Package 'spectator'

October 14, 2022

Title Interface to the 'Spectator Earth' API

Version 0.1.1

Description Provides interface to the 'Spectator Earth' API <https://api.spectator.earth/>, mainly for obtaining the acquisition plans and satellite overpasses for Sentinel-1, Sentinel-2, Landsat-8 and Landsat-9 satellites. Current position and trajectory can also be obtained for a much larger set of satellites. It is also possible to search the archive for available images over the area of interest for a given (past) period, get the URL links to download the whole image tiles, or alternatively to download the image for just the area of interest based on selected spectral bands.

License GPL-3

Depends R (>= 3.5.0)

Imports geojsonsf, httr, sf

Suggests calendar, calendR, httptest, knitr, lubridate, lutz, maps, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

Author Zivan Karaman [aut, cre, cph] (<https://orcid.org/0000-0002-8933-4589>)

Maintainer Zivan Karaman <zivan.karaman@gmail.com>

Repository CRAN

Date/Publication 2022-03-29 11:40:03 UTC

R topics documented:

`spectator-package` *Package providing interface to the 'Spectator Earth' API*

Description

The `spectator` package for R was developed to allow access to 'Spectator Earth' API from R. Spectator Earth offers a Web app providing Earth Observation imagery, mainly from open data satellites like the Sentinel and the Landsat family. These features are also exposed through an API, and the goal of the `spectator` package is to provide easy access to this functionality from R.

The main functions allow to retrieve the acquisition plans for Sentinel-1, Sentinel-2, Landsat-8 and Landsat-9 satellites and to get the past or (near)future overpasses over an area of interest for these satellites. It is also possible to search the archive for available images over the area of interest for a given (past) period, get the URL links to download the whole image tiles, or alternatively to download the image for just the area of interest based on selected spectral bands.

One can also get a current position and trajectory for a much larger set of satellites.

Other functions might be added in subsequent releases of the package.

Demos: `demo(package = "spectator")`

API key

Some of the functions (mainly those specific to Sentinel and Landsat satellites) require to pass an API key as a parameter to the function (because the underlying API endpoint requires it). The API key is automatically generated for every registered user at `https://app.spectator.earth`. You can find it under 'Your profile' (bottom left button) and copy it to clipboard. The functions in the `spectator` package by default retrieve the API key from the environment variable `"spectator_earth_api_key"`. You can choose any other way of providing it, but keep in mind that for security reasons it is **NOT** recommended to hard-code (include it as clear text) it in your scripts.

Author(s)

Zivan Karaman zivan.karaman@gmail.com

`GetAcquisitionPlan` *Gets a satellite acquisition plan*

Description

Retrieves the acquisition plans for the Sentinel 1 & 2 and Landsat 8 & 9 satellites.

Usage

```
GetAcquisitionPlan(satellites = NULL, date = NULL)
```

Arguments

`satellites`	character vector, if specified only the listed satellites will be retrieved, if `NULL` (default value) the acquisition plans for all possible satellites will be retrieved. For simplicity, the satellites names can be abbreviated to "S-1A", "S-1B", "S-2A", "S-2B", "L-8", "L-9" or "S1A", "S1B", "S2A", "S2B", "L8", "L9". Default: NULL
`date`	date or character convertible to date by `as.Date`, indicating the day for which the acquisition plans are requested. If `NULL` (default value), today's date is used. If too far in the future, will return empty data set. Default: NULL

Details

For Sentinels the acquisition plans usually have a range of 10-15 days, while for Landsat-8 it is 2-4 days. The time range that you can view is limited to 24 hours due to a large number of polygons.

Value

Object of class '`sf`' with '`POLYGON`' geometry type. The attributes of the output will vary, depending on the satellite. For more information check out acquisition plan file descriptions for Sentinel-1, Sentinel-2, Landsat-8, Landsat-9

Data source

Based on the files provided by ESA (Sentinel-1, Sentinel-2) and USGS (Landsat-8), more information available on the above mentioned web pages.

Source

https://api.spectator.earth/#acquisition-plan

Examples

```
if(interactive()){
 library(sf)
 # get plans for all eligible satellites for today
 plans <- GetAcquisitionPlan()
 # explore the content of the data frame,
 # -> you'll see that the available attributes vary with the satellite
 # focus on Sentinel 2
 sat <- c("Sentinel-2A", "Sentinel-2B")
 # day after tomorrow
 day <- Sys.Date()  + 2
 plan <- GetAcquisitionPlan(satellites = sat, date = day)
 # do some nice graphs
 library(maps)
 map("world", fill = TRUE, col = "lightgrey")
 plot(st_geometry(plan), border = "red", add = TRUE)
 title(main = sprintf("%s acquisition plan for %s", paste(sat, collapse = "/"), day))
 }
```

`GetAllSatellites` *Gets all referenced satellites info*

Description

Gets the information about all the satellites known in the Spectator Earth database, and possibly their current positions.

Usage

```
GetAllSatellites(positions = TRUE)
```

Arguments

`positions`	logical indicating if the current position should be included. Default: TRUE

Value

If `positions` is `FALSE`, a data frame with following attributes:

`id` integer identifier

`name` character satellite name

`norad_id` integer satellite catalog number

`sensors` character type of sensors available on the satellite (SAR or Optical)

`open` logical whether the data produced by the satellite is freely accessible

`platform` character platform name

If `positions` is `TRUE`, object of class '`sf`' with '`POINT`' geometry type, with the same attributes as above.

Source

https://api.spectator.earth/#satellites

See Also

GetSatellite

Examples

```
if(interactive()){
 library(sf)
 # get all satellites withe their positions
 pos <- GetAllSatellites(positions = TRUE)
 # do some nice graphs
 library(maps)
 map("world", fill = TRUE, col = "lightgrey")
 # show open data satellites in green
 plot(st_geometry(subset(pos, open == TRUE)), add = TRUE, col = "green", pch = 15)
 # show others in red
 plot(st_geometry(subset(pos, open == FALSE)), add = TRUE, col = "red", pch = 16)
 # add labels
 xy <- st_coordinates(pos)
 # shift labels up to be able to read them
 xy[, 2] <- xy[, 2] + 2
 text(xy, labels = pos$name, cex = 0.5)
 }
```

GetHighResolutionImage

Get image limited to an area of interest

Description

Provides you with a high resolution image for the area within the acquisition boundaries, if you're not interested in downloading the whole image.

Usage

```
GetHighResolutionImage(
  aoi,
  id,
  bands,
  width,
  height,
  file = "image.jpg",
  api_key = Sys.getenv("spectator_earth_api_key")
)
```

Arguments

aoi	'sf' (or 'Spatial*') object defining the area of interest. Can be of any geometry as only the bounding box is used.
id	integer, id of the image from the SearchImages result
bands	integer vector of length 1 or 3 indicating the spectral bands to use for creating the image (typically the bands corresponding to Red, Green and Blue)
width	integer indicating the width of the image (in pixels)
height	integer indicating the height of the image (in pixels)
file	character indicating the name of the image file to create. Default: 'image.jpg'
api_key	character containing your API key. Default: Sys.getenv("spectator_earth_api_key")

Details

As a side effect, the image file is written to the provided path.

Value

The name of the image file is returned invisibly.

Source

https://api.spectator.earth/#high-resolution-image

See Also

SearchImages

Examples

```
if(interactive()){
 library(sf)
 my_key <- Sys.getenv("spectator_earth_api_key")
 # get the New York City Central Park shape as area of interest
 dsn <- system.file("extdata", "centralpark.geojson", package = "spectator")
 boundary <- sf::read_sf(dsn, as_tibble = FALSE)
 # search for May 2021 Sentinel 2 images
 catalog <- SearchImages(aoi = boundary, satellites = "S2",
     date_from = "2021-05-01", date_to = "2021-05-30",
     footprint = FALSE, api_key = my_key)
 # get the id of the image with minimal cloud coverage
 best_id <- catalog[order(catalog$cloud_cover_percentage), ][1, "id"]
 # get the high resolution image of the Central Park
 img <- GetHighResolutionImage(aoi = boundary, id = best_id, bands = c(4, 3, 2),
     width = 1024, height = 1024,
     file = tempfile(pattern = "img", fileext = ".jpg"),
     api_key = my_key)
 }
```

`GetImageryFilesList` *List of downloadable files*

Description

List of files that can be downloaded directly (separate files for every spectral band) for the given image.

Usage

```
GetImageryFilesList(
  id,
  all = FALSE,
  api_key = Sys.getenv("spectator_earth_api_key")
)
```

Arguments

`id`	integer indicating the image id (from `SearchImages`)
`all`	logical indicating if the auxiliary files should be included. Default: FALSE
`api_key`	character containing your API key. Default: `Sys.getenv("spectator_earth_api_key")`

Details

Besides the raw images (`jp2` files) as single bands, various auxiliary files are also available. These include image thumbnails, metadata, etc. By default, only the full-sized images are returned by the function. To download the files, all the paths should be prepended with `https://api.spectator.earth/imagery/{id}/fi`
The raw image files are quite big, if the area of interest is relatively small it might be better to use `GetHighResolutionImage`.

Value

A data frame with attributes

`name` character, name of the file

`path` character, path (relative) to download the file

`size` integer, size of the file (in bytes)

Source

https://api.spectator.earth/#imagery-files

See Also

`SearchImages`, `GetHighResolutionImage`

Examples

```
if(interactive()){
 library(sf)
 my_key <- Sys.getenv("spectator_earth_api_key")
 # get the New York City Central Park shape as area of interest
 dsn <- system.file("extdata", "centralpark.geojson", package = "spectator")
 boundary <- sf::read_sf(dsn, as_tibble = FALSE)
 # search for May 2021 Sentinel 2 images
 catalog <- SearchImages(aoi = boundary, satellites = "S2",
     date_from = "2021-05-01", date_to = "2021-05-30",
     footprint = FALSE, api_key = my_key)
 # get the id of the image with minimal cloud coverage
 best_id <- catalog[order(catalog$cloud_cover_percentage), ][1, "id"]
 # list all downloadable files for the image with minimal cloud coverage
 images <- GetImageryFilesList(best_id, all = FALSE, api_key = my_key)
 }
```

`GetOverpasses` *Gets overpasses for an area of interest*

Description

Retrieves the footprint polygons of past and/or (near)future overpasses of specified satellites over an area of interest.

Usage

```
GetOverpasses(
  aoi,
  satellites = NULL,
  days_before = 0,
  days_after = 7,
  acquisitions = TRUE,
  api_key = Sys.getenv("spectator_earth_api_key")
)
```

Arguments

`aoi`	'sf' (or 'Spatial*') object defining the area of interest. Can be of any geometry as only the bounding box is used.
`satellites`	character vector, if specified only the listed satellites will be retrieved, if NULL (default value) the acquisition plans for all possible satellites will be retrieved. For simplicity, the satellites names can be abbreviated to "S-1A", "S-1B", "S-2A", "S-2B", "L-8", "L-9" or "S1A", "S1B", "S2A", "S2B", "L8", "L9". Default: NULL
`days_before`	integer indicating the number of days before the current date for which overpasses should be computed. Default: 0

days_after	integer indicating the number of days after the current date for which overpasses should be computed. Default: 7
acquisitions	logical indicating if only the overpasses when the data acquisition will take place should be reported. Default: TRUE
api_key	character containing your API key. Default: Sys.getenv("spectator_earth_api_key")

Details

This function requires an API key that is automatically generated for every registered account at https://app.spectator.earth. You can find it under 'Your profile' (bottom left button). By default, the API key is retrieved from the environment variable spectator_earth_api_key, but you can choose any other way of providing it (it is recommended **NOT** to include it in your scripts).

Value

Object of class 'sf' with 'POLYGON' geometry type and attributes

id integer identifier

acquisitions logical whether the overpass collects the data

date POSIXct timestamp of the overpass (UTC - to be checked)

satellite character satellite name

Source

https://api.spectator.earth/#satellite-overpasses

Examples

```
if(interactive()){
 library(sf)
 # get the Luxembourg country shape as area of interest
 boundary <- read_sf(system.file("extdata", "luxembourg.geojson", package = "spectator"))
 # look for Sentinel-2 A and B, use shorthand notation, default time frame
 pass <- GetOverpasses(boundary, satellites = "S-2", acquisitions = TRUE)
 # do some nice graphs
 library(maps)
 days <- range(as.Date(pass$date))
 satellites <- sort(unique(pass$satellite))
 map(database = "world", region = c("Belgium", "Netherlands", "Germany", "Luxembourg",
     "France", "Switzerland"), col = "lightgrey", fill = TRUE)
 plot(sf::st_geometry(boundary), add = TRUE, col = "red", border = FALSE)
 plot(sf::st_geometry(pass), add = TRUE)
 title(main = sprintf("%s overpasses for period %s", paste(satellites, collapse = "/"),
                      paste(days, collapse = ":")))
 }
```

`GetSatellite` *Gets info for a satellite*

Description

Gets the information about the specified satellite, and possibly its current position.

Usage

```
GetSatellite(satellite, positions = TRUE)
```

Arguments

`satellite`	character name of the satellite for which to retrieve the trajectory. The satellite name is not case sensitive, and can be abbreviated as long as an unambiguous match can be obtained. Only one satellite can be queried at a time.
`positions`	logical indicating if the current position should be included. Default: TRUE

Value

If `positions` is `FALSE`, a single row data frame with following attributes:

`id` integer identifier

`name` character satellite name

`norad_id` integer satellite catalog number

`sensors` character type of sensors available on the satellite (SAR or Optical)

`open` logical whether the data produced by the satellite is freely accessible

`platform` character platform name

If `positions` is `TRUE`, a single row object of class '`sf`' with '`POINT`' geometry type, with the same attributes as above.

Source

https://api.spectator.earth/#satellites

See Also

`GetAllSatellites`

Examples

```
if(interactive()){
 library(sf)
 # get trajectory and current position for a selected satellite
 sat <- "SPOT-7"
 traj <- GetTrajectory(satellite = sat)
 pos <- GetSatellite(satellite = sat, positions = TRUE)
 # do some nice graphs
 library(maps)
 map("world", fill = TRUE, col = "lightgrey")
 plot(st_geometry(traj), lwd = 2, col = "red", add = TRUE)
 plot(st_geometry(pos), pch = 15, col = "green", cex = 1.5, add = TRUE)
 title(main = sprintf("current %s trajectory & position", sat))
 }
```

GetTrajectory *Gets a satellite trajectory*

Description

Gets the current trajectory for the specified satellite.

Usage

```
GetTrajectory(satellite)
```

Arguments

satellite	character name of the satellite for which to retrieve the trajectory. The satellite name is not case sensitive, and can be abbreviated as long as an unambiguous match can be obtained. Only one satellite can be queried at a time.

Value

Object of class 'sf' with 'LINESTRING' geometry type

Source

https://api.spectator.earth/#trajectories

Examples

```
if(interactive()){
 library(sf)
 # get trajectory and current position for a selected satellite
 sat <- "SPOT-7"
 traj <- GetTrajectory(satellite = sat)
 pos <- GetSatellite(satellite = sat, positions = TRUE)
 # do some nice graphs
```

```
library(maps)
map("world", fill = TRUE, col = "lightgrey")
plot(st_geometry(traj), lwd = 2, col = "red", add = TRUE)
plot(st_geometry(pos), pch = 15, col = "green", cex = 1.5, add = TRUE)
title(main = sprintf("current %s trajectory & position", sat))
}
```

`satellites_db` *Satellites database*

Description

List all the satellites available in the Spectator Earth database with main attributes

Usage

```
satellites_db
```

Format

A data frame with 48 rows and 6 variables:

`id` integer identifier

`name` character satellite name

`norad_id` integer satellite catalog number

`sensors` character type of sensors available on the satellite (SAR or Optical)

`open` logical whether the data produced by the satellite is freely accessible

`platform` character platform name

Details

The information in this data frame is a local cache of the list of all the satellites available in the Spectator Earth database. It enables the fast retrieval of some data by satellite name instead of id. The current up-to-date list of satellites described in Spectator Earth database can be obtained by `GetAllSatellites(positions = FALSE)`

Source

https://api.spectator.earth/#satellites

See Also

`GetAllSatellites`

SearchImages *Search Spectator database for available images*

Description

Returns the list of available images for an area of interest, specified time interval and selected satellites.

Usage

```
SearchImages(
  aoi,
  satellites = NULL,
  date_from = NULL,
  date_to = NULL,
  footprint = FALSE,
  api_key = Sys.getenv("spectator_earth_api_key")
)
```

Arguments

aoi	'sf' (or 'Spatial*') object defining the area of interest. Can be of any geometry as only the bounding box is used.
satellites	character vector, if specified only the listed satellites will be retrieved, if NULL (default value) the acquisition plans for all possible satellites will be retrieved. For simplicity, the satellites names can be abbreviated to "S-1A", "S-1B", "S-2A", "S-2B", "L-8", "L-9" or "S1A", "S1B", "S2A", "S2B", "L8", "L9". Default: NULL
date_from	date or character convertible to date by as.Date, indicating the earliest image date. Default: NULL
date_to	date or character convertible to date by as.Date, indicating the latest image date. Default: NULL
footprint	logical indicating if the polygons describing the image tiles should be returned. Default: FALSE
api_key	character containing your API key. Default: Sys.getenv("spectator_earth_api_key")

Details

The data frame contains some useful attributes: id which enables to download images using the functions GetImageryFilesList or GetHighResolutionImage, cloud_cover_percentage (for the whole image tile), satellite (name), begin_position_date and end_position_date indicating when the image was taken.

Value

Either a data frame (if 'footprint' is 'FALSE') or an object of class 'sf' with 'POLYGON' geometry type (if 'footprint' is 'TRUE').

Source

https://api.spectator.earth/#searching-for-images

Examples

```
if(interactive()){
 library(sf)
 my_key <- Sys.getenv("spectator_earth_api_key")
 # get the New York City Central Park shape as area of interest
 dsn <- system.file("extdata", "centralpark.geojson", package = "spectator")
 boundary <- sf::read_sf(dsn, as_tibble = FALSE)
 # search for May 2021 Sentinel 2 images
 catalog <- SearchImages(aoi = boundary, satellites = "S2",
     date_from = "2021-05-01", date_to = "2021-05-30",
     footprint = FALSE, api_key = my_key)
 }
```

Index