") |>
  pubtator_context()

net <- pubtator_network(ctx)
net$nodes
net$edges
net$evidence

## End(Not run)
```

`search_pubmed` *Search 'PubMed' Records*

Description

Performs a 'PubMed' search based on a query, optionally filtered by publication years. Returns a unique set of 'PubMed' IDs matching the query.

Usage

```
search_pubmed(
  x,
  start_year = NULL,
  end_year = NULL,
  retmax = 9999,
  use_pub_years = FALSE
)
```

Arguments

`x`	Character string, the search query.
`start_year`	Integer, the start year of publication date range (used if 'use_pub_years' is TRUE).
`end_year`	Integer, the end year of publication date range (used if 'use_pub_years' is TRUE).
`retmax`	Integer, maximum number of records to retrieve, defaults to 9999.
`use_pub_years`	Logical, whether to filter search by publication years, defaults to TRUE.

Value

Numeric vector of unique PubMed IDs.

Examples

```
if (interactive()) {
  ethnob1 <- search_pubmed("ethnobotany")
}
```

Index

∗ **datasets**
data_mesh_frequencies, 5

citation_network, 2
citation_snowball, 3

data_mesh_frequencies, 5, *9, 10*
data_mesh_thesaurus, 5
data_mesh_trees, 6

endpoint_info, 7

get_records, *2, 3*, 8, *9*, *12*

mesh_keyness, *5*, 9

pmid_to_ftp, *9*, 10
pmid_to_pmc, 11
pubtator_context, 12, *13, 14*
pubtator_cooccurrence, 13, *14*
pubtator_network, 14

search_pubmed, 15