Package 'rcoreoa'

October 14, 2022

Type Package

Title Client for the CORE API

Description Client for the CORE API (<https://core.ac.uk/docs/>). CORE (<https://core.ac.uk>) aggregates open access research outputs from repositories and journals worldwide and make them available to the public.

Version 0.4.0

License MIT + file LICENSE

URL https://docs.ropensci.org/rcoreoa, https://github.com/ropensci/rcoreoa

BugReports https://github.com/ropensci/rcoreoa/issues

VignetteBuilder knitr

Encoding UTF-8

Language en-US

Imports crul, jsonlite, pdftools, hoardr

Suggests roxygen2 (>= 7.1.0), testthat, knitr, rmarkdown, rcrossref, vcr

RoxygenNote 7.1.0

X-schema.org-applicationCategory Literature

X-schema.org-keywords text-ming, literature, pdf, publications, citations, full-text, metadata

X-schema.org-isPartOf https://ropensci.org

NeedsCompilation no

Author Scott Chamberlain [aut, cre], Aristotelis Charalampous [ctb], Simon Goring [ctb]

Maintainer Scott Chamberlain <myrmecocystus+r@gmail.com>

Repository CRAN

Date/Publication 2020-07-07 17:00:02 UTC

R topics documented:

`rcoreoa-package` *rcoreoa - CORE R client*

Description

CORE is a web service for metadata on scholarly journal articles. Find CORE at `https://core.ac.uk` and their API docs at `https://core.ac.uk/docs/`.

Package API

Each API endpoint has two functions that interface with it - a higher level interface and a lower level interface. The lower level functions have an underscore (_) at the end of the function name, while their corresponding higher level companions do not. The higher level functions parse to list/data.frame's (as tidy as possible). Lower level functions give back JSON (character class) thus are slightly faster not spending time on parsing to R structures.

- `core_articles()` / `core_articles_()` - get article metadata
- `core_articles_history()` / `core_articles_history_()` - get article history metadata
- `core_articles_pdf()` / `core_articles_pdf_()` - download article PDF, and optionally extract text
- `core_journals()` / `core_journals_()` - get journal metadata
- `core_repos()` / `core_repos_()` - get repository metadata
- `core_repos_search()` / `core_repos_search_()` - search for repositories
- `core_search()` / `core_search_()` - search articles
- `core_advanced_search()` - advanced search of articles

Authentication

You'll need a CORE API token/key to use this package. Get one at `https://core.ac.uk/api-keys/register`

Pagination

Note that you are limited to a maximum of 100 results for the search functions; use combination of `page` and `limit` parameters to paginate through results. For example:

```
x1 <- core_search(query = 'ecology', limit = 100, page = 1)
x2 <- core_search(query = 'ecology', limit = 100, page = 2)
```

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Aristotelis Charalampous

`core_advanced_search` *Advanced Search CORE*

Description

Advanced Search CORE

Usage

```
core_advanced_search(
  ...,
  page = 1,
  limit = 10,
  key = NULL,
  parse = TRUE,
  .list = list()
)

core_query(..., op = "AND")
```

Arguments

`...`	for `core_query()`, query fields, see Details. for `core_advanced_search()` any number of queries, results of calling `core_query()`. Required. See Details.
`page`	(character) page number (default: 1), optional
`limit`	(character) records to return (default: 10, minimum: 10, maximum: 100), optional
`key`	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name `CORE_KEY` or an R option as `core_key`. See `?Startup` for how to work with env vars and R options
`parse`	(logical) Whether to parse to list (`FALSE`) or data.frame (`TRUE`). Default: `TRUE`

.list	alternative to passing core_query() calls to ...; just create a list of them and pass to this parameter; easier for programming with
op	(character) operator to combine multiple fields. options: AND, OR

Details

query should be one or more calls to core_query(), (at least one is required):

- title
- description
- fullText
- authors
- publisher: string, to be used as an absolute match against the publisher name metadata field
- repositories.id: repository id
- repositories.name: repository name
- doi: string, to be used as an absolute match against the repository name metadata field (all other fields will be ignored if included)
- oai
- identifiers
- language.name: string, to filter against languages specified in https://en.wikipedia.org/wiki/ISO_639-1
- year: year to filter to

core_advanced_search does the HTTP request and parses, while core_advanced_search_ just does the HTTP request, gives back JSON as a character string

Value

data.frame with the following columns:

- status: string, which will be 'OK' or 'Not found' or 'Too many queries' or 'Missing parameter' or 'Invalid parameter' or 'Parameter out of bounds'
- totalHits: integer, Total number of items matching the search criteria
- data: list, a list of relevant resource

References

https://core.ac.uk/docs/#!/all/searchBatch

Examples

```
## Not run:
## compose queries
core_query(title="psychology", year=2014)
core_query(title="psychology", year=2014, op="OR")
core_query(identifiers='"oai:aura.abdn.ac.uk:2164/3837"',
  identifiers='"oai:aura.abdn.ac.uk:2164/3843"', op="OR")
```

```
## do actual searches
core_advanced_search(
  core_query(identifiers='"oai:aura.abdn.ac.uk:2164/3837"',
    identifiers='"oai:aura.abdn.ac.uk:2164/3843"', op="OR")
)

res <- core_advanced_search(
  core_query(title="psychology"),
  core_query(doi='"10.1186/1471-2458-6-309"'),
  core_query(language.name="german")
)
res

## End(Not run)
```

`core_articles` *Get articles*

Description

Get articles

Usage

```
core_articles(
  id,
  metadata = TRUE,
  fulltext = FALSE,
  citations = FALSE,
  similar = FALSE,
  duplicate = FALSE,
  urls = FALSE,
  extractedUrls = FALSE,
  faithfulMetadata = FALSE,
  key = NULL,
  method = "GET",
  parse = TRUE,
  ...
)

core_articles_(
  id,
  metadata = TRUE,
  fulltext = FALSE,
  citations = FALSE,
  similar = FALSE,
  duplicate = FALSE,
```

```
  urls = FALSE,
  extractedUrls = FALSE,
  faithfulMetadata = FALSE,
  key = NULL,
  method = "GET",
  ...
)
```

Arguments

id	(integer) CORE ID of the article that needs to be fetched. Required
metadata	Whether to retrieve the full article metadata or only the ID. Default: TRUE
fulltext	Whether to retrieve full text of the article. Default: FALSE
citations	Whether to retrieve citations found in the article. Default: FALSE
similar	Whether to retrieve a list of similar articles. Default: FALSE Because the similar articles are calculated on demand, setting this parameter to true might slightly slow down the response time query
duplicate	Whether to retrieve a list of CORE IDs of different versions of the article. Default: FALSE
urls	Whether to retrieve a list of URLs from which the article can be downloaded. This can include links to PDFs as well as HTML pages. Default: FALSE
extractedUrls	Whether to retrieve a list of URLs which were extracted from the article fulltext. Default: FALSE. This parameter is not available in CORE API v2.0 beta
faithfulMetadata	Whether to retrieve the raw XML metadata of the article. Default: FALSE
key	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name CORE_KEY or an R option as core_key. See ?Startup for how to work with env vars and R options
method	(character) one of 'GET' (default) or 'POST'
parse	(logical) Whether to parse to list (FALSE) or data.frame (TRUE). Default: TRUE
...	Curl options passed to HttpClient

Details

core_articles does the HTTP request and parses, while core_articles_ just does the HTTP request, gives back JSON as a character string

These functions take one article ID at a time. Use lapply/loops/etc for many ids

References

https://core.ac.uk/docs/#!/articles/getArticleByCoreIdBatch https://core.ac.uk/docs/#!/articles/getArticleByCoreId

Examples

```
## Not run:
core_articles(id = 21132995)
core_articles(id = 21132995, fulltext = TRUE)
core_articles(id = 21132995, citations = TRUE)

# when passing >1 article ID
ids <- c(20955435, 21132995, 21813171, 22815670, 23828884,
   23465055, 23831838, 23923390, 22559733)
## you can use method="GET" with lapply or similar
res <- lapply(ids, core_articles)
vapply(res, "[[", "", c("data", "datePublished"))

## or use method="POST" passing all at once
res <- core_articles(ids, method = "POST")
head(res$data)
res$data$authors

# just http request, get text back
core_articles_(id = '21132995')
## POST, can pass many at once
core_articles_(id = ids, method = "POST")

## End(Not run)
```

---

core_articles_dedup *Article deduplication*

Description

Article deduplication

Usage

```
core_articles_dedup(
  doi = NULL,
  title = NULL,
  year = NULL,
  description = NULL,
  fulltext = NULL,
  identifier = NULL,
  repositoryId = NULL,
  key = NULL,
  parse = TRUE,
  ...
)

core_articles_dedup_(
```

```
  doi = NULL,
  title = NULL,
  year = NULL,
  description = NULL,
  fulltext = NULL,
  identifier = NULL,
  repositoryId = NULL,
  key = NULL,
  ...
)
```

Arguments

doi	(character) the DOI for which the duplicates will be identified. optional
title	(character) title to match when looking for duplicate articles. Either year or description should also be supplied if this parameter is supplied. optional
year	(character) year the article was published. Only used in combination with the value for title parameter. optional
description	(character) abstract for an article based on which its duplicates will be found. This should be more than 500 characters. Value for the title parameter should also be supplied if this is supplied. optional
fulltext	(character) Full text for an article based on which its duplicates will be found. optional
identifier	(character) CORE ID of the article for which the duplicates will be identified. optional
repositoryId	(character) Limit the duplicates search to particular repository id. optional
key	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name CORE_KEY or an R option as core_key. See ?Startup for how to work with env vars and R options
parse	(logical) Whether to parse to list (FALSE) or data.frame (TRUE). Default: TRUE
...	Curl options passed to HttpClient

References

https://core.ac.uk/docs/#!/articles/nearDuplicateArticles

Examples

```
## Not run:
core_articles_dedup(title = "Managing exploratory innovation", year = 2010)
core_articles_dedup_(title = "Managing exploratory innovation", year = 2010)

ab = 'Neonicotinoid seed dressings have caused concern world-wide. We use
large field experiments to assess the effects of neonicotinoid-treated crops
on three bee species across three countries (Hungary, Germany, and the
```

```
United Kingdom). Winter-sown oilseed rape was grown commercially with
either seed coatings containing neonicotinoids (clothianidin or
thiamethoxam) or no seed treatment (control). For honey bees, we found
both negative (Hungary and United Kingdom) and positive (Germany)
effects during crop flowering. In Hungary, negative effects on honey
bees (associated with clothianidin) persisted over winter and resulted
in smaller colonies in the following spring (24% declines). In wild
bees (Bombus terrestris and Osmia bicornis), reproduction was
negatively correlated with neonicotinoid residues. These findings
point to neonicotinoids causing a reduced capacity of bee species
to establish new populations in the year following exposure.'
core_articles_dedup(
 title = "Country-specific effects of neonicotinoid pesticides on honey bees and wild bees",
  description = ab, verbose = TRUE)

## End(Not run)
```

`core_articles_history` *Get article history*

Description

Get article history

Usage

```
core_articles_history(id, page = 1, limit = 10, key = NULL, parse = TRUE, ...)

core_articles_history_(id, page = 1, limit = 10, key = NULL, ...)
```

Arguments

`id`	(integer) CORE ID of the article that needs to be fetched. One or more. Required
`page`	(character) page number (default: 1), optional
`limit`	(character) records to return (default: 10, minimum: 10, maximum: 100), optional
`key`	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name `CORE_KEY` or an R option as `core_key`. See `?Startup` for how to work with env vars and R options
`parse`	(logical) Whether to parse to list (`FALSE`) or data.frame (`TRUE`). Default: `TRUE`
`...`	Curl options passed to `HttpClient`

Details

`core_articles_history` does the HTTP request and parses, while `core_articles_history_` just does the HTTP request, gives back JSON as a character string

Value

core_articles_history_ returns a JSON string on success. core_articles_history returns a list (equal to id length) where each element is a list of length two with elements for data and status of the request; on failure the data slot is NULL.

References

https://core.ac.uk/docs/#!/articles/getArticleHistoryByCoreId

Examples

```
## Not run:
core_articles_history(id = 21132995)

ids <- c(20955435, 21132995, 21813171, 22815670, 14045109, 23828884,
   23465055, 23831838, 23923390, 22559733)
res <- core_articles_history(ids)
vapply(res, function(z) z$data$datetime[1], "")

# just http request, get text back
core_articles_history_(21132995)

## End(Not run)
```

---

core_articles_pdf *Download article pdf*

Description

Download article pdf

Usage

```
core_articles_pdf(id, key = NULL, overwrite = FALSE, ...)

core_articles_pdf_(id, key = NULL, overwrite = FALSE, ...)
```

Arguments

id	(integer) CORE ID of the article that needs to be fetched. One or more. Required
key	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name CORE_KEY or an R option as core_key. See ?Startup for how to work with env vars and R options
overwrite	(logical) overwrite file or not if already on disk. Default: FALSE
...	Curl options passed to crul::HttpClient()

Details

`core_articles_pdf` does the HTTP request and parses PDF to text, while `core_articles_pdf_` just does the HTTP request and gives back the path to the file

If you get a message like `Error: Not Found (HTTP 404)`, that means a PDF was not found. That is, it does not exist. That is, there is no PDF associated with the article ID you searched for. This is the correct behavior, and nothing is wrong with this function or this package. We could do another web request to check if the id you pass in has a PDF or not first, but that's another request, slowing this function down.

Value

`core_articles_pdf_` returns a file path on success. When many IDs passed to `core_articles_pdf` it returns a list (equal to length of IDs) where each element is a character vector of length equal to number of pages in the PDF; but on failure throws warning and returns NULL. When single ID apssed to `core_articles_pdf` it returns a character vector of length equal to number of pages in the PDF, but on failure stops with message

References

https://core.ac.uk/docs/#!/articles/getArticlePdfByCoreId

Examples

```
## Not run:
# just http request, get file path back
core_articles_pdf_(11549557)

# get paper and parse to text
core_articles_pdf(11549557)

ids <- c(11549557, 385071)
res <- core_articles_pdf(ids)
cat(res[[1]][1])
cat(res[[2]][1])

## End(Not run)
```

`core_articles_search` *Search CORE articles*

Description

Search CORE articles

Usage

```
core_articles_search(
  query,
  metadata = TRUE,
  fulltext = FALSE,
  citations = FALSE,
  similar = FALSE,
  duplicate = FALSE,
  urls = FALSE,
  faithfulMetadata = FALSE,
  page = 1,
  limit = 10,
  key = NULL,
  parse = TRUE,
  ...
)

core_articles_search_(
  query,
  metadata = TRUE,
  fulltext = FALSE,
  citations = FALSE,
  similar = FALSE,
  duplicate = FALSE,
  urls = FALSE,
  faithfulMetadata = FALSE,
  page = 1,
  limit = 10,
  key = NULL,
  ...
)
```

Arguments

query	(character) query string, required
metadata	(logical) Whether to retrieve the full article metadata or only the ID. Default: TRUE
fulltext	(logical) Whether to retrieve full text of the article. Default: FALSE
citations	(logical) Whether to retrieve citations found in the article. Default: FALSE
similar	(logical) Whether to retrieve a list of similar articles. Default: FALSE. Because the similar articles are calculated on demand, setting this parameter to true might slightly slow down the response time
duplicate	(logical) Whether to retrieve a list of CORE IDs of different versions of the article. Default: FALSE
urls	(logical) Whether to retrieve a list of URLs from which the article can be downloaded. This can include links to PDFs as well as HTML pages. Default: FALSE

faithfulMetadata
: (logical) Returns the records raw XML metadata from the original repository. Default: FALSE

page
: (character) page number (default: 1), optional

limit
: (character) records to return (default: 10, minimum: 10, maximum: 100), optional

key
: A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name CORE_KEY or an R option as core_key. See ?Startup for how to work with env vars and R options

parse
: (logical) Whether to parse to list (FALSE) or data.frame (TRUE). Default: TRUE

...
: Curl options passed to HttpClient

Details

core_articles_search does the HTTP request and parses, while core_articles_search_ just does the HTTP request, gives back JSON as a character string

References

https://core.ac.uk/docs/#!/all/search

Examples

```
## Not run:
core_articles_search(query = 'ecology')
core_articles_search(query = 'ecology', parse = FALSE)
core_articles_search(query = 'ecology', limit = 12)
out = core_articles_search(query = 'ecology', fulltext = TRUE)

core_articles_search_(query = 'ecology')
jsonlite::fromJSON(core_articles_search_(query = 'ecology'))

# post request
query <- c('data mining', 'semantic web')
res <- core_articles_search(query)
head(res$data)
res$data[[2]]$doi

## End(Not run)
```

core_cache *Caching*

Description

Manage cached rcoreoa files with **hoardr**

Details

The dafault cache directory is `paste0(rappdirs::user_cache_dir(), "/R/rcoreoa")`, but you can set your own path using `cache_path_set()`

`cache_delete` only accepts 1 file name, while `cache_delete_all` doesn't accept any names, but deletes all files. For deleting many specific files, use `cache_delete` in a `lapply()` type call

Useful user functions

- `core_cache$cache_path_get()` get cache path
- `core_cache$cache_path_set()` set cache path
- `core_cache$list()` returns a character vector of full path file names
- `core_cache$files()` returns file objects with metadata
- `core_cache$details()` returns files with details
- `core_cache$delete()` delete specific files
- `core_cache$delete_all()` delete all files, returns nothing

Examples

```
## Not run:
core_cache

# list files in cache
core_cache$list()

# delete certain database files
# core_cache$delete("file path")
# core_cache$list()

# delete all files in cache
# core_cache$delete_all()
# core_cache$list()

# set a different cache path from the default

## End(Not run)
```

core_journals *Get journal via its ISSN*

Description

Get journal via its ISSN

Usage

```
core_journals(id, key = NULL, method = "GET", parse = TRUE, ...)

core_journals_(id, key = NULL, method = "GET", ...)

core_repos_(id, key = NULL, method = "GET", ...)
```

Arguments

id	(integer) One or more journal ISSNs. Required
key	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name CORE_KEY or an R option as core_key. See ?Startup for how to work with env vars and R options
method	(character) one of 'GET' (default) or 'POST'
parse	(logical) Whether to parse to list (FALSE) or data.frame (TRUE). Default: TRUE
...	Curl options passed to HttpClient

Details

core_journals does the HTTP request and parses, while core_journals_ just does the HTTP request, gives back JSON as a character string

These functions take one article ID at a time. Use lapply/loops/etc for many ids

References

https://core.ac.uk/docs/#!/journals/getJournalByIssnBatch https://core.ac.uk/docs/#!/journals/getJournalByIssn

Examples

```
## Not run:
core_journals(id = '2167-8359')

ids <- c("2167-8359", "2050-084X")
res <- lapply(ids, core_journals)
vapply(res, "[[", "", c("data", "title"))
```

```
# just http request, get text back
core_journals_('2167-8359')

# post request, ideal for lots of ISSNs
if (requireNamespace("rcrossref", quietly = TRUE)) {
 library(rcrossref)
 res <- lapply(c("bmc", "peerj", "elife", "plos", "frontiers"), function(z)
    cr_journals(query = z))
 ids <- na.omit(unlist(lapply(res, function(b) b$data$issn)))
 out <- core_journals(ids, method = "POST")
 head(out)
}


## End(Not run)
```

`core_repos` *Get repositories via their repository IDs*

Description

Get repositories via their repository IDs

Usage

```
core_repos(id, key = NULL, method = "GET", parse = TRUE, ...)
```

Arguments

`id`	(integer) One or more repository IDs. Required
`key`	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name `CORE_KEY` or an R option as `core_key`. See `?Startup` for how to work with env vars and R options
`method`	(character) one of 'GET' (default) or 'POST'
`parse`	(logical) Whether to parse to list (`FALSE`) or data.frame (`TRUE`). Default: `TRUE`
`...`	Curl options passed to `HttpClient`

Details

`core_repos` does the HTTP request and parses, while `core_repos_` just does the HTTP request, gives back JSON as a character string

These functions take one article ID at a time. Use lapply/loops/etc for many ids

References

https://core.ac.uk/docs/#!/repositories/getRepositoryById https://core.ac.uk/docs/#!/repositories/getRepositoryByIdBatch

Examples

```
## Not run:
core_repos(id = 507)
core_repos(id = 444)

ids <- c(507, 444, 70)
res <- lapply(ids, core_repos)
vapply(res, "[[", "", c("data", "name"))

# just http request, get json as character vector back
core_repos_(507)

## End(Not run)
```

---

core_repos_search *Search CORE repositories*

Description

Search CORE repositories

Usage

```
core_repos_search(query, page = 1, limit = 10, key = NULL, parse = TRUE, ...)
```

Arguments

query	(character) query string, required
page	(character) page number (default: 1), optional
limit	(character) records to return (default: 10, minimum: 10, maximum: 100), optional
key	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name CORE_KEY or an R option as core_key. See ?Startup for how to work with env vars and R options
parse	(logical) Whether to parse to list (FALSE) or data.frame (TRUE). Default: TRUE
...	Curl options passed to HttpClient

Details

core_repos_search does the HTTP request and parses, while core_repos_search_ just does the HTTP request, gives back JSON as a character string

A POST method is allowed on this route, but it's not supported yet.

References

https://core.ac.uk/docs/#!/repositories/search

Examples

```
## Not run:
core_repos_search(query = 'mathematics')
core_repos_search(query = 'physics', parse = FALSE)
core_repos_search(query = 'pubmed')

core_repos_search_(query = 'pubmed')
library("jsonlite")
jsonlite::fromJSON(core_repos_search_(query = 'pubmed'))

## End(Not run)
```

---

core_repos_search_ *Search CORE*

Description

Search CORE

Usage

```
core_repos_search_(query, page = 1, limit = 10, key = NULL, ...)

core_search(query, page = 1, limit = 10, key = NULL, parse = TRUE, ...)

core_search_(query, page = 1, limit = 10, key = NULL, ...)
```

Arguments

query	(character) query string, required
page	(character) page number (default: 1), optional
limit	(character) records to return (default: 10, minimum: 10, maximum: 100), optional
key	A CORE API key. Get one at https://core.ac.uk/api-keys/register. Once you have the key, you can pass it into this parameter, or as a much better option, store your key as an environment variable with the name CORE_KEY or an R option as core_key. See ?Startup for how to work with env vars and R options

...	Curl options passed to HttpClient
parse	(logical) Whether to parse to list (FALSE) or data.frame (TRUE). Default: TRUE

Details

core_search does the HTTP request and parses, while core_search_ just does the HTTP request, gives back JSON as a character string

References

https://core.ac.uk/docs/#!/all/search

Examples

```
## Not run:
core_search(query = 'ecology')
core_search(query = 'ecology', parse = FALSE)
core_search(query = 'ecology', limit = 12)

core_search_(query = 'ecology')
library("jsonlite")
jsonlite::fromJSON(core_search_(query = 'ecology'))

# post request
query <- c('data mining', 'machine learning', 'semantic web')
res <- core_search(query)
res
res$status
res$totalHits
res$data
head(res$data[[1]])

## End(Not run)
```

Index