Package 'rcoreoa'

June 20, 2017

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 0.1.0

License MIT + file LICENSE

URL https://github.com/ropensci/rcoreoa

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

VignetteBuilder knitr

Imports crul (>= 0.3.8), jsonlite (>= 1.5), pdftools (>= 1.3)

Suggests roxygen2 (>= 6.0.1), testthat, knitr, rmarkdown, rcrossref

RoxygenNote 6.0.1

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2017-06-20 19:01:56 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()` / `core_advanced_search_()` - advanced search of articles

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Aristotelis Charalampous

core_advanced_search *Advanced Search CORE*

Description

Advanced Search CORE

Usage

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

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

Arguments

query	data.frame, required (details for structure)
page	(character) page number (default: 1), optional
limit	(character) records to return (default: 10, minimum: 10), optional
key	A IUCN API token
parse	(logical) Whether to parse to list (‘FALSE‘) or data.frame (‘TRUE‘). Default: ‘TRUE‘
...	Curl options passed to [crul::HttpClient()]

Details

query should include columns with the following information (at least one is required):

1. all_of_the_words: string, with space separated terms that should all exist in target document(s)
2. exact_phrase: string, used as an absolute match in comparison with all_of_the_words
3. at_least_one_of_the_words: string, with space separated terms of which at least one should exist in target document(s)
4. without_the_words: string, with space separated terms of which none should exist in target document(s)
5. find_those_words: 3 available options, a. "anywhere in the article" (default), b. "in the title", c. "in the title and abstract" to either do a fulltext search, a title only or a title and abstract respectively
6. author: string, to be used as an absolute match against the author name metadata field
7. publisher: string, to be used as an absolute match against the publisher name metadata field
8. repository: string, to be used as an absolute match against the repository name metadata field
9. doi: string, to be used as an absolute match against the repository name metadata field (all other fields will be ignored if included)

10. year_from: string, to filter target document(s) publisher earlier than indicated
11. year_to: string, to filter target document(s) publisher later than indicated 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 resources

Examples

```
## Not run:
query <- data.frame(
"all_of_the_words" = c("data mining", "machine learning"),
"without_the_words" = c("social science", "medicine"),
"year_from" = c("2013","2000"),
"year_to" = c("2014", "2016"))

res <- core_advanced_search(query)
head(res$data)
res$data[[1]]$id

## 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 boolean
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 IUCN API token
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 [crul::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, similar = TRUE)
core_articles(id = 21132995, fulltext = TRUE)
core_articles(id = 21132995, citations = TRUE)

ids <- c(20955435, 21132995, 21813171, 22815670, 14045109, 23828884,
   23465055, 23831838, 23923390, 22559733)
res <- lapply(ids, core_articles)
vapply(res, "[[", "", c("data", "datePublished"))
```

```
# post request
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_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. Required
`page`	(character) page number (default: 1), optional
`limit`	(character) records to return (default: 10, minimum: 10), optional
`key`	A IUCN API token
`parse`	(logical) Whether to parse to list (‘FALSE‘) or data.frame (‘TRUE‘). Default: ‘TRUE‘
`...`	Curl options passed to [crul::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

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

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 <- lapply(ids, core_articles_history)
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 = TRUE, parse = TRUE, ...)

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

Arguments

id	(integer) CORE ID of the article that needs to be fetched. Required
key	A IUCN API token
overwrite	(logical) overwrite file or not. Default: TRUE
parse	(logical) Whether to parse to list ('FALSE') or data.frame ('TRUE'). Default: 'TRUE'
...	Curl options passed to [crul::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

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

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 <- lapply(ids, core_articles_pdf)
vapply(res, "[[", "", 1)

## 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 IUCN API token
`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 [crul::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", "1471-2105", "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 <- 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 IUCN API token
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 [crul::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), optional
key	A IUCN API token
parse	(logical) Whether to parse to list ('FALSE') or data.frame ('TRUE'). Default: 'TRUE'
...	Curl options passed to [crul::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), optional
key	A IUCN API token
...	Curl options passed to [crul::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'))

query <- c('data mining', 'machine learning', 'semantic web')

# post request
res <- core_search(query)
head(res$data)
res$data$id

## End(Not run)
```

Index