Package 'PlanetNICFI'

September 28, 2023

Type Package

Title Processing of the 'Planet NICFI' Satellite Imagery

Version 1.0.5

Date 2023-09-28

URL https://github.com/mlampros/PlanetNICFI

Description It includes functions to download and process the 'Planet NICFI' (Norway's International Climate and Forest Initiative) Satellite Imagery utilizing the Planet Mosaics API <https://developers.planet.com/docs/basemaps/reference/#tag/Basemaps-and-Mosaics>. 'GDAL' (library for raster and vector geospatial data formats) and 'aria2c' (paralleled download utility) must be installed and configured in the user's Operating System.

License GPL-3

Copyright inst/COPYRIGHTS

Encoding UTF-8

SystemRequirements aria2: apt-get install -y aria2 (deb), gdal-bin: apt-get install -y gdal-bin (deb), libgdal-dev: apt-get install -y libgdal-dev (deb)

Depends R(>= 3.5.0)

Imports httr, sf, data.table, glue, utils, terra

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

RoxygenNote 7.2.3

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Lampros Mouselimis [aut, cre] (<https://orcid.org/0000-0002-8024-1546>), Planet Labs Inc [cph]

Maintainer Lampros Mouselimis <mouselimislampros@gmail.com>

Repository CRAN

Date/Publication 2023-09-28 07:50:02 UTC

R topics documented:

`aria2c_bulk_donwload` *Bulk download of files using 'aria2c'*

Description

Bulk download of files using 'aria2c'

Usage

```
aria2c_bulk_donwload(
  vector_or_file_path,
  default_directory,
  user = NULL,
  password = NULL,
  threads = 1,
  verbose = FALSE,
  secondary_args_aria = "--allow-overwrite --retry-wait=5 --max-tries=0"
)
```

Arguments

`vector_or_file_path`	either a vector of character strings or a valid path to a text file. See the output of the 'aria2c_download_paths()' function for the correct format.
`default_directory`	a character string specifying a valid path where the files will be saved
`user`	either NULL or a character string specifying the 'user' (normally this is the 'username' required in specific websites to have access and download files)
`password`	either NULL or a character string specifying the 'password' (normally this is the 'password' required in specific websites to have access and download files)
`threads`	an integer value specifying the number of threads to run in parallel
`verbose`	a boolean. If TRUE then information will be printed out in the console

secondary_args_aria
: a character vector specifying the additional parameters that can be passed to the 'aria2c' function. For instance, "–retry-wait": specifies the seconds to wait between retries and "–max-tries=0" means unlimited re-tries. See the References section for more details.

Value

a character vector based on the verbosity of the function

References

https://aria2.github.io/manual/en/html/aria2c.html

https://aria2.github.io/manual/en/html/aria2c.html#cmdoption-retry-wait

https://aria2.github.io/manual/en/html/aria2c.html#cmdoption-m

https://aria2.github.io/manual/en/html/aria2c.html#exit-status

Examples

```
## Not run:

require(PlanetNICFI)

#.................................
# first extract the available Mosaics
#.................................

api_key = 'use_your_planet_nicfi_API_key'

mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'monthly',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)

#.................................
# keep the mosaic of 'September 2020'
#.................................

keep_idx = 1
mosaic_ID = mosaic_files$dtbl_mosaic$id[keep_idx]


#.............................................................
# then extract the available Quad files for the Mosaic for an AOI
#.............................................................

wkt_file = system.file('data_files/Sugar_Cane_Bolivia.wkt', package = "PlanetNICFI")
WKT = readLines(wkt_file, warn = FALSE)
```

```
quad_files = nicfi_quads_bbox(planet_api_key = api_key,
                              mosaic_id = mosaic_ID,
                              bbox_AOI = NULL,
                              wkt_AOI = WKT,
                              page_size = 10,
                              crs_bbox = 4326,
                              verbose = TRUE)

#..............................
# formated aria2c download weblinks
#..............................

web_links_aria2c = aria2c_download_paths(mosaic_output = mosaic_files,
                                         mosaic_id = mosaic_ID,
                                         quads_output = quad_files,
                                         img_type = 'tif')

#....................................................
# download the .tif files that intersect with the bbox AOI
#....................................................

temp_dir_out = tempdir()

all_threads = parallel::detectCores()
set_threads = length(web_links_aria2c) / 2
num_threads = ifelse(set_threads < all_threads, set_threads, all_threads)
aria_args = '--allow-overwrite --file-allocation=none --retry-wait=5 --max-tries=0'

res_downl = aria2c_bulk_donwload(vector_or_file_path = web_links_aria2c,
                                 default_directory = temp_dir_out,
                                 user = NULL,
                                 password = NULL,
                                 threads = num_threads,
                                 verbose = TRUE,
                                 secondary_args_aria = aria_args)

## End(Not run)
```

`aria2c_download_paths` *Format Mosaic and Quad weblinks to serve as input to the 'aria2c_bulk_donwload' function*

Description

Format Mosaic and Quad weblinks to serve as input to the 'aria2c_bulk_donwload' function

Usage

```
aria2c_download_paths(mosaic_output, mosaic_id, quads_output, img_type = "tif")
```

Arguments

mosaic_output	this parameter must be the output list of the 'nicfi_mosaics()' function
mosaic_id	a character string specifying the mosaic-id as appears in the output column 'id' of the 'nicfi_mosaics()' function
quads_output	this parameter must be the output list of the 'nicfi_quads_bbox()' function
img_type	a character string specifying the image type to download. One of 'tif' or 'thumbnail'. The 'thumbnail' come with a .png image extension

Details

The 'thumbnail' are smaller in size and it might be a good idea to download these images first (just for an overview) before proceeding to the download of the .tif files (which are more than 100 MB each)

Value

a character vector

Examples

```
## Not run:

require(PlanetNICFI)

#.................................
# first extract the available Mosaics
#.................................

api_key = 'use_your_planet_nicfi_API_key'

mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'monthly',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)

#.................................
# keep the mosaic of 'September 2020'
#.................................

keep_idx = 1
mosaic_ID = mosaic_files$dtbl_mosaic$id[keep_idx]


#................................................
# then extract the available Quad files for the Mosaic
#................................................

wkt_file = system.file('data_files/Sugar_Cane_Bolivia.wkt', package = "PlanetNICFI")
```

```
WKT = readLines(wkt_file, warn = FALSE)

quad_files = nicfi_quads_bbox(planet_api_key = api_key,
                              mosaic_id = mosaic_ID,
                              bbox_AOI = NULL,
                              wkt_AOI = WKT,
                              page_size = 10,
                              crs_bbox = 4326,
                              verbose = TRUE)

#.............................
# download the .png thumbnails  (smaller size for overview)
#.............................

web_links_aria2c = aria2c_download_paths(mosaic_output = mosaic_files,
                                         mosaic_id = mosaic_ID,
                                         quads_output = quad_files,
                                         img_type = 'thumbnail')

#.........................
# download the .tif files
#.........................

web_links_aria2c = aria2c_download_paths(mosaic_output = mosaic_files,
                                         mosaic_id = mosaic_ID,
                                         quads_output = quad_files,
                                         img_type = 'tif')

## End(Not run)
```

`create_VRT_from_dir` *Create a Virtual Raster (VRT) file from the .tif files*

Description

Create a Virtual Raster (VRT) file from the .tif files

Usage

```
create_VRT_from_dir(
  dir_tifs,
  output_path_VRT,
  file_extension = ".tif",
  verbose = FALSE
)
```

Arguments

`dir_tifs`	a valid path to a directory where the .tif files are saved

output_path_VRT
: a valid path to a file where the Virtual Raster (VRT) will be saved

file_extension
: a character string specifying the image file extension from which the .vrt file will be built

verbose
: a boolean. If TRUE then information will be printed out in the console

Value

it doesn't return an object but it saves the output to a file

Examples

```
## Not run:

require(PlanetNICFI)

#.................................
# first extract the available Mosaics
#.................................

api_key = 'use_your_planet_nicfi_API_key'

mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'monthly',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)

#.................................
# keep the mosaic of 'September 2020'
#.................................

keep_idx = 1
mosaic_ID = mosaic_files$dtbl_mosaic$id[keep_idx]


#.............................................................
# then extract the available Quad files for the Mosaic for an AOI
#.............................................................

wkt_file = system.file('data_files/Sugar_Cane_Bolivia.wkt', package = "PlanetNICFI")
WKT = readLines(wkt_file, warn = FALSE)

quad_files = nicfi_quads_bbox(planet_api_key = api_key,
                              mosaic_id = mosaic_ID,
                              bbox_AOI = NULL,
                              wkt_AOI = WKT,
                              page_size = 10,
                              crs_bbox = 4326,
                              verbose = TRUE)
```

```
#..............................
# formated aria2c download weblinks
#..............................

web_links_aria2c = aria2c_download_paths(mosaic_output = mosaic_files,
                                         mosaic_id = mosaic_ID,
                                         quads_output = quad_files,
                                         img_type = 'tif')

#.....................................................
# download the .tif files that intersect with the bbox AOI
#.....................................................

temp_dir_out = tempdir()

all_threads = parallel::detectCores()
set_threads = length(web_links_aria2c) / 2
num_threads = ifelse(set_threads < all_threads, set_threads, all_threads)
aria_args = '--allow-overwrite --file-allocation=none --retry-wait=5 --max-tries=0'

res_downl = aria2c_bulk_donwload(vector_or_file_path = web_links_aria2c,
                                 default_directory = temp_dir_out,
                                 user = NULL,
                                 password = NULL,
                                 threads = num_threads,
                                 verbose = TRUE,
                                 secondary_args_aria = aria_args)

#..................................
# create a Virtual Raster (VRT) file from
# the downloaded .tif files
#..................................

VRT_out = file.path(temp_dir_out, glue::glue("{mosaic_ID}.vrt"))

res_vrt = create_VRT_from_dir(dir_tifs = temp_dir_out,
                              output_path_VRT = VRT_out,
                              file_extension = '.tif',
                              verbose = TRUE)

#................................................
# load the saved VRT file as raster (which might
# consist of multiple files, i.e. a mosaic) and plot it
#................................................

rst = terra::rast(VRT_out)
terra::plot(rst, axes = FALSE, legend = FALSE)


## End(Not run)
```

`nicfi_crop_images` *Crop the downloaded NICFI .tif or .vrt file using 'gdalwarp'*

Description

Crop the downloaded NICFI .tif or .vrt file using 'gdalwarp'

Usage

```
nicfi_crop_images(
  input_pth,
  output_pth,
  bbox_AOI,
  threads = 1,
  of = "GTiff",
  resize_method = "lanczos",
  verbose = FALSE
)
```

Arguments

`input_pth`	a valid path to either a .tif or a .vrt (virtual raster) file that should be cropped based on the bounding box using 'gdalwarp'
`output_pth`	a valid path to the output .tif file. This file path can also point to a .vrt file by setting the 'of' parameter to 'VRT'
`bbox_AOI`	a list of the format "list(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax)" that includes the bounding box 'xmin', 'xmax', 'ymin', 'ymax' coordinate values of the Area of Interest (AOI)
`threads`	an integer. In case that this parameter is greater than 1 then multiple threads will be utilized in the 'gdalwarp' function
`of`	a character string specifying the format of the saved file. The default is GeoTIFF (GTiff). For more information see the 'gdal_utils' function of the 'sf' package
`resize_method`	a character string specifying the resize method. Can be one of 'near', 'bilinear', 'cubic', 'cubicspline', 'lanczos', 'average', 'mode', 'max', 'min', 'med', 'q1', 'q3'. For more information see the 'r' parameter of the 'gdal_utils' function of the 'sf' package
`verbose`	a boolean. If TRUE then information will be printed out in the console

Value

a logical indicating success (i.e., TRUE); in case of failure, an error is raised

Examples

```
## Not run:

require(PlanetNICFI)

#.................................
# first extract the available Mosaics
#.................................

api_key = 'use_your_planet_nicfi_API_key'

mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'monthly',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)

#.................................
# keep the mosaic of 'September 2020'
#.................................

keep_idx = 1
mosaic_ID = mosaic_files$dtbl_mosaic$id[keep_idx]


#.............................................................
# then extract the available Quad files for the Mosaic for an AOI
#.............................................................

wkt_file = system.file('data_files/Sugar_Cane_Bolivia.wkt', package = "PlanetNICFI")
WKT = readLines(wkt_file, warn = FALSE)

quad_files = nicfi_quads_bbox(planet_api_key = api_key,
                              mosaic_id = mosaic_ID,
                              bbox_AOI = NULL,
                              wkt_AOI = WKT,
                              page_size = 10,
                              crs_bbox = 4326,
                              verbose = TRUE)

#..............................
# formated aria2c download weblinks
#..............................

web_links_aria2c = aria2c_download_paths(mosaic_output = mosaic_files,
                                         mosaic_id = mosaic_ID,
                                         quads_output = quad_files,
                                         img_type = 'tif')

#.....................................................
# download the .tif files that intersect with the bbox AOI
```

```
#.....................................................

temp_dir_out = tempdir()

all_threads = parallel::detectCores()
set_threads = length(web_links_aria2c) / 2
num_threads = ifelse(set_threads < all_threads, set_threads, all_threads)
aria_args = '--allow-overwrite --file-allocation=none --retry-wait=5 --max-tries=0'

res_downl = aria2c_bulk_donwload(vector_or_file_path = web_links_aria2c,
                                 default_directory = temp_dir_out,
                                 user = NULL,
                                 password = NULL,
                                 threads = num_threads,
                                 verbose = TRUE,
                                 secondary_args_aria = aria_args)

#...................................
# create a Virtual Raster (VRT) file from
# the downloaded .tif files
#...................................

VRT_out = file.path(temp_dir_out, glue::glue("{mosaic_ID}.vrt"))

res_vrt = create_VRT_from_dir(dir_tifs = temp_dir_out,
                              output_path_VRT = VRT_out,
                              file_extension = '.tif',
                              verbose = TRUE)

#.............................................
# Adjust the Coordinate Reference System of the
# bounding box from 4326 to the one of the .tif files
#.............................................

wkt_sf = sf::st_as_sfc(WKT, crs = 4326)
proj_info = proj_info_extract(path_to_raster = VRT_out)

wkt_transf = sf::st_transform(wkt_sf, crs = proj_info)
bbx_transf = sf::st_bbox(wkt_transf)


#.............................................
# crop the output .vrt file based on the bounding box
#.............................................

pth_crop_out = file.path(temp_dir_out, glue::glue("{mosaic_ID}_CROPPED.tif"))

bbx_crop = list(xmin = as.numeric(bbx_transf['xmin']),
                xmax = as.numeric(bbx_transf['xmax']),
                ymin = as.numeric(bbx_transf['ymin']),
                ymax = as.numeric(bbx_transf['ymax']))

warp_obj = nicfi_crop_images(input_pth = VRT_out,
```

```
                              output_pth = pth_crop_out,
                              bbox_AOI = bbx_crop,
                              threads = num_threads,
                              of = 'GTiff',
                              resize_method = 'lanczos',
                              verbose = TRUE)

## End(Not run)
```

`nicfi_mosaics` *Returns all 'monthly' or 'bi-annually' mosaic files of the NICFI data*

Description

Returns all 'monthly' or 'bi-annually' mosaic files of the NICFI data

Usage

```
nicfi_mosaics(
  planet_api_key,
  type = "monthly",
  crs_bbox = 4326,
  URL = "https://api.planet.com/basemaps/v1/mosaics",
  verbose = FALSE
)
```

Arguments

`planet_api_key`	a character string specifying the Planet API key (see the references on how to acquire this key)
`type`	a character string specifying the type of NICFI data. It can be either 'monthly' or 'bi_annually'
`crs_bbox`	an integer specifying the Coordinates Reference System for the bounding box computation only.
`URL`	this character string specifies the default URL which is required to come to the output mosaic metadata information
`verbose`	a boolean. If TRUE then information will be printed out in the console

Value

an object of class list

References

https://developers.planet.com/quickstart/

https://developers.planet.com/quickstart/apis/

https://developers.planet.com/docs/basemaps/reference/#tag/Basemaps-and-Mosaics

Examples

```
## Not run:

require(PlanetNICFI)

api_key = 'use_your_planet_nicfi_API_key'

#........
# monthly
#........

mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'monthly',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)
#............
# bi-annually
#............

mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'bi_annually',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)

#....................................
# WKT of the area covered from NICFI data
#....................................

nicfi_aoi = sf::st_as_sfc(mosaic_files$dtbl_mosaic$mosaic_wkt[1], crs = 4326)
cat(sf::st_as_text(nicfi_aoi))


## End(Not run)
```

`nicfi_quads_bbox` *Computes the NICFI Quads based on a mosaic-id and a specified Area of Interest (bounding box or Well Known Text)*

Description

Computes the NICFI Quads based on a mosaic-id and a specified Area of Interest (bounding box or Well Known Text)

Usage

```
nicfi_quads_bbox(
```

```
  planet_api_key,
  mosaic_id,
  bbox_AOI = NULL,
  wkt_AOI = NULL,
  page_size = 50,
  crs_bbox = 4326,
  verbose = FALSE
)
```

Arguments

planet_api_key	a character string specifying the Planet API key (see the references on how to acquire this key)
mosaic_id	a character string specifying the 'Mosaic' id as returned from the 'nicfi_mosaics()' function
bbox_AOI	either NULL or a list of the format "list(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax)" that includes the bounding box 'xmin', 'xmax', 'ymin', 'ymax' coordinate values of the Area of Interest (AOI) based on which the intersected NICFI Quads have to be computed
wkt_AOI	either NULL or a character string specifying the Well Known Text (WKT) of the Area of Interest (AOI) based on which the intersected NICFI Quads have to be computed
page_size	an integer value specifying the number of Quads to return (that intersect with the input bounding box or Well known text)
crs_bbox	an integer specifying the Coordinates Reference System for the bounding box computation only.
verbose	a boolean. If TRUE then information will be printed out in the console

Value

an object of class list

References

https://developers.planet.com/docs/basemaps/reference/#tag/Basemaps-and-Mosaics

Examples

```
## Not run:

require(PlanetNICFI)

#.................................
# first extract the available Mosaics
#.................................

api_key = 'use_your_planet_nicfi_API_key'
```

```
mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'monthly',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)

#................................
# keep the mosaic of 'September 2020'
#................................

keep_idx = 1
mosaic_ID = mosaic_files$dtbl_mosaic$id[keep_idx]


#................................................
# then extract the available Quad files for the Mosaic
#................................................

wkt_file = system.file('data_files/Sugar_Cane_Bolivia.wkt', package = "PlanetNICFI")
WKT = readLines(wkt_file, warn = FALSE)

quad_files = nicfi_quads_bbox(planet_api_key = api_key,
                              mosaic_id = mosaic_ID,
                              bbox_AOI = NULL,
                              wkt_AOI = WKT,
                              page_size = 10,
                              crs_bbox = 4326,
                              verbose = TRUE)

## End(Not run)
```

`proj_info_extract` *Extract the Projection from a (virtual) raster file*

Description

Extract the Projection from a (virtual) raster file

Usage

```
proj_info_extract(path_to_raster, verbose = FALSE)
```

Arguments

`path_to_raster`	a valid path to a raster file
`verbose`	a boolean. If TRUE then information will be printed out in the console

Value

a character string with the projection information

Examples

```
require(PlanetNICFI)

pth_vrt = system.file('data_files/virt_rast.vrt', package = "PlanetNICFI")

proj_info = proj_info_extract(path_to_raster = pth_vrt)
```

---

sequential_download_paths

Download the Planet NICFI images sequentially

Description

Download the Planet NICFI images sequentially

Usage

```
sequential_download_paths(
  aria2c_file_paths,
  default_directory,
  download_method = "wget",
  verbosity = 0
)
```

Arguments

aria2c_file_paths	a vector of character strings. See the output of the 'aria2c_download_paths()' function for the correct format.
default_directory	a character string specifying a valid path where the files will be saved
download_method	a character string specifying the download method. Can be for instance "wget", "curl" or any available method of the "download.file()" function
verbosity	an integer specifying the verbosity (between 0 and 2). If 0 then verbosity is disabled, if 1 then only essential verbosity is displayed and if 2 then all available information will be printed out in the console.

Details

This function does not require the 'aria2c' tool (system requirement) to download the imagery. It uses the 'utils::download.file()' function internally

Value

it doesn't return an R object but it saves the files to a directory

Examples

```
## Not run:

require(PlanetNICFI)

#.................................
# first extract the available Mosaics
#.................................

api_key = 'use_your_planet_nicfi_API_key'

mosaic_files = nicfi_mosaics(planet_api_key = api_key,
                             type = 'monthly',
                             crs_bbox = 4326,
                             URL = 'https://api.planet.com/basemaps/v1/mosaics',
                             verbose = TRUE)

#.................................
# keep the mosaic of 'September 2020'
#.................................

keep_idx = 1
mosaic_ID = mosaic_files$dtbl_mosaic$id[keep_idx]


#................................................
# then extract the available Quad files for the Mosaic
#................................................

wkt_file = system.file('data_files/Sugar_Cane_Bolivia.wkt', package = "PlanetNICFI")
WKT = readLines(wkt_file, warn = FALSE)

quad_files = nicfi_quads_bbox(planet_api_key = api_key,
                              mosaic_id = mosaic_ID,
                              bbox_AOI = NULL,
                              wkt_AOI = WKT,
                              page_size = 10,
                              crs_bbox = 4326,
                              verbose = TRUE)
#.....................
# download the .tif files
#.....................

web_links_aria2c = aria2c_download_paths(mosaic_output = mosaic_files,
                                         mosaic_id = mosaic_ID,
                                         quads_output = quad_files,
                                         img_type = 'tif')
```

```
DIR_SAVE = tempdir(check = FALSE)
print(DIR_SAVE)

res_dat = sequential_download_paths(aria2c_file_paths = web_links_aria2c,
                                    default_directory = DIR_SAVE,
                                    download_method = 'wget',
                                    verbosity = 1)

## End(Not run)
```

Index