

# Package ‘pleiades’

December 20, 2016

**Title** Interface to the 'Pleiades' 'Archeological' Database

**Description** Provides a set of functions for interacting with the 'Pleiades' 'API', including getting status data, places data, and creating a 'GeoJSON' based map on 'GitHub' 'gists'.

**Version** 0.1.0

**License** MIT + file LICENSE

**URL** <https://github.com/ropensci/pleiades>

**BugReports** <https://github.com/ropensci/pleiades/issues>

**Imports** dplyr (>= 0.5.0), crul (>= 0.1.6), jsonlite (>= 1.1), gistr (>= 0.3.6), rappdirs, RSQLite

**Suggests** roxygen2 (>= 5.0.1), testthat, knitr

**VignetteBuilder** knitr

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Author** Scott Chamberlain [aut, cre]

**Maintainer** Scott Chamberlain <myrmecocystus@gmail.com>

**Repository** CRAN

**Date/Publication** 2016-12-20 01:51:22

## R topics documented:

pleiades . . . . .	2
pl_cache . . . . .	2
pl_gist . . . . .	3
pl_places . . . . .	4
pl_search . . . . .	4
pl_status . . . . .	5

<b>Index</b>	<b>6</b>
--------------	----------

---

pleiades	<b>pleiades</b>
----------	-----------------

---

**Description****pleiades****Author(s)**

Scott Chamberlain &lt;myrmecocystus@gmail.com&gt;

---

pl_cache	<i>Cache data locally for later usage.</i>
----------	--

---

**Description**

Cache data locally for later usage.

**Usage**`pl_cache(force = FALSE, ...)``pl_cache_clear(which = NULL, prompt = TRUE)`**Arguments**

<code>force</code>	(logical) Force update of the cache. Default: FALSE
<code>...</code>	Curl options, see <a href="#">curl_options</a>
<code>which</code>	(character) One of locations, names, or places.
<code>prompt</code>	(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 <a href="#">pl_places</a>
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 <a href="#">curl_options</a>

## Details

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

- Generate a personal access token at <https://help.github.com/articles/creating-an-access-token-for-command-line-use> and record in the GITHUB\_PAT envvar.
- 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	<i>Get data for a place given a place ID</i>
-----------	--

---

**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 <a href="#">curl_options</a>

**Examples**

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

## End(Not run)
```

---

pl_search	<i>Search for a place, name or location.</i>
-----------	--

---

**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 <code>tbl</code> , which can then be passed on to other <b>dplyr</b> functions.
...	Further args passed on to <a href="#">tbl</a>

## 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 of the files are present.

## 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	<i>Get Pleiades status data, number of places, number of locations, number of names</i>
-----------	---

---

## Description

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

## Usage

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

## Arguments

... Curl options, see [curl\\_options](#)

## Examples

```
## Not run:
pl_status()

## End(Not run)
```

# Index

`crul (pleiades)`, 2  
`curl_options`, 2–5

`pl_cache`, 2, 5  
`pl_cache_clear (pl_cache)`, 2  
`pl_gist`, 3  
`pl_places`, 3, 4  
`pl_search`, 4  
`pl_search_loc (pl_search)`, 4  
`pl_search_names (pl_search)`, 4  
`pl_search_places (pl_search)`, 4  
`pl_status`, 5  
`pleiades`, 2  
`pleiades-package (pleiades)`, 2

`tbl`, 4