Package 'jentre'

May 8, 2026

Title Toolkit for the 'Entrez' API

Version 0.1.0

Description Interact with the 'Entrez' API
hosted by the National Center for Biotechnology Information (NCBI),
<https://www.ncbi.nlm.nih.gov/books/NBK25499/>.
This package is focused on working with sequence metadata and links.
It handles pagination and compensates for some API limitations to simplify these tasks.
API calls are printed to the console to highlight how high-level queries are translated
into individual HTTP requests.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'count.R' 'process.R' 'fetch.R' 'id_set.R' 'id_set_validate.R'
'info.R' 'jentre-package.R' 'link.R' 'post.R' 'request.R'
'search.R' 'utils.R'

Depends R (>= 4.1.0)

Imports cli, glue, httr2, purrr, rlang (>= 1.1.0), vctrs (>= 0.7.0),
xml2

Suggests httpuv, testthat (>= 3.0.0), tibble, withr

Config/testthat/edition 3

URL https://github.com/cidm-ph/jentre,
https://cidm-ph.github.io/jentre/

BugReports https://github.com/cidm-ph/jentre/issues

NeedsCompilation no

Author Carl Suster [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-7021-9380>)

Maintainer Carl Suster <carl.suster@sydney.edu.au>

Repository CRAN

Date/Publication 2026-03-30 08:50:13 UTC

Contents

`check_id_set` *Check ID set is well formed*

Description

Check ID set is well formed

Usage

```
check_id_set(
  x,
  database = NULL,
  arg = rlang::caller_arg(x),
  call = rlang::caller_env()
)

check_id_list(x, arg = rlang::caller_arg(x), call = rlang::caller_env())

check_web_history(x, arg = rlang::caller_arg(x), call = rlang::caller_env())

entrez_database(x)
```

Arguments

`x`	ID set object.
`database`	name of intended database. If `NULL` the database name is not checked.
`arg`	name of argument to use in error reporting.
`call`	execution environment, for error reporting. See rlang::topic-error-call and the `call` argument of `cli::cli_abort()`.

Value

- For `check_*`, these function raise an error if the check fails.
- For `entrez_database()` the name of the database.

`efetch` *Fetch records from Entrez*

Description

Fetching can be slow, and Entrez will time out requests that take too long. This helper supports pagination if you specify `retmax`.

Usage

```
efetch(
  id_set,
  ...,
  retstart = 0L,
  retmax = NA,
  retmode = "xml",
  rettype = NULL,
  .method = NA,
  .cookies = NA,
  .paginate = 200L,
  .process = NA,
  .progress = "Fetching",
  .path = NULL,
  .call = rlang::current_env()
)
```

Arguments

`id_set`	ID set object.
`...`	additional API parameters (refer to Entrez documentation). Any set to `NULL` are removed.
`retstart`	integer: index of first result (starts from 0).
`retmax`	integer: maximum number of results to return. When `NA` this returns all results. When `NULL`, uses the Entrez default (typically 20). Note that when using pagination with web history, it is possible that slightly more than `retmax` results will be returned.
`retmode`	character: requested document file format.
`rettype`	character: requested document type.
`.method`	HTTP verb. If `NA`, a sensible default is chosen based on the request parameters.

.cookies	path to persist cookies. If NULL, cookies are not added to the request. For helper functions: when NA, a temporary file is created (in this case only, the temporary file will be cleaned up once all requests are performed).
.paginate	controls how multiple API requests are used to complete the call. Pagination is performed using the retstart and retmax API parameters. When set to an integer, no more than .pagniate items will be requested per API call. When FALSE or 0, only one API request is sent.
.process	function that processes the API results. Can be a function or builtin processor as described in process. Additional builtin processors are available: • "uilist" to extract a list of IDs (suitable for rettype = "uilist"), • NA to use a sensible choice based on parameters. In particular, for "uilist" requests, it will return an id_list object.
.progress	controls progress bar; see the progress argument of httr2::req_perform_iterative().
.path	path specification for saving raw responses. See path argument of httr2::req_perform_iterative().
.call	call environment to use in error messages/traces. See rlang::topic-error-call and the call argument of cli::cli_abort(). You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

Combined output of .process from each page of results. For the default where .process does nothing, this will be a list of XML documents. For other choices, it can be a vector, list, or data frame.

See Also

Other API methods: einfo(), elink(), entrez_validate(), epost(), esearch(), esummary()

Examples

```
library(xml2)
id_set <- id_list("sra", c("39889350", "39889348", "39889347"))

## Not run:
efetch(id_set)
# -> efetch db="sra" retstart="0" retmax="3" retmode="xml" * id="39889350,...,39889347"[3]
# [[1]]
# {xml_document}
# <EXPERIMENT_PACKAGE_SET>
# [1] <EXPERIMENT_PACKAGE>\n  <EXPERIMENT accession="SRX29833825" alias="24-MYP-0283_50325"> ...
# [2] <EXPERIMENT_PACKAGE>\n  <EXPERIMENT accession="SRX29833823" alias="24-MYP-0273_50325"> ...
# [3] <EXPERIMENT_PACKAGE>\n  <EXPERIMENT accession="SRX29833822" alias="24-MYP-0270_50325"> ...

extract_alias <- function(document) {
  xml_find_all(document, "//EXPERIMENT/@alias") |> xml_text()
}
efetch(id_set, .process = extract_alias)
# -> efetch db="sra" retstart="0" retmax="3" retmode="xml" * id="39889350,...,39889347"[3]
# [1] "24-MYP-0283_50325" "24-MYP-0273_50325" "24-MYP-0270_50325"
## End(Not run)
```

einfo *Get details about Entrez databases*

Description

These functions call the EInfo endpoint. `einfo()` provides the number of entries in the databases, the name and description, list of terms usable in the query syntax, and list of link names usable with the ELink endpoint.

Usage

```
einfo(db, ..., retmode = "xml", version = "2.0", .call = rlang::current_env())

einfo_databases(..., retmode = "xml", .call = rlang::current_env())
```

Arguments

db	name of database to provide information about.
...	additional API parameters (refer to Entrez documentation). Any set to NULL are removed.
retmode	response format.
version	response format version.
.call	call environment to use in error messages/traces. See rlang::topic-error-call and the call argument of cli::cli_abort(). You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

Character vector of database names for `einfo_databases()`. An XML document with root node <eInfoResult> for `einfo()`.

See Also

Other API methods: efetch(), elink(), entrez_validate(), epost(), esearch(), esummary()

Examples

```
library(xml2)

## Not run:
einfo("sra") |> xml_find_first("//Description") |> xml_text()
# [1] "SRA Database"
## End(Not run)
```

elink *ELink API for fetching links between databases*

Description

`elink()` offers direct access to the ELink API endpoint, which has many different input and output formats depending on parameters. If you just want a one-to-one mapping of `neighbor` links, use `elink_map()`, which handles this for you.

Usage

```
elink(
  id_set,
  db,
  ...,
  retmode = "xml",
  cmd = NA,
  .paginate = 100L,
  .process = NA,
  .method = NA,
  .multi = "explode",
  .progress = TRUE,
  .cookies = NA,
  .path = NULL,
  .call = current_env()
)

elink_map(id_set, db, ..., .cookies = NA, .path = NULL, .call = current_env())
```

Arguments

`id_set`	ID set object.
`db`	target database name.
`...`	additional API parameters (refer to Entrez documentation). Any set to `NULL` are removed.
`retmode`	response format.
`cmd`	ELink command. If `NA` either `"neighbor"` or `"neighbor_history"` will be used based on the type of input.
`.paginate`	maximum number of UIDs to submit per request. `.paginate` only applies when `id_set` is an explicit list. Set to `FALSE` to prevent batching.
`.process`	function that processes the API results. Can be a function or builtin processor as described in `process`. Additional builtin processors are available: `"sets"` to get a data frame of ID set objects, `"flat"` to get a data frame of UIDs, or `NA` to use a sensible choice based on parameters.

.method	HTTP verb. For "POST", any params with vector values (usually just id) are sent in the request body instead of the URL.
.multi	controls how repeated params are handled (see httr2::req_url_query()).
.progress	controls progress bar; see the progress argument of httr2::req_perform_iterative().
.cookies	path to persist cookies. If NULL, cookies are not added to the request. For helper functions: when NA, a temporary file is created (in this case only, the temporary file will be cleaned up once all requests are performed).
.path	path specification for saving raw responses. See path argument of httr2::req_perform_iterative().
.call	call environment to use in error messages/traces. See rlang::topic-error-call and the call argument of cli::cli_abort(). You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

concatenated output of .process. For elink(.process = "sets") a data frame with columns

from Source link set.

to Target link set.

linkname Link name (see https://eutils.ncbi.nlm.nih.gov/entrez/query/static/entrezlinks.html or einfo).

For elink_map() and elink(.process = "flat") a data frame with columns

db_from Source database name.

id_from Source identifier. Can be a list column depending on how elink was called.

db_to Target database name.

linkname Link name (see https://eutils.ncbi.nlm.nih.gov/entrez/query/static/entrezlinks.html or einfo).

id_to Target identifier. In general this will be a list column.

One-to-one mapping

Note that some ways of calling this API on multiple UIDs result in the one-to-one association of the input and output sets getting lost. The way around this is to specify each ID as a separate parameter rather than a single comma-separated param. This is handled by the default choice of .multi = "explode". When using a web history token as input, there is no corresponding way to ensure one-to-one mapping. To ensure that the result is always one-to-one, use elink_map(), which may make several API requests to achieve the result.

See Also

Other API methods: efetch(), einfo(), entrez_validate(), epost(), esearch(), esummary()

Examples

```
id_set <- id_list("sra", c("39889350", "39889348", "39889347"))

## Not run:
links <- elink(id_set, "bioproject", linkname = "sra_bioproject")
# -> elink db="bioproject" dbfrom="sra" retmode="xml" cmd="neighbor" linkname="sra_bioproject"
#    * id="39889350" id="39889348" id="39889347"
links
# # A tibble: 3 x 5
#   db_from id_from   db_to      linkname       id_to
#   <chr>   <list>    <chr>      <chr>          <list>
# 1 sra     <chr [1]> bioproject sra_bioproject <chr [1]>
# 2 sra     <chr [1]> bioproject sra_bioproject <chr [1]>
# 3 sra     <chr [1]> bioproject sra_bioproject <chr [1]>

links[c("id_from", "id_to")] |> igraph::graph_from_data_frame()
# IGRAPH a807b82 DN-- 4 3 --
# + attr: name (v/c)
# + edges from a807b82 (vertex names):
# [1] 39889350->1241475 39889348->1241475 39889347->1241475
## End(Not run)
```

`entrez_count` *Count the number of entries in an ID set*

Description

If `id_set` is an `id_list` then this is equivalent to `length()`. If it is a `web_history`, this may involve an Entrez API call to get the number of entries. In this case the result is cached so that subsequent calls don't hit the API again.

Usage

```
entrez_count(id_set, .call = current_env())
```

Arguments

`id_set`	an ID set object.
`.call`	call environment to use in error messages/traces. See rlang::topic-error-call and the `call` argument of `cli::cli_abort()`. You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

integer number of entries.

Examples

```
id_set <- id_list("sra", c("39889350", "39889348", "39889347"))
entrez_count(id_set)
```

`entrez_request` *Construct a request to the Entrez API*

Description

This is a low-level helper that builds a request object but does not perform the request. In general you'll likely use higher-level methods like `efetch()` instead.

Usage

```
entrez_request(
  endpoint,
  ...,
  .method = "GET",
  .multi = "comma",
  .cookies = NULL,
  .verbose = getOption("jentre.verbose", default = TRUE),
  .call = current_env()
)

entrez_api_key(default = NULL)
```

Arguments

`endpoint`	Entrez endpoint name (e.g. `"efetch.fcgi"`).
`...`	additional API parameters (refer to Entrez documentation). Any set to `NULL` are removed.
`.method`	HTTP verb. For `"POST"`, any params with vector values (usually just `id`) are sent in the request body instead of the URL.
`.multi`	controls how repeated params are handled (see `httr2::req_url_query()`).
`.cookies`	path to persist cookies. If `NULL`, cookies are not added to the request. For helper functions: when `NA`, a temporary file is created (in this case only, the temporary file will be cleaned up once all requests are performed).
`.verbose`	logical: when TRUE logs all API requests as messages in a compact format. This uses a summarised format that does not include the request body for POST. Use normal httr verbosity controls (e.g. `httr2::local_verbosity()`) to override this behaviour and see more details.
`.call`	call environment to use in error messages/traces. See rlang::topic-error-call and the `call` argument of `cli::cli_abort()`. You only need to specify this in internal helper functions that don't need to be mentioned in error messages.
`default`	default value to return if no global configuration is found.

Details

`email`, `tool`, and `api_key` have default values but these can be overridden, or can be removed by setting them to `NULL`.

Value

- for `entrez_request()` an `httr2::request` object.
- for `entrez_api_key()`, the API key as a character, or `default` if no global config exists.

API limits

The Entrez APIs are rate limited. Requests in this package respect the API headers returned by Entrez. Without an API key you will be rate limited more aggressively, so it is recommended to obtain an API key. `jentre` searches for the API key in the following order:

- the API parameter `entrez_key` provided to any API request function,
- the option `"jentre.api_key"`, then
- the environment variable `ENTREZ_KEY`.

You can check the value is found properly using `entrez_api_key()`. If no API key is set, a warning will be displayed. This can be suppressed by setting the option `"jentre.silence_api_warning"` to `TRUE`.

Examples

```
library(httr2)

req <- entrez_request("esearch.fcgi", db = "nucleotide", term = "biomol+trna[prop]")
## Not run:
# You'll need to perform the request with httr2 and parse it yourself:
req_perform(req) |> resp_body_xml()
## End(Not run)
```

`entrez_validate` *Look up accessions and other IDs on Entrez*

Description

Passes the provided IDs through Entrez which has the effect of normalising the accepted UIDs, and removing invalid UIDs. For web history lists, this forces results to be freshly downloaded (unlike `as_id_list()` which can use cached results).

Usage

```
entrez_validate(id_set, .paginate = 5000L, .path = NULL, .call = current_env())
```

Arguments

id_set	an id_list object.
.paginate	controls how multiple API requests are used to complete the call. Pagination is performed using the retstart and retmax API parameters. When set to an integer, no more than .pagniate items will be requested per API call. When FALSE or 0, only one API request is sent.
.path	path specification for saving raw responses.
.call	call environment to use in error messages/traces. See rlang::topic-error-call and the call argument of cli::cli_abort(). You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

id_list object

See Also

Other API methods: efetch(), einfo(), elink(), epost(), esearch(), esummary()

Examples

```
id_set <- id_list("sra", c("SRX29833825", "SRX29833823", "SRX29833822"))
## Not run:
entrez_validate(id_set)
# <entrez/sra[3]>
# [1] 39889350 39889348 39889347
## End(Not run)
```

---

epost *Register UIDs with the Entrez history server*

Description

Register UIDs with the Entrez history server

Usage

```
epost(id_set, ..., WebEnv = NULL, .path = NULL, .call = rlang::current_env())
```

Arguments

id_set	an id_list object.
...	additional API parameters (refer to Entrez documentation). Any set to NULL are removed.
WebEnv	either a character to pass on as-is, or a web_history object.
.path	path specification for saving raw responses.

`.call` call environment to use in error messages/traces. See rlang::topic-error-call and the `call` argument of `cli::cli_abort()`. You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

A `web_history` object usable with other API functions.

See Also

Other API methods: `efetch()`, `einfo()`, `elink()`, `entrez_validate()`, `esearch()`, `esummary()`

Examples

```
id_set <- id_list("sra", c("39889350", "39889348", "39889347"))

## Not run:
epost(id_set)
# -> epost db="sra" * id="39889350,...,39889347"[3]
# <entrez@/sra[1]>
# [1] MCID_69c36.#1[3]
## End(Not run)
```

---

esearch *Search Entrez databases*

Description

The search term field names are documented in the EInfo API endpoint: see `einfo()`.

Usage

```
esearch(
  term,
  db,
  ...,
  retstart = 0L,
  retmax = NA,
  retmode = "xml",
  rettype = "uilist",
  usehistory = is.null(retmax) || is.na(retmax),
  WebEnv = NULL,
  query_key = NULL,
  .cookies = NA,
  .paginate = 10000L,
  .progress = "ESearch",
  .path = NULL,
  .verbose = getOption("jentre.verbose", default = TRUE),
  .call = current_env()
)
```

Arguments

term	search query.
db	Entrez database name.
...	additional API parameters (refer to Entrez documentation). Any set to NULL are removed.
retstart	integer: index of first result (starts from 0). Ignored when usehistory is TRUE.
retmax	integer: maximum number of results to return. When NA this returns all results. When NULL, uses the Entrez default (typically 20). Note that it is possible that slightly more than retmax results will be returned when paginating. Ignored when usehistory is TRUE.
retmode	character: currently only "xml" is supported.
rettype	character: currently only "uilist" is supported.
usehistory	logical: when TRUE use the history server to return the result.
WebEnv, query_key	either characters to pass on as-is, or web_history objects.
.cookies	path to persist cookies. If NULL, cookies are not added to the request. For helper functions: when NA, a temporary file is created (in this case only, the temporary file will be cleaned up once all requests are performed).
.paginate	controls how multiple API requests are used to complete the call. Pagination is performed using the retstart and retmax API parameters. When set to an integer, no more than .pagniate items will be requested per API call. When FALSE or 0, only one API request is sent. Ignored when usehistory is TRUE.
.progress	controls progress bar; see the progress argument of httr2::req_perform_iterative().
.path	path specification for saving raw responses. See path argument of httr2::req_perform_iterative().
.verbose	logical: when TRUE logs all API requests as messages in a compact format. This uses a summarised format that does not include the request body for POST. Use normal httr verbosity controls (e.g. httr2::local_verbosity()) to override this behaviour and see more details.
.call	call environment to use in error messages/traces. See rlang::topic-error-call and the call argument of cli::cli_abort(). You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

An id set object (either a web_history or an id_list).

See Also

https://www.ncbi.nlm.nih.gov/books/NBK25499/#chapter4.ESearch

Other API methods: efetch(), einfo(), elink(), entrez_validate(), epost(), esummary()

Examples

```
## Not run:
esearch("mpox virus[orgn]", "biosample")
# -> esearch db="biosample" term="mpox virus[orgn]" retmode="xml" rettype="uilist" usehistory="y"
#  i eSearch query "\"Monkeypox virus\"[Organism]" has 7189 results
# <entrez@/biosample[1]>
# [1] MCID_69c36.#1[7189]

## End(Not run)
```

`esummary` *Fetch document summaries from Entrez*

Description

ESummary is faster than EFetch because it only interacts with the frontend rather than the full database. It contains more limited information.

Usage

```
esummary(
  id_set,
  ...,
  retstart = 0L,
  retmax = NA,
  retmode = "xml",
  version = "2.0",
  .method = NA,
  .cookies = NA,
  .paginate = 5000L,
  .process = "identity",
  .progress = "Fetching summaries",
  .path = NULL,
  .call = rlang::current_env()
)
```

Arguments

`id_set`	ID set object.
`...`	additional API parameters (refer to Entrez documentation). Any set to `NULL` are removed.
`retstart`	integer: index of first result (starts from 0).
`retmax`	integer: maximum number of results to return. When `NA` this returns all results. When `NULL`, uses the Entrez default (typically 20). Note that when using pagination with web history, it is possible that slightly more than `retmax` results will be returned.

retmode	character: requested document file format.
version	character: requested format version.
.method	HTTP verb. If NA, a sensible default is chosen based on the request parameters.
.cookies	path to persist cookies. If NULL, cookies are not added to the request. For helper functions: when NA, a temporary file is created (in this case only, the temporary file will be cleaned up once all requests are performed).
.paginate	controls how multiple API requests are used to complete the call. Pagination is performed using the retstart and retmax API parameters. When set to an integer, no more than .pagniate items will be requested per API call. When FALSE or 0, only one API request is sent.
.process	function that processes the API results. Can be a function or builtin processor as described in process. Additional builtin processors are available: • "uilist" to extract a list of IDs (suitable for rettype = "uilist"), • NA to use a sensible choice based on parameters. In particular, for "uilist" requests, it will return an id_list object.
.progress	controls progress bar; see the progress argument of httr2::req_perform_iterative().
.path	path specification for saving raw responses. See path argument of httr2::req_perform_iterative().
.call	call environment to use in error messages/traces. See rlang::topic-error-call and the call argument of cli::cli_abort(). You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Value

Combined output of .process from each page of results. For the default where .process does nothing, this will be a list of XML documents. For other choices, it can be a vector, list, or data frame.

See Also

Other API methods: efetch(), einfo(), elink(), entrez_validate(), epost(), esearch()

---

id_list *Entrez identifier sets*

Description

Many Entrez APIs accept either a UID list or tokens that point to a result stored on its history server. The classes here wrap these and keep track of the database name that the identifiers belong to. Most of the API helpers in this package are generic over the type of ID set and so can be used the same way with either type.

Usage

```
id_list(db, ids = character())

web_history(db, WebEnv, query_key, length = NA)

is_id_set(x)

is_id_list(x)

is_web_history(x)

as_id_list(x, .paginate = 5000L, .path = NULL, .call = current_env())
```

Arguments

db	name of the associated Entrez database (e.g. "biosample").
ids	UIDs, coercible to a character vector (can be accessions or GI numbers).
query_key, WebEnv	history server tokens returned by another Entrez API call.
length	number of UIDs in the set, if known.
x	object to test or convert.
.paginate	controls how multiple API requests are used to complete the call. Pagination is performed using the retstart and retmax API parameters. When set to an integer, no more than .pagniate items will be requested per API call. When FALSE or 0, only one API request is sent. Ignored when usehistory is TRUE.
.path	path specification for saving raw responses. See path argument of httr2::req_perform_iterative().
.call	call environment to use in error messages/traces. See rlang::topic-error-call and the call argument of cli::cli_abort(). You only need to specify this in internal helper functions that don't need to be mentioned in error messages.

Details

It usually will not make sense to create web_history() objects directly - they are short-lived pointers to results on the Entrez history server and are created by other API calls.

id_list is a vector and can be manipulated to take subsets (e.g. id_set[1:10] or tail(id_set)).

web_history is an opaque reference to an ID list stored on the Entrez history server. Through the course of API calls, information about the length or the actual list of IDs may be discovered and cached, avoiding subsequent API calls. as_id_list() can be used to extract the list of IDs.

Convert id_list to web_history with epost(). Convert web_history to id_list with as_id_list().

Value

- For id_list() and as_id_list() an id_list vector.
- For web_history() a web_history object.
- For is_id_set(), is_id_list(), and is_web_history() a logical.

See Also

entrez_validate() and entrez_count()

Examples

```
bioprojects <- id_list("bioproject", c("1241475"))
```

---

topic-process *Process API results*

Description

Function to turn the parsed response document into meaningful data. It must accept one argument, doc, the parsed response document. The return value must be compatible with vctrs::list_combine(), e.g. a vector, list, or data frame.

Details

API results are parsed based on the retmode parameter. XML documents will be parsed into xml2::xml_document objects and an error will be raised if it contains an <ERROR> node.

Builtin processors can be referred to by name instead of specifying your own function. Some helpers provide additional processors, but these are always available:

- "identity": Puts the parsed output document into a list. Where multiple requests are made (e.g. using the batched APIs like efetch()) these will then be concatenated into a single list.

Index