Package: scholar (via r-universe)

October 29, 2026

Type Package

Title Analyse Citation Data from Google Scholar

Version 1.0.1

Maintainer Guangchuang Yu <guangchuangyu@gmail.com>

Description Provides tools to retrieve, analyse, and format publication and citation data from Google Scholar profiles. It supports profile lookup, publication records, citation histories, scholar comparison, coauthor networks, journal ranking queries, CV-ready publication lists, and citation metric summaries.

Depends R (>= 4.1.0)

Imports R.cache, dplyr, httr, rlang, rvest, stats, stringr, xml2, tidygraph, ggraph, ggplot2

Suggests knitr, quarto, rmarkdown, roxygen2, spelling, testthat (>= 2.1.0), withr, yulab.utils

VignetteBuilder quarto

License MIT + file LICENSE

URL https://github.com/YuLab-SMU/scholar

BugReports https://github.com/YuLab-SMU/scholar/issues

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Guangchuang Yu [aut, cre] (ORCID: <https://orcid.org/0000-0002-6485-8781>), James Keirstead [aut], Gregory Jefferis [aut] (ORCID: <https://orcid.org/0000-0002-0587-9355>), Gordon Getzinger [ctb], Jorge Cimentada [ctb], Max Czapanskiy [ctb], Dominique Makowski [ctb]

Repository https://cran.r-universe.dev

Date/Publication 2026-10-29 08:20:07 UTC

RemoteUrl https://github.com/cran/scholar

RemoteRef HEAD

RemoteSha 675c04616e175048195145d882e816d9cebda0da

Contents

author_position . . . 3
compare_scholar_careers . . . 4
compare_scholars . . . 4
format_authors . . . 5
format_publications . . . 5
get_article_cite_history . . . 6
get_article_scholar_url . . . 7
get_citation_history . . . 7
get_citing_articles . . . 8
get_coauthors . . . 9
get_complete_authors . . . 10
get_h_index_history . . . 11
get_journalrank . . . 12
get_num_articles . . . 13
get_num_distinct_journals . . . 13
get_num_top_journals . . . 14
get_oldest_article . . . 14
get_profile . . . 15
get_publication_abstract . . . 16
get_publication_data_extended . . . 16
get_publication_date . . . 17
get_publication_metrics . . . 17
get_publication_url . . . 18
get_publications . . . 18
get_publications_all_authors . . . 19
get_scholar_id . . . 20
get_scholar_metrics . . . 21
get_scholar_resp . . . 21
plot_coauthors . . . 22
predict_h_index . . . 22
search_openalex_authors . . . 23
search_scholar_ids . . . 24
set_scholar_cookie . . . 25
set_scholar_mirror . . . 26

Index **27**

`author_position` *Get author order.*

Description

Get author rank in authors list.

Usage

```
author_position(authorlist, author)
```

Arguments

`authorlist`	list of publication authors
`author`	author's name to look for

Value

dataframe with author's position and normalized position (a normalized index, with 0 corresponding, 1 to last and 0.5 to the middle. Note that single authorship will be considered as last, i.e., 1).

Author(s)

Dominique Makowski

Examples

```
## Not run:
library(scholar)

id <- "DO5oG40AAAAJ"

authorlist <- scholar::get_publications(id)$author
author <- scholar::get_profile(id)$name

author_position(authorlist, author)

## End(Not run)
```

compare_scholar_careers *Compare the careers of multiple scholars*

Description

Compares the careers of multiple scholars based on their citation histories. The scholar's *career* is defined by the number of citations to his or her work in a given year (i.e. the bar chart at the top of a scholar's profile). The function has an `career` option that allows users to compare scholars directly, i.e. relative to the first year in which their publications are cited.

Usage

```
compare_scholar_careers(ids, career = TRUE)
```

Arguments

`ids`	a character vector of Google Scholar IDs
`career`	a boolean, should a column be added to the results measuring the year relative to the first citation year. Default = TRUE

Examples

```
## How do Richard Feynmann and Stephen Hawking compare?
# Compare Feynman and Stephen Hawking
ids <- c("B7vSqZsAAAAJ", "DO5oG40AAAAJ")
df <- compare_scholar_careers(ids)
```

compare_scholars *Compare the citation records of multiple scholars*

Description

Compares the citation records of multiple scholars. This function compiles a data frame comparing the citations received by each of the scholar's publications by year of publication.

Usage

```
compare_scholars(ids, pagesize = 100)
```

Arguments

`ids`	a vector of Google Scholar IDs
`pagesize`	an integer specifying the number of articles to fetch for each scholar

Value

a data frame giving the ID of each scholar and the total number of citations received by work published in a year.

Examples

```
## Not run:
    ## How do Richard Feynmann and Stephen Hawking compare?
    ids <- c("B7vSqZsAAAAJ", "DO5oG40AAAAJ")
    df <- compare_scholars(ids)

## End(Not run)
```

`format_authors` *format_authors*

Description

This function converts first and middle names to initials

Usage

```
format_authors(string)
```

Arguments

`string`	a character vector of names

`format_publications` *format_publications*

Description

Format publication list

Usage

```
format_publications(scholar.profile, author.name = NULL)
```

Arguments

`scholar.profile`	scholar profile ID
`author.name`	name of author to be highlighted using bold font

Value

a vector of formatted publications

Author(s)

R Thériault and modified by Guangchuang Yu

Examples

```
## Not run:
 library(scholar)
 format_publications("DO5oG40AAAAJ")

## End(Not run)
```

`get_article_cite_history` *Gets the citation history of a single article*

Description

Gets the citation history of a single article

Usage

```
get_article_cite_history(id, article)
```

Arguments

`id`	a character string giving the id of the scholar
`article`	a character string giving the article id.

Value

a data frame giving the year, citations per year, and publication id

get_article_scholar_url *Gets the URL to the google scholar website of an article.*

Description

Gets the URL to the google scholar website of an article.

Usage

```
get_article_scholar_url(id, pubid)
```

Arguments

id	a character string specifying the Google Scholar ID.
pubid	a character string specifying the article id.

Value

a String that contains the URL to the scholar website of the article

get_citation_history *Get historical citation data for a scholar*

Description

Gets the number of citations to a scholar's articles over the past nine years.

Usage

```
get_citation_history(id)
```

Arguments

id	a character string specifying the Google Scholar ID. If multiple ids are specified, only the first value is used and a warning is generated.

Details

This information is displayed as a bar plot at the top of a standard Google Scholar page and only covers the past nine years.

Value

a data frame giving the number of citations per year to work by the given scholar

`get_citing_articles` *List the publications that cite a publication*

Description

Follows the "cited by" link of one publication and returns the works Google Scholar lists as citing it, ten at a time. The citation counts on a profile say how often a paper was cited; this says by whom, and through which venue, so that a citation curve can be split by community, an autocitation rate measured, or the speed of adoption plotted.

Usage

```
get_citing_articles(id, pubid, max_pages = 5, delay = 0, publications = NULL)
```

Arguments

`id`	a character string specifying the Google Scholar ID of the author whose publication is to be expanded.
`pubid`	a character vector of publication IDs, as found in the `pubid` column of `get_publications`.
`max_pages`	the number of ten-entry pages to read per publication.
`delay`	seconds to wait between pages, for the benefit of the shared rate limit.
`publications`	a table as returned by `get_publications`, used to resolve `pubid` to a citation key. Supply it when expanding several publications, so that the list is fetched once rather than per pubid.

Details

number on a profile page, and it is bounded the same way: Scholar exposes at most the first thousand citing works in ten-entry pages, and folds the versions of one work into a single entry. Google guards this list more strictly than it guards a profile page, so a hosted or data-centre IP is frequently answered with a captcha instead of data; see `scholar-package` and `set_scholar_cookie`. A refusal is reported as a warning and returns an empty table, never a silently short list.

Value

a data frame with one row per citing publication and the columns `pubid`, the publication it cites; `title`; `author`; `venue`; `year`; `cites`, citations to the citing work itself; `url`, the publisher page where Scholar links to one; and `cid`, Scholar's citation key for the citing work, which can be handed to this function's internals to follow the list one step further. Rows keep the order Scholar uses, which is by relevance. The table is empty when nothing cites the publication or when Scholar refused.

See Also

`get_publications` for the `pubid` column, and `get_article_cite_history` for citations to one publication per year.

Examples

```
pubs <- get_publications("B7vSqZsAAAAJ")
# Scholar can refuse the list outright, in which case there is nothing to
# expand: get_publications() answers with NA rather than an empty table
if (is.data.frame(pubs) && nrow(pubs)) {
  get_citing_articles("B7vSqZsAAAAJ", pubs$pubid[1], publications = pubs,
                      max_pages = 1)
}
```

---

get_coauthors *Gets the network of coauthors of a scholar*

Description

Gets the network of coauthors of a scholar

Usage

```
get_coauthors(id, n_coauthors = 5, n_deep = 1)
```

Arguments

id	a character string specifying the Google Scholar ID. If multiple ids are specified, only the first value is used and a warning is generated.
n_coauthors	Number of coauthors to explore. This number should usually be between 1 and 10 as choosing many coauthors can make the network graph too messy.
n_deep	The number of degrees that you want to go down the network. When n_deep is equal to 1 then grab_coauthor will only grab the coauthors of Joe and Mary, so Joe – > Mary –> All coauthors. This can get out of control very quickly if n_deep is set to 2 or above. The preferred number is 1, the default.

Details

Considering that scraping each publication for all coauthors is error prone, get_coauthors grabs the coauthors listed by Google Scholar for the profile, not from all publications. It first tries Google Scholar's full colleagues list and falls back to the profile sidebar if that list is unavailable.

Value

A data frame with two columns showing all authors and coauthors.

See Also

plot_coauthors

Examples

```
## Not run:

library(scholar)
coauthor_network <- get_coauthors('amYIKXQAAAAJ')
plot_coauthors(coauthor_network)

## End(Not run)
```

`get_complete_authors` *Get the Complete list of authors for a Publication*

Description

Found as Muhammad Qasim Pasta's solution here https://github.com/jkeirstead/scholar/issues/21

Usage

```
get_complete_authors(id, pubid, delay = 0.4, initials = TRUE)
```

Arguments

`id`	a Google Scholar ID
`pubid`	a Publication ID from a given google Scholar ID
`delay`	average delay between requests. A delay is needed to stop Google identifying you as a bot
`initials`	if TRUE (default), first and middle names will be abbreviated

Value

a string containing the complete list of authors

Author(s)

Muhammad Qasim Pasta

Abram B. Fleishman

James H. Conigrave

`get_h_index_history` *Track how an h-index developed over the years*

Description

A profile reports one h-index, at one moment. This rebuilds the number year by year, from the citation history of each publication, so that the shape of a career is visible: whether the h-index was set early by a few papers, is still climbing, or plateaued when an author stopped publishing.

Usage

```
get_h_index_history(id, publications = NULL, delay = 0, flush = FALSE)
```

Arguments

`id`	a character string specifying the Google Scholar ID.
`publications`	a table, as returned by `get_publications`, to take the publication IDs from instead of fetching them again.
`delay`	seconds to wait between the per-publication requests. Worth setting on a shared or data-centre IP, where a loop over a long publication list is exactly what gets a connection rate limited.
`flush`	whether to discard the cached citation histories and fetch them again.

Details

apiece. That is the expensive part, so the histories are cached with `R.cache` the way `get_publication_date` caches its pages, and a second call for the same profile costs nothing. Publications with no citations are skipped: they cannot enter an h-index in any year. The years covered are the years Google reports per publication, so a publication whose history Google truncates contributes only the years it publishes.

The h-index in a given year is counted from the citations accumulated up to and including that year, which is what the number means on a profile today. The result is a plain data frame, so plotting the trajectory is one ggplot2 call away (`geom_step(direction = "hv")` suits a step function); the package vignette carries the worked example.

Value

a data frame with one row per year, oldest first: `year`, `h_index` counted up to that year, and `citations`, the citations accumulated across all publications by that year. The h-index never falls as the years advance, since accumulated citations do not decrease.

See Also

`get_profile` for the h-index Google reports today, `get_article_cite_history` for one publication's history, and `get_publication_metrics` for h-index, g-index and i10-index counted over a set of publications.

Examples

```
pubs <- get_publications("B7vSqZsAAAAJ")
# Scholar may refuse the list outright, answering with NA
if (is.data.frame(pubs) && nrow(pubs)) {
  # one request per publication, so start from the few that matter
  few <- head(pubs[pubs$cites > 0, c("pubid", "cites")], 3)
  head(get_h_index_history("B7vSqZsAAAAJ", publications = few))
}
```

`get_journalrank` *Get journal ranking (deprecated)*

Description

Deprecated: 'get_journalrank()' no longer downloads anything. SCImago has served its ranking page behind a JavaScript challenge since late 2025, so a script is answered with HTTP 403 and cannot read the table at all, and the table itself carries no published licence - it derives from Elsevier's Scopus. Calling the function now says where journal metrics can honestly be obtained instead of failing after a download nobody can make.

Usage

```
get_journalrank(journals, max.distance = 0.05)
```

Arguments

`journals`	a character vector of journal names. Unused; kept so that existing calls reach this message rather than an unrelated error.
`max.distance`	unused, kept for the same reason.

Details

For journal metrics, OpenAlex is openly licensed, made for programmatic use and needs no scraping: https://api.openalex.org/sources?search=Nature returns the h-index, i10-index, work and citation counts, and the ISSNs of a journal. If you need SCImago's SJR quartiles specifically, the table has to be saved by hand from https://www.scimagojr.com/journalrank.php in a browser, and can then be read with `utils::read.csv2()`.

Value

Never returns: an error is raised with the alternatives.

Author(s)

Dominique Makowski and Guangchuang Yu

Examples

```
## Not run:
get_journalrank(c("Nature", "Science"))

## End(Not run)
```

`get_num_articles` *Calculates how many articles a scholar has published*

Description

Calculate how many articles a scholar has published.

Usage

```
get_num_articles(id)
```

Arguments

`id`	a character string giving the Google Scholar ID

Value

an integer value (max 100)

`get_num_distinct_journals`
Gets the number of distinct journals in which a scholar has published

---

Description

Gets the number of distinct journals in which a scholar has published. Note that Google Scholar doesn't provide information on journals *per se*, but instead gives a title for the containing publication where applicable. So a *journal* here might actually be a journal, a book, a report, or some other publication outlet.

Usage

```
get_num_distinct_journals(id)
```

Arguments

`id`	a character string giving the Google Scholar id

Value

the number of distinct journals

get_num_top_journals *Gets the number of top journals in which a scholar has published*

Description

Gets the number of top journals in which a scholar has published. The definition of a 'top journal' comes from Acuna et al. and the original list was based on the field of neuroscience. This function allows users to specify that list for themselves, or use the default Acuna et al. list.

Usage

```
get_num_top_journals(id, journals)
```

Arguments

id	a character string giving a Google Scholar ID
journals	a character vector giving the names of the top journals. Defaults to Nature, Science, Nature Neuroscience, Proceedings of the National Academy of Sciences, and Neuron.

Source

DE Acuna, S Allesina, KP Kording (2012) Future impact: Predicting scientific success. Nature 489, 201-202. doi:10.1038/489201a.

get_oldest_article *Gets the year of the oldest article for a scholar*

Description

Gets the year of the oldest article published by a given scholar.

Usage

```
get_oldest_article(id)
```

Arguments

id	a character string giving the Google Scholar ID

Value

the year of the oldest article

`get_profile` *Gets profile information for a scholar*

Description

Gets profile information for a researcher from Google Scholar. Each scholar profile page gives the researcher's name, affiliation, their homepage (if specified), and a summary of their key citation and publication availability metrics. The scholar ID can be found by searching Google Scholar at http://scholar.google.com.

Usage

```
get_profile(id)
```

Arguments

`id`	a character string specifying the Google Scholar ID. If multiple ids are specified, only the first value is used and a warning is generated. See the example below for how to profile multiple scholars.

Value

a list containing the scholar's name, affiliation, citations, impact and publication availability metrics, research interests, homepage and coauthors.

Metrics include:

- total_cites combined citations to all publications
- h_index the largest number h such that h publications each have at least h citations
- i10_index the number of publications that each have at least 10 citations
- available the number of publications that have a version online that can be read for free (though not necessarily reusable under an open access license)
- not_available the number of publications only available behind a paywall

Examples

```
## Gets profiles of some famous physicists
ids <- c("xJaxiEEAAAAJ", "DO5oG40AAAAJ")
profiles <- lapply(ids, get_profile)
```

`get_publication_abstract` *Gets the abstract for a publication id.*

Description

Gets the abstract for a publication id.

Usage

```
get_publication_abstract(id, pub_id, flush = FALSE)
```

Arguments

`id`	a character string specifying the Google Scholar ID.
`pub_id`	a character string specifying the publication id.
`flush`	Whether or not to clear the cache

Value

a String that contains the abstract of the publication.

`get_publication_data_extended` *Gets the full data for a publication*

Description

Gets the full data for a publication

Usage

```
get_publication_data_extended(id, pub_id, flush = FALSE)
```

Arguments

`id`	a character string specifying the Google Scholar ID.
`pub_id`	a character string specifying the publication id.
`flush`	Whether or not to clear the cache

Value

a list that contains the full data

get_publication_date *Gets the full date for a publication*

Description

Gets the full date for a publication

Usage

```
get_publication_date(id, pub_id, flush = FALSE)
```

Arguments

id	a character string specifying the Google Scholar ID.
pub_id	a character string specifying the publication id.
flush	Whether or not to clear the cache

Value

a String that contains the publication date

get_publication_metrics *Calculate citation metrics from publication data*

Description

Calculates common citation metrics from a publication data frame, such as the result returned by get_publications.

Usage

```
get_publication_metrics(publications)
```

Arguments

publications	a data frame containing a numeric cites column

Value

a data frame with h-index, g-index, i10-index, i50-index, i100-index, total citations, and publication count

get_publication_url *Gets the PDF URL for a publication id.*

Description

Gets the PDF URL for a publication id.

Usage

```
get_publication_url(id, pub_id, flush = FALSE)
```

Arguments

id	a character string specifying the Google Scholar ID.
pub_id	a character string specifying the publication id.
flush	Whether or not to clear the cache

Value

a String that contains the URL to the PDF of the publication.

get_publications *Gets the publications for a scholar*

Description

Gets the publications of a specified scholar.

Usage

```
get_publications(
  id,
  cstart = 0,
  cstop = Inf,
  pagesize = 100,
  flush = FALSE,
  sortby = "citation"
)
```

Arguments

id	a character string specifying the Google Scholar ID. If multiple IDs are specified, only the publications of the first scholar will be retrieved.
cstart	an integer specifying the first article to start counting. To get all publications for an author, omit this parameter.
cstop	an integer specifying the last article to process.
pagesize	an integer specifying the number of articles to fetch in one batch. It is recommended to leave the default value of 100 unless you experience time-out errors. Note this is *not* the **total** number of publications to fetch.
flush	should the cache be flushed? Search results are cached by default to speed up repeated queries. If this argument is TRUE, the cache will be cleared and the data reloaded from Google.
sortby	a character with value "citation" or value "year" specifying how results are sorted.

Details

Google uses two id codes to uniquely reference a publication. The results of this method includes cid which can be used to link to a publication's full citation history (i.e. if you click on the number of citations in the main scholar profile page), and pubid which links to the details of the publication (i.e. if you click on the title of the publication in the main scholar profile page.)

Value

a data frame listing the publications and their details. These include the publication title, author, journal, number, cites, year, and two id codes (see details).

---

get_publications_all_authors

Gets publications with complete author lists

Description

Gets a scholar's publications and fills truncated author lists by calling get_complete_authors only for publications whose author field contains ....

Usage

```
get_publications_all_authors(id, delay = 0.8, initials = TRUE, ...)
```

Arguments

id	a character string specifying the Google Scholar ID.
delay	average delay between requests for complete author lists. See get_complete_authors.
initials	if TRUE, first and middle names in completed author lists will be abbreviated. See get_complete_authors.
...	additional arguments passed to get_publications.

Value

a data frame listing the publications and their details, with truncated author lists replaced where possible.

`get_scholar_id` *Search for Google Scholar ID by name and affiliation*

Description

Deprecated for anonymous use: Google now answers author search with a sign-in page, so this works only where a browser session has been supplied with `set_scholar_cookie`, or through a mirror that serves the endpoint. `search_openalex_authors` does the same job with no session at all and is the route to prefer for new code.

Usage

```
get_scholar_id(last_name = "", first_name = "", affiliation = NA)
```

Arguments

`last_name`	Researcher last name.
`first_name`	Researcher first name.
`affiliation`	Researcher affiliation.

Value

Google Scholar ID as a character string.

Examples

```
get_scholar_id(first_name = "kristopher", last_name = "mcneill")
get_scholar_id(first_name = "michael", last_name = "sander", affiliation = NA)
get_scholar_id(first_name = "michael", last_name = "sander", affiliation = "eth")
get_scholar_id(first_name = "michael", last_name = "sander", affiliation = "ETH Zurich")
get_scholar_id(first_name = "michael", last_name = "sander", affiliation = "Mines")
get_scholar_id(first_name = "james", last_name = "babler")
```

get_scholar_metrics *Calculate citation metrics for a scholar*

Description

Fetches a scholar's publications with `get_publications` and calculates common citation metrics.

Usage

```
get_scholar_metrics(id, pagesize = 100, flush = FALSE)
```

Arguments

`id`	a character string specifying the Google Scholar ID
`pagesize`	an integer specifying the number of articles to fetch in one batch. See `get_publications`.
`flush`	should the publication cache be flushed?

Value

a data frame with h-index, g-index, i10-index, i50-index, i100-index, total citations, and publication count

get_scholar_resp *Recursively try to GET a Google Scholar Page storing session cookies*

Description

see `scholar-package` documentation for details about Scholar session cookies.

Usage

```
get_scholar_resp(url, attempts_left = 5)
```

Arguments

`url`	URL to fetch
`attempts_left`	The number of times to try and fetch the page

Value

an [response][httr::response] object

See Also

[GET][httr::GET]

plot_coauthors *Plot a network of coauthors*

Description

Plot a network of coauthors

Usage

```
plot_coauthors(network, size_labels = 5)
```

Arguments

network	A data frame given by get_coauthors
size_labels	Size of the label names

Value

a ggplot2 object but prints a plot as a side effect.

See Also

get_coauthors

Examples

```
## Not run:
library(scholar)
coauthor_network <- get_coauthors('amYIKXQAAAAJ&hl')
plot_coauthors(coauthor_network)

## End(Not run)
```

predict_h_index *Predicts the h-index for a researcher*

Description

Predicts the h-index for a researcher each year for ten years into the future using Acuna et al's method (see source). The model was fit to data from neuroscience researchers with an h-index greater than 5 and between 5 to 12 years since publishing their first article. So naturally if this isn't you, then the results should be taken with a large pinch of salt.

Usage

```
predict_h_index(id, journals)
```

Arguments

id	a character string giving the Google Scholar ID
journals	optional character vector of top journals. See get_num_top_journals for more details.

Details

Since the model is calibrated to neuroscience researchers, it is entirely possible that very strange (e.g. negative) h-indices will be predicted if you are a researcher in another field. A warning will be displayed if the sequence of predicted h-indices contains a negative value or is non-increasing.

Value

a data frame giving predicted h-index values in future

Note

A scientist has an h-index of n if he or she publishes n papers with at least n citations each. Values returned are fractional so it's up to your own vanity whether you want to round up or down.

Source

DE Acuna, S Allesina, KP Kording (2012) Future impact: Predicting scientific success. Nature 489, 201-202. doi:10.1038/489201a. Thanks to DE Acuna for providing the full regression coefficients for each year ahead prediction.

Examples

```
  ## Predict h-index of original method author
## Not run:
  id <- "DO5oG40AAAAJ"
  df <- predict_h_index(id)

## End(Not run)
```

---

search_openalex_authors

Look up authors and their citation metrics in OpenAlex

Description

Google Scholar put author search behind a Google login, so get_scholar_id and search_scholar_ids can only reach it with a session borrowed from a browser. OpenAlex is the alternative that needs no session and no scraping: it is an openly licensed index of scholarly records, maintained for programmatic use, and it publishes the h-index and i10-index alongside the works it counted.

Usage

```
search_openalex_authors(name, max_results = 10)
```

Arguments

name	a character string to search for, such as "Guangchuang Yu".
max_results	the number of author records to return, at most 50. Results come back in OpenAlex's relevance order, so the person you want is usually first when the name is not common.

Details

The numbers will not match Google Scholar's. OpenAlex counts citations in its own corpus, so an h-index here is usually lower than the one a Scholar profile shows, and the two should not be mixed in one table.

Value

a data frame with one row per candidate author and the columns openalex_id, orcid, display_name, affiliation, works_count, cited_by_count, h_index, i10_index and url.

The polite pool

OpenAlex asks callers who can be identified to pass an email address, which routes the request to a separately managed pool. Set options(scholar_openalex_mailto = "you@example.org") or the SCHOLAR_OPENALEX_MAILTO environment variable to do so; without it the lookup still works.

See Also

get_scholar_id for the Google Scholar route, and get_scholar_metrics for the same metrics as Scholar reports them.

Examples

```
search_openalex_authors("Guangchuang Yu")
search_openalex_authors("Richard Feynman", max_results = 3)
```

---

search_scholar_ids *Search Google Scholar IDs by author name*

Description

Searches Google Scholar author results and returns matching author IDs. Author search is deprecated for anonymous use: Google redirects it to a sign-in page unless the request carries a browser session, see set_scholar_cookie. search_openalex_authors returns the same kind of candidate list without any session.

Usage

```
search_scholar_ids(name, max_pages = Inf, delay = 0)
```

Arguments

name	author search query
max_pages	maximum number of search result pages to fetch
delay	number of seconds to wait between result pages

Value

a data frame with columns id, name, affiliation, email, interests, and url

Examples

```
search_scholar_ids("hao xu", max_pages = 1)
```

---

set_scholar_cookie *set_scholar_cookie*

Description

set the Scholar session

Usage

```
set_scholar_cookie(cookie = NULL, file = NULL)
```

Arguments

cookie	a Cookie header value copied from a browser, such as "NID=...; GSP=...", or NULL to leave it unchanged
file	path to a cookie file exported from a browser, either in the Netscape format used by curl or as plain name=value text. Both a cookie and a file may be set; their entries are merged.

Details

Supply the session that lets Google Scholar treat this R session like the browser you are already signed in with. Some endpoints, author search among them, answer an anonymous request with a Google sign-in page while showing data in a browser; the difference is the cookie.

Google's own cookies are HttpOnly, so a browser will not reveal them to a script or a copy button: take them from the developer tools under Application, or with an extension that exports a cookie file. Published lifetimes are six months for NID and thirteen for _Secure-ENID and SOCS, counted from last use, and Google ends a session sooner when it looks automated, so expect to export it

again. On a server or in CI, where there is no browser to copy from, set the SCHOLAR_COOKIE_FILE environment variable rather than passing the value here: a session is a credential, and an option or an argument lands in the command history. The session is sent only to a Google host, so a mirror set with set_scholar_mirror never receives it.

Value

NULL, invisibly; the values are stored as options

Author(s)

Guangchuang Yu

Examples

```
## Not run:
  set_scholar_cookie(cookie = "NID=abc123; GSP=XQwERT")
  set_scholar_cookie(file = "~/scholar-cookies.txt")

## End(Not run)
```

---

set_scholar_mirror *set_scholar_mirror*

Description

set scholar mirror

Usage

```
set_scholar_mirror(mirror = NULL)
```

Arguments

mirror	compatible scholar mirror

Details

setting google scholar mirror

Author(s)

Guangchuang Yu

Index

author_position, 3

compare_scholar_careers, 4
compare_scholars, 4

format_authors, 5
format_publications, 5

get_article_cite_history, 6, *8*, *11*
get_article_scholar_url, 7
get_citation_history, 7
get_citing_articles, 8
get_coauthors, 9, *22*
get_complete_authors, 10, *19*
get_h_index_history, 11
get_journalrank, 12
get_num_articles, 13
get_num_distinct_journals, 13
get_num_top_journals, 14, *23*
get_oldest_article, 14
get_profile, *11*, 15
get_publication_abstract, 16
get_publication_data_extended, 16
get_publication_date, *11*, 17
get_publication_metrics, *11*, 17
get_publication_url, 18
get_publications, *8*, *11*, *17*, 18, *19*, *21*
get_publications_all_authors, 19
get_scholar_id, 20, *23, 24*
get_scholar_metrics, 21, *24*
get_scholar_resp, 21

plot_coauthors, *9*, 22
predict_h_index, 22

search_openalex_authors, *20*, 23, *24*
search_scholar_ids, *23*, 24
set_scholar_cookie, *8*, *20*, *24*, 25
set_scholar_mirror, 26, *26*