Package 'theiaR'

July 22, 2025

Title Download and Manage Data from Theia

Version 0.4.0

Description Provides a simple interface to search available data provided by Theia (<https://theia.cnes.fr>), download it, and manage it. Data can be downloaded based on a search result or from a cart file downloaded from Theia website.

Language en-US

Depends R (>= 3.5)

Imports askpass (>= 1.1), httr (>= 1.3), R6 (>= 2.3), raster (>= 2.6), tools (>= 3.5), XML (>= 3.86)

License GPL (>= 3.0)

URL https://github.com/norival/theiaR

BugReports https://github.com/norival/theiaR/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, gdalcubes

Collate 'TheiaAuth.R' 'TheiaTile.R' 'TheiaCollection.R' 'TheiaQuery.R' 'theiaR.R' 'utils.R'

VignetteBuilder knitr

NeedsCompilation no

Author Xavier Laviron [aut, cre] (ORCID: <https://orcid.org/0000-0002-9882-3253>)

Maintainer Xavier Laviron <xavier@norival.dev>

Repository CRAN

Date/Publication 2020-11-19 09:30:02 UTC

Contents

`TheiaAuth` *Authentication system to Theia website*

Description

Generate and manage authentication to Theia website from login and password. It requests a token to download tiles when created and automatically request a new one when it has expired (after 2h). It is used to download tiles from TheiaTile and TheiaCollection objects.

Usage

```
a <- TheiaAuth$new(auth.file)

a$token()
```

Arguments

a: A `TheiaAuth` object

auth.file The path to the file containing login and password. It will be created if it does not exist. See 'Details' for more informations

Details

`TheiaAuth$new(auth.file)` Create a new instance of the class

`a$token()` Return the current token or generate a next one if it has expired

This class is used to manage authentication to Theia website, without intervention from the user. Login and password must be stored in a separate text file with these two lines:

login password

File content is read each time authentication is needed (to request a new token), so login and password are not stored in R's memory. If this file does not exist, R will prompt you to enter your login and password and will create the file.

Examples

```
## Not run:
# create an authentication object
myauth <- TheiaAuth$new("path_to_auth_file.txt")

# show the access token (and request a new one if needed)
myauth$token

## End(Not run)
```

`TheiaCollection` *A collection of tiles from Theia*

Description

Generate and manage collection of tiles from Theia. This collection can be created either from a cart file ('.meta4') downloaded from Theia website, from a `TheiaQuery` object or from a list of `TheiaTile` (not implemented yet).

Usage

```
c <- TheiaCollection$new(cart.path = NULL,
                         tiles     = NULL,
                         query     = NULL,
                         dir.path  = NULL,
                         check     = TRUE)
                         quiet     = TRUE)

c$download(auth, overwrite = FALSE, check = TRUE, quiet = TRUE)
c$check()
c$status
c$extract(overwrite = FALSE, dest.dir = NULL)
c$read(bands)
c$as_gdalcube(out.file = "gdalcube_collection.sqlite")
```

Arguments

c: A `TheiaCollection` object

dir.path: The path to the directory containing zip files

check: Whether or not to check existing files on collection's creation

quiet: Control verbose output

tiles: A list of TheiaTile objects

cart: An XML cart parsed from a 'meta4' file downloaded from Theia website. Used only if Collection is created from a cart

query: A TheiaQuery object, used only if collection is created from a TheiaQuery object. Can also be a list with search terms. In this case, it will create a 'TheiaQuery' object from it. See `TheiaQuery` for details on query syntax

auth: A character string giving the file path to Theia credentials. Or a `TheiaAuth` object

overwrite: Overwrite existing tiles (default to 'FALSE')

bands: A character vector of bands to load from tiles

out.file: Filename to store gdalcubes' image collection

Details

`TheiaCollection$new()` Create a new instance of the class

`c$download(overwrite = FALSE, check = TRUE)` Download the tiles of the collection and check the resulting files

`$ccheck()` Check the tiles of the collection

`c$status` Return the status of each tile of the collection

`c$extract(overwrite = FALSE, dest.dir = NULL)` Extract archives to dest.dir if supplied, or to the same directory as the archives otherwise

`c$read(bands)` Read requested bands, apply corrections on values (as specified in Theia's product information), and return a list of RasterStack objects (one stack per tile)

`c$as_gdalcube(out.file)` Create a 'gdalcubes' image collection from downloaded tiles. See https://github.com/appelmar/gdalcubes_R for more details.

Examples

```
# Create a collection from a query
## Create a query to Theia database, looking for tiles from Sentinel2
## satellite around Grenoble, between 2018-07-01 and 2018-07-06.

query <- list(collection = "SENTINEL2",
              town       = "Grenoble",
              start.date = "2018-07-01",
              end.date   = "2018-07-06")

## Create a collecion of tiles from this query
mycollection <- TheiaCollection$new(query = query, dir.path = ".")

print(mycollection)


# Alternatively, you can create a collection from a cart file (that you can
# download from Theia's website)
cart.path <- system.file("extdata", "cart.meta4", package = "theiaR")

mycollection <- TheiaCollection$new(cart.path = cart.path,
                                    dir.path  = ".")

print(mycollection)
```

```
## Not run:
# Download the tiles in the collection
mycollection$download()

## End(Not run)

## Not run:
# Finally, you can extract zip archives containing the tiles
mycollection$extract(overwrite = FALSE)

## End(Not run)
```

`TheiaQuery` *A query to the Theia website*

Description

Generate an send a query to Theia web API to get and download tiles based on input given by the user.

Usage

```
q <- TheiaQuery$new(query)

q$update_token()
q$submit()
```

Arguments

q: A `TheiaQuery` object

query: `list`, the users' request, see 'Queries' for more informations

Details

`TheiaQuery$new()` Create a new instance of the class, parse 'query' list and submit the query to Theia to retrieve files catalog

`q$submit()` Submit the query to Theia and get a list of tiles corresponding to search criteria

Queries

Search criteria are given with a 'list' accepting these fields:

- collection: The collection to look for. Accepted values are: 'SENTINEL2', 'LANDSAT', 'Landsat57', 'SpotWorldHeritage', 'Snow'. Defaults to 'SENTINEL2'
- platform: The platform to look for. Accepted values are: 'LANDSAT5', 'LANDSAT7', 'LANDSAT8', 'SPOT1', 'SPOT2', 'SPOT3', 'SPOT4', 'SPOT5', 'SENTINEL2A', 'SENTINEL2B'

- level: Processing level. Accepted values are: 'LEVEL1C', 'LEVEL2A', LEVEL3A', 'N2A'. Defaults to 'LEVEL2A' (or 'N2A' if querying Landsat57 collection).
- town: The location to look for. Give a common town name.
- tile: The tile identifier to retrieve.
- start.date: The first date to look for (format: YYYY-MM-DD).
- end.date: The last date to look for (format: YYYY-MM-DD). Must be after start.date. Defaults to today's date.
- latitude: The x coordinate of a point
- longitude: The y coordinate of a point
- latmin: The minimum latitude to search
- latmax: The maximum latitude to search
- lonmin: The minimum longitude to search
- lonmax: The maximum longitude to search
- orbit.number: The orbit number
- rel.orbit.number: The relative orbit number
- max.clouds: The maximum of cloud cover wanted (0-100)
- max.records: The maximum of tiles to search

See Also

https://github.com/olivierhagolle/theia_download for an alternative download method based on Python. Inspiration for this function.

Examples

```
# Create a query to Theia database, looking for tiles from Sentinel2
# satellite around Grenoble, between 2018-07-01 and 2018-07-06.

query <- list(collection = "SENTINEL2",
              town       = "Grenoble",
              start.date = "2018-07-01",
              end.date   = "2018-07-06")
q <- TheiaQuery$new(query)

# Show informations on found tiles
print(q$tiles)
```

---

theiaR *theiaR: search, download and manage theia data*

Description

Search, manage and download data from Theia website

`TheiaTile` *A tile from Theia*

Description

Generate and manage a tile from Theia (download, check, load).

Usage

```
t <- TheiaTile$new(file.path,
                   url,
                   file.hash,
                   check = TRUE,
                   quiet = TRUE)

t$download(overwrite = FALSE, check = TRUE, quiet = TRUE)
t$check()
t$extract(overwrite = FALSE, dest.dir = NULL)
t$read(bands)
```

Arguments

t: A `TheiaTile` object

file.path: The path to the zip file containing the tile

url: The url to download the tile

file.hash: The md5sum used to check the zip file

check: Whether or not to check existing files on tile's creation

quiet: Control verbose output

auth: A character string giving the file path to Theia credentials. Or a `TheiaAuth` object

overwrite: Overwrite existing tiles (default to 'FALSE')

bands: A character vector of bands to load from tiles

Details

`TheiaTile$new(file.path, url, file.hash, check)` Create a new instance of the class

`t$download(auth, overwrite = FALSE, check = TRUE)` Download the tiles of the collection and check the resulting files

`t$check()` Check the tiles of the collection

`t$extract(overwrite = FALSE, dest.dir = NULL)` Extract archive to dest.dir if supplied, or to the same directory as the archive otherwise

`t$read(bands)` Read requested bands, apply corrections on values (as specified in Theia's product information), and return a RasterStack

`t$bands` List bands available in the tile

Index