Package: IPBES.R (via r-universe)

October 13, 2024

Type Package

Title Tool functions used by the Data and Knowledge Technical Support Unit of IPBES

Version 0.4.2

Date 2023-11-13

Maintainer Rainer M. Krug <Rainer@krugs.de>

URL https://ipbes-data.github.io/IPBES.R/, https://github.com/ipbes-data/IPBES.R

BugReports https://github.com/ipbes-data/IPBES.R/issues

Description More about what it does (maybe more than one line).

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports openalexR, dplyr, writexl, tidygraph, ggplot2, ggraph, networkD3, httr2, tibble, tidyterra, terra, countrycode, geodata, DT

Suggests roxyglobals, tinytest

Encoding UTF-8

RoxygenNote 7.2.3

Roxygen list(roclets = c(``collate'', ``namespace'', ``rd'', ``roxyglobals::global_roclet''))

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

Repository https://ipbes-data.r-universe.dev

RemoteUrl https://github.com/IPBES-Data/IPBES.R

RemoteRef HEAD

RemoteSha ac474dcd35030b658937a6a3728d891e993e1b30

Contents

`IPBES.R-package` *Tool functions used by the Data and Knowledge Technical Support Unit of IPBES*

Description

More about what it does (maybe more than one line).

Package Content

Index: This package was not yet installed at build time.

Maintainer

Rainer M. Krug <Rainer@krugs.de>

Author(s)

Rainer M. Krug [aut, cre] (<https://orcid.org/0000-0001-8049-7069>)

`abbreviate_authors` *Abbreviate Authors*

Description

This function abbreviates the author names in a given data frame of works. The output is a list of author names in the format "First Author et al. (Year)" or "Author1 & Author2 (Year)".

Usage

```
abbreviate_authors(oa_works_df)
```

Arguments

`oa_works_df`	A data frame containing the works. It should have columns 'publication_year' and 'author'. 'author' should be a data frame with a column 'au_display_name'. One example is the data frame from 'openalexR'.

Value

A vector of abbreviated author names and publication years.

Examples

```
## Not run:
abbreviate_authors(oa_works_df)

## End(Not run)
```

`doi_exists` *Check if DOIs exist*

Description

This function checks if a given list of DOIs exist by sending HTTP requests to the DOI resolver.

Usage

```
doi_exists(dois, cache_file = NULL)
```

Arguments

`dois`	A character vector of DOIs to check.
`cache_file`	A file name of the cache to be used, i.e. the confirmed existing dois. The format is a character vector with the DOIs which exist. If the cache exist, it will be updated at the end. Temporary caches will be written after 100 checks.

Details

This function uses the httr package to send HTTP GET requests to the DOI resolver and checks the response status code. A status code of 200 indicates that the DOI exists, while any other status code indicates that the DOI does not exist.

Value

A named logical vector indicating whether each DOI does exist or not, names are the dois.

Examples

```
dois <- c("sbcd1234", "10.1234/abcd", "10.1002/jcb.23190", "10.47366/sabia.v5n1a3")
doi_exists(dois)
# Output: [1] FALSE  TRUE
```

`doi_not_retracted` *Check if DOIs are not retracted*

Description

This function checks if a given list of DOIs (Digital Object Identifiers) are retracted. It uses the Crossref API to query the Retraction Watch database.

Usage

```
doi_not_retracted(dois, cache_file = NULL, email = NULL)
```

Arguments

`dois`	A character vector of DOIs to be checked.
`cache_file`	A file name of the cache to be used, i.e. the downloaded retraction data. THe file is an 'rds' file as downloaded from the retractionwatch site. If NULL, the data will not be cached.
`email`	An optional email address to be included in the API request [RECOMMENDET!].

Value

A named logical vector indicating whether each DOI is retracted ('FALSE') or or not ('TRUE'), names are the dois.

Examples

```
# Check if a single DOI is retracted
doi_not_retracted("10.1234/abcd")

# Check if multiple DOIs are retracted
dois <- c("sbcd1234", "10.1234/abcd", "10.1002/jcb.23190", "10.47366/sabia.v5n1a3")
doi_not_retracted(dois)
```

`doi_valid` *Validate DOIs*

Description

This function validates a vector of DOIs (Digital Object Identifiers) using a regular expression pattern.#' It is taken from https://github.com/libscie/retractcheck/blob/23f1e5c7d572d9470583288d951d1bad98392f82/R/utils.I

Usage

```
doi_valid(dois)
```

Arguments

`dois`	A vector of DOIs to be validated.

Details

The function uses a regular expression pattern to validate the format of each DOI in the input vector. The regular expression pattern is based on the pattern used by the `retractcheck` package and can be found at https://github.com/libscie/retractcheck/blob/23f1e5c7d572d9470583288d951d1bad98392f82/R/utils.R#L16. Alternatively, you can uncomment the second regular expression pattern and comment out the first one to use the pattern from the `rorcid` package, which can be found at https://github.com/ropensci-archive/rorcid/blob/master/R/check_dois.R.

Value

A named logical vector indicating whether each DOI is valid or not, names are the dois.

Examples

```
dois <- c("sbcd1234", "10.1234/abcd", "10.1002/jcb.23190", "10.47366/sabia.v5n1a3")
doi_valid(dois)
```

`get_count` *Get Count*

Description

This function takes a search term and a list of DOIs, and returns the count of the search term in each DOI. If duplicate DOIs are provided, the function will stop and throw an error.

Usage

```
get_count(search_term, dois = NULL, ...)
```

Arguments

`search_term`	The term to search for.
`dois`	A list of DOIs to search within. Default is NULL.
`...`	Additional Additional filter arguments to pass to 'oa_query' as

Value

A named list where the names are the DOIs and the values are the counts of the search term in each DOI.

Examples

```
## Not run:
get_count("climate change", c("10.1038/nature12350", "10.1126/science.1259855"))

## End(Not run)
```

`map_country_codes` *Draw Map Map of Country Codes*

Description

This function downloads and plots the GADM and IPBES regions according to a list of ISO codes provided.

Usage

```
map_country_codes(
  data = NULL,
  values = NULL,
  map_type = "countries",
  geodata_path = tempfile()
)
```

Arguments

`data`	`data.frame` or `tibble` containing at least one column called `iso2c` or ‘iso3c.
`values`	The name of the column to be plotted as the value in the maps. This is also used for the legend title.
`map_type`	A character string specifying the type of map to plot. Must be one of 'countries', 'regions', or 'subregions'. Default is 'countries'.
`geodata_path`	A character string specifying the path to store the geospatial directory to download data to. Default is a temporary file.

Value

A ggplot object representing the map of the specified map_type.

Examples

```
map_country_codes()
map_country_codes(iso3c = c("USA", "CAN"), map_type = "regions")
```

`oa_request_IPBES` *Get bibliographic records from OpenAlex database*

Description

This a slight adaptation from the function `oa_request` from the package openalexR It has the additional argument `output_path` to save the results in a file and not compile them in memory. When the transfer is interrupted, the existing files are not overwritten but skipped. **From Here the original documentation:** `oa_request` makes a request and downloads bibliographic records from OpenAlex database https://openalex.org/. The function `oa_request` queries OpenAlex database using a query formulated through the function `oa_query`.

Usage

```
oa_request_IPBES(
  query_url,
  per_page = 200,
  paging = "cursor",
  pages = NULL,
  count_only = FALSE,
  mailto = openalexR:::oa_email(),
  api_key = openalexR:::oa_apikey(),
  verbose = FALSE,
  output_path = NULL
)
```

Arguments

`query_url`	Character string. A search query formulated using the OpenAlex API language and can be generated with `oa_query`.
`per_page`	Numeric. Number of items to download per page. The per-page argument can assume any number between 1 and 200. Defaults to 200.
`paging`	Character. Either "cursor" for cursor paging or "page" for basic paging. When used with `options$sample` and or `pages`, paging is also automatically set to basic paging: `paging = "page"` to avoid duplicates and get the right page. See https://docs.openalex.org/how-to-use-the-api/get-lists-of-entities/paging.
`pages`	Integer vector. The range of pages to return. If NULL, return all pages.
`count_only`	Logical. If TRUE, the function returns only the number of item matching the query. Defaults to FALSE.
`mailto`	Character string. Gives OpenAlex an email to enter the polite pool.
`api_key`	Character string. Your OpenAlex Premium API key, if available.
`verbose`	Logical. If TRUE, print information about the querying process. Defaults to TRUE.
`output_path`	Character string. If `NULL` (default), the results are compiled in memory as in the original function. If a character string, the results are saved in files named `page_PAGENO.rds` in the output path `output_path`.

Value

a data.frame or a list of bibliographic records.

For more extensive information about OpenAlex API, please visit: https://docs.openalex.org

Examples

```
## Not run:

### EXAMPLE 1: Full record about an entity.

# Query to obtain all information about a particular work/author/institution/etc.:

#  The following paper is associated to the OpenAlex-id W2755950973.

#  Aria, M., & Cuccurullo, C. (2017). bibliometrix:
#   An R-tool for comprehensive science mapping analysis.
#   Journal of informetrics, 11(4), 959-975.


query_work <- oa_query(
    identifier = "W2755950973",
    entity = "works",
    endpoint = "https://api.openalex.org"
)

res <- oa_request(
    query_url = query_work,
    count_only = FALSE,
    verbose = FALSE
)

#  The author Massimo Aria is associated to the OpenAlex-id A5069892096.


query_author <- oa_query(
    identifier = "A5069892096",
    entity = "authors",
    endpoint = "https://api.openalex.org"
)

res <- oa_request(
    query_url = query_author,
    count_only = FALSE,
    verbose = FALSE
)



### EXAMPLE 2: all works citing a particular work.

# Query to search all works citing the article:
#  Aria, M., & Cuccurullo, C. (2017). bibliometrix:
#   An R-tool for comprehensive science mapping analysis.
#   Journal of informetrics, 11(4), 959-975.

#  published in 2021.
#  The paper is associated to the OpenAlex id W2755950973.

#  Results have to be sorted by relevance score in a descending order.

query2 <- oa_query(
    identifier = NULL,
    entity = "works",
    filter = "cites:W2755950973",
    from_publication_date = "2021-01-01",
    to_publication_date = "2021-12-31",
    search = NULL,
    endpoint = "https://api.openalex.org"
)

res2 <- oa_request(
    query_url = query2,
    count_only = FALSE,
    verbose = FALSE
)

### EXAMPLE 3: All works matching a string in their title

# Query to search all works containing the exact string
```

```
# "bibliometric analysis" OR "science mapping" in the title, published in 2020 or 2021.

# Results have to be sorted by relevance score in a descending order.


query3 <- oa_query(
    identifier = NULL,
    entity = "works",
    filter = 'title.search:"bibliometric analysis"|"science mapping"',
    from_publication_date = "2020-01-01",
    to_publication_date = "2021-12-31",
    search = NULL,
    endpoint = "https://api.openalex.org"
)

res3 <- oa_request(
    query_url = query3,
    count_only = FALSE,
    verbose = FALSE
)

### EXAMPLE 4: How to check how many works match a query
# Query to search all works containing the exact string
# "bibliometric analysis" OR "science mapping" in the title, published in 2020 or 2021.

# Query only to know how many works could be retrieved (count_only=TRUE)

query4 <- oa_query(
    identifier = NULL,
    entity = "works",
    filter = 'title.search:"bibliometric analysis"|"science mapping"',
    from_publication_date = "2020-01-01",
    to_publication_date = "2021-12-31",
    search = NULL,
    endpoint = "https://api.openalex.org"
)

res4 <- oa_request(
    query_url = query4,
    count_only = TRUE,
    verbose = FALSE
)

res4$count # number of items retrieved by our query

## End(Not run)
```

plot_snowball *Plot Snowball*

Description

This function takes a snowball object and a name, and creates two plots: one sized by cited_by_count and the other by cited_by_count_by_year. The plots are saved as both PDF and PNG in the specified path.

Usage

```
plot_snowball(snowball, name, path = "figures")
```

Arguments

`snowball`	A snowball object containing the data to be plotted.
`name`	The name to be used in the plot titles and file names.
`path`	The path where the plot files will be saved. Default is "figures".

Value

No return value, called for side effects.

Examples

```
## Not run:
plot_snowball(snowball, "example")

## End(Not run)
```

`plot_snowball_interactive`

Plot an Interactive Citation Network from a Snowball Search

Description

This function creates a interactive snowball seaarch network using the networkD3 package.

Usage

```
plot_snowball_interactive(snowball, key_works, file)
```

Arguments

`snowball`	The snowball object containing the network data. The object is returned from the oa_snowball function in the ‘openalexR“ package
`key_works`	A data frame, as returned. e.g. by `oa_fetch(entity = "works", ...`, containing the key-works from the snowball search which will be highlighted in the network.
`file`	The file name to save the network to. TThe directory has tro esxist. Default: `NULL`, i.e. not saved.

Value

A networkD3 object representing the interactive network plot.

Examples

```
## Not run:
plot_snowball_interactive(snowball, key_works, file)

## End(Not run)
```

`table_dt` *Create a DataTable with various export options*

Description

This is a wrapper around the DT::datatable() function that adds buttons for exporting the table to CSV, Excel, and PDF, as well as a print button. It also enables scrolling and fixed columns by default.

Usage

```
table_dt(
  data,
  fn = "datatable",
 buttons = list(list(extend = "csv", filename = fn), list(extend = "excel", filename =
   fn), list(extend = "pdf", filename = fn, orientation = "landscape", customize =
   DT::JS("function(doc) {", "  doc.defaultStyle.fontSize = 5;", "}")), "print"),
  scroller = TRUE,
  scrollY = JS("window.innerHeight * 0.7 + 'px'"),
  scrollX = TRUE,
  fixedColumns = list(leftColumns = 4),
  escape = FALSE,
  ...
)
```

Arguments

`data`	The data to be displayed in the DataTable
`fn`	The filename to be used for the exported files when using the default button definition
`buttons`	A list of buttons to be displayed in the DataTable
`scroller`	Logical value indicating whether to enable scrolling in the DataTable
`scrollY`	JavaScript code to set the height of the DataTable
`scrollX`	Logical value indicating whether to enable horizontal scrolling in the DataTable
`fixedColumns`	A list specifying the number of fixed columns in the DataTable
`escape`	Logical value indicating whether to escape HTML entities in the DataTable
`...`	Additional options to be passed to the DT::datatable function

Value

A DataTable object

Examples

```
table_dt(iris)
```

`to_xlsx` *Convert Snowball to XLSX*

Description

This function takes a snowball object and a filename, and converts the snowball object to an XLSX file.

Usage

```
to_xlsx(snowball, xls_filename = NULL)
```

Arguments

`snowball`	A snowball object containing the data to be converted.
`xls_filename`	If not ‘`NULL` the name of the XLSX file to be created. if `NULL` the data will not be saved in a csv but only returned invisilbly

Value

Invisibly the `data.frame` generated

Examples

```
## Not run:
to_xlsx(snowball, "example.xlsx")

## End(Not run)
```

Index