Package 'pleiades'

October 7, 2020

Title Interface to the 'Pleiades' 'Archeological' Database

Description Provides a set of functions for interacting with the 'Pleiades' (<https://pleiades.stoa.org/>) 'API', including getting status data, places data, and creating a 'GeoJSON' based map on 'GitHub' 'gists'.

Version 0.3.0

License MIT + file LICENSE

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

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

Encoding UTF-8

Language en-US

Imports DBI, dplyr, dbplyr, RSQLite, crul, jsonlite, gistr, rappdirs

Suggests testthat, knitr, rmarkdown, markdown, vcr

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Scott Chamberlain [aut, cre]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2020-10-07 18:00:02 UTC

R topics documented:

`pleiades-package` *pleiades*

Description

Interface to the Pleiades Archeological Database

Author(s)

Scott Chamberlain

`pl_cache` *Cache data locally for later usage.*

Description

Cache data locally for later usage.

Usage

```
pl_cache(force = FALSE, ...)

pl_cache_clear(which = NULL, prompt = TRUE)
```

Arguments

`force`	(logical) Force update of the cache. Default: `FALSE`
`...`	Curl options, see `curl::curl_options()`
`which`	(character) One of locations, names, or places.
`prompt`	(logical) Prompt before clearing all files in cache? No prompt used when DOIs passed in. Default: `TRUE`

Details

data are cached in `rappdirs::user_cache_dir("pleiades")`

Examples

```
## Not run:
pl_cache()
# pl_cache(force = TRUE)

# clear all files
# pl_cache_clear()

# clear a single file
# pl_cache_clear(which = "locations")
# pl_cache_clear(which = "places")
# pl_cache_clear(which = "names")

## End(Not run)
```

`pl_gist` *Make an interactive map to view in the browser as a GitHub gist*

Description

Make an interactive map to view in the browser as a GitHub gist

Usage

```
pl_gist(x, file = NULL, description = "", public = TRUE, browse = TRUE, ...)
```

Arguments

`x`	Output from `pl_places()`
`file`	(character) File name (without file extension) for your geojson file. Default is 'gistmap'.
`description`	(character) Description for the Github gist, or leave to default (=no description)
`public`	(logical) Whether gist is public (default: `TRUE`)
`browse`	(logical) If `TRUE` (default) the map opens in your default browser.
`...`	Curl options, see `curl::curl_options()`

Details

There are two ways to authorise gistr to work with your GitHub account:

- Generate a personal access token at https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token and record in the GITHUB_PAT envar.
- Interactively login into your GitHub account and authorise with OAuth

Using the GITHUB_PAT option is recommended.

Value

Creates a gist on your GitHub account

Examples

```
## Not run:
x <- pl_places(place_id = 462471)
pl_gist(x)

## End(Not run)
```

`pl_places` *Get data for a place given a place ID*

Description

Get data for a place given a place ID

Usage

```
pl_places(place_id, ...)
```

Arguments

`place_id`	(integer/numeric) A place ID
`...`	Curl options, see `curl::curl_options()`

Examples

```
## Not run:
pl_places(place_id = 462471)

## End(Not run)
```

`pl_search` *Search for a place, name or location.*

Description

This function searches a locally created SQLite database created from csv files.

Usage

```
pl_search(query = NULL, ...)

pl_search_loc(query = NULL, ...)

pl_search_names(query = NULL, ...)

pl_search_places(query = NULL, ...)
```

Arguments

query	A place ID. If left NULL, returns the table, which is of class `tbl`, which can then be passed on to other **dplyr** functions.
...	Further args passed on to `dplyr::tbl()`

Details

On the first query if not run before, the function takes a bit to get the raw data (if not already gotten), temporarily load the raw csv data, then create a SQLite database, and create the pointer to it. Subsequent calls should be very fast.

There is a function pl_cache, used to download the raw csv files. That function is run internally in these functions if you have not run it before, or if only some fo the files are present.

Note

Requires `RSQLite` package

Examples

```
## Not run:
pl_search()
pl_search_loc()
pl_search_names()
pl_search_places()

pl_search_loc("SELECT * FROM locations limit 5")
pl_search_names("SELECT * FROM names limit 5")
pl_search_places("SELECT * FROM places limit 5")

library(dplyr)
locs <- pl_search("SELECT * FROM locations limit 1000") %>%
  select(pid, reprLat, reprLong)
nms <- pl_search("SELECT * FROM names limit 1000") %>% select(pid)
left_join(locs, nms, "pid", copy = TRUE) %>% collect %>% NROW

## End(Not run)
```

`pl_status` *Get Pleiades status data, number of places, number of locations, number of names*

Description

Get Pleiades status data, number of places, number of locations, number of names

Usage

```
pl_status(...)
```

Arguments

`...`	Curl options, see `curl::curl_options()`

Examples

```
## Not run:
pl_status()

## End(Not run)
```

Index