Package: MODIStsp (via r-universe)

October 1, 2024

Title Find, Download and Process MODIS Land Products Data

Type Package

Version 2.1.0

Description Allows automating the creation of time series of rasters derived from MODIS satellite land products data. It performs several typical preprocessing steps such as download, mosaicking, reprojecting and resizing data acquired on a specified time period. All processing parameters can be set using a user-friendly GUI. Users can select which layers of the original MODIS HDF files they want to process, which additional quality indicators should be extracted from aggregated MODIS quality assurance layers and, in the case of surface reflectance products, which spectral indexes should be computed from the original reflectance bands. For each output layer, outputs are saved as single-band raster files corresponding to each available acquisition date. Virtual files allowing access to the entire time series as a single file are also created. Command-line execution exploiting a previously saved processing options file is also possible, allowing users to automatically update time series related to a MODIS product whenever a new image is available. For additional documentation refer to the following article: Busetto and Ranghetti (2016) <doi:10.1016/j.cageo.2016.08.020>.

License GPL-3

Depends R (>= 3.5.0)

Imports assertthat, bitops (>= 1.0-6), data.table (>= 1.9.6), gdalUtilities, geojsonio, httr (>= 1.4.2), jsonlite, parallel, raster (>= 3.3.13), sf (>= 0.9.3), stringr (>= 1.0.0), xml2 (>= 1.2.0), xts (>= 0.9-7)

Suggests dplyr, DT, formatR, ggplot2, grid, httptest, knitr, leafem (>= 0.1.3), leaflet, magrittr, mapedit (>= 0.6.0), png, rappdirs, rmarkdown, shiny, shinyalert (>= 3.0.0), shinydashboard, shinyFiles (>= 0.9.0), shinyjs, spelling, testthat, tibble, tibbletime, tidyr, qpdf, webshot, xtable

SystemRequirements GDAL (>= 2.1.2) with support for HDF4 format, PROJ (>= 4.9.1).

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

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

LazyData true

VignetteBuilder knitr

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Encoding UTF-8

Language en-US

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/MODIStsp

RemoteRef master

RemoteSha fdd94a41fb5461fe9a76648469a630791234e476

Contents

`MODIStsp-package` *MODIStsp: a package to automatize the creation of time series of raster images derived from MODIS Land Products*

Description

MODIStsp allows automating the creation of time series of rasters derived from MODIS Satellite Land Products data. It performs several typical preprocessing steps such as download, mosaicking, reprojection and resize of data acquired on a specified time period. All processing parameters can be set using a user-friendly GUI. Users can select which layers of the original MODIS HDF files they want to process, which additional Quality Indicators should be extracted from aggregated MODIS Quality Assurance layers and, in the case of Surface Reflectance products , which Spectral Indexes should be computed from the original reflectance bands. For each output layer, outputs are saved as single-band raster files corresponding to each available acquisition date. Virtual files allowing access to the entire time series as a single file are also created. Command-line execution exploiting a previously saved processing options file is also possible, allowing to automatically update time series related to a MODIS product whenever a new image is available.

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015-2017)

See Also

https://docs.ropensci.org/MODIStsp/

https://github.com/ropensci/MODIStsp

`bbox_from_file` *Retrieve bbox from a spatial file*

Description

Helper function used to retrieve the bounding box of a specified spatial file recognized by `sf` or `raster`: the function reads the extent using `sf::st_bbox()`

Usage

```
bbox_from_file(file_path, crs_out)
```

Arguments

file_path	character path of a spatial file.
crs_out	(crs | character) crs of the desired output projection, or string coercible to it using sf::st_crs() (e.g., WKT or numeric EPSG code)

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2017)

Luigi Ranghetti, phD (2017)

---

check_files_existence *Check if all files required for a given date already exist*

Description

Accessory function used to see if all expected out files for the selected date are already present in the output folder. If all expected out files are already present, check_files is set to TRUE, and the date is skipped in MODIStsp_process.

Usage

```
check_files_existence(
  out_prod_folder,
  file_prefix,
  yy,
  DOY,
  bandnames,
  bandsel_orig_choice,
  indexes_bandnames,
  indexes_bandsel,
  quality_bandnames,
  quality_bandsel,
  out_format
)
```

Arguments

out_prod_folder	character MODIStsp output folder
file_prefix	character File prefix of the processed product (e.g., MOD13Q1)
yy	character year
DOY	character doy

`bandnames` `character array` Bandnames of the MODIS product

`bandsel_orig_choice`
`numeric 0/1 array` Indicates which original MODIS layers were selected for processing (does not contain names of bands needed to compute SIs but not selected by the user!)

`indexes_bandnames`
`character array` Names of available spectral indexes (standard + custom) available for the currently processed product

`indexes_bandsel`
`numeric 0/1 array` Indicates which spectral indexes were selected for processing

`quality_bandnames`
`character array` Name of available Quality Indicators for the currently processed product

`quality_bandsel`
`numeric 0/1 array` Indicates which Quality Indicators were selected

`out_format` `character` GTiff or ENVI

Value

check - logical = 1 if all expected output files are already existing

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

`check_projection` *Check the validity of the input projection*

Description

helper function used to check that the input projection (passed as UTM zone, EPSG code, WKT string) is a valid projection for MODIStsp.

Usage

```
check_projection(projection, abort = FALSE, verbose = TRUE)

## Default S3 method:
check_projection(projection, abort = FALSE, verbose = TRUE)

## S3 method for class 'numeric'
```

```
check_projection(projection, abort = FALSE, verbose = TRUE)

## S3 method for class 'character'
check_projection(projection, abort = FALSE, verbose = TRUE)

## S3 method for class 'crs'
check_projection(projection, abort = FALSE, verbose = TRUE)
```

Arguments

projection	character or integer corresponding to the an EPSG code, a UTM zone (e.g. "32N") or a WKT representation of a projection;
abort	logical if TRUE, the function aborts in case an invalid invalid projection is passed. Otherwise, the function returns "NA", Default: TRUE
verbose	logical if TRUE, return messages

Value

character proj4string of the object or file

Note

This function was forked from package sprawl, version 0.3.0.

Author(s)

Lorenzo Busetto, phD (2017)

Luigi Ranghetti, phD (2017)

Examples

```
## Not run:
check_projection("32632")

check_projection("32631")

check_projection(32633)

check_projection(30, abort = FALSE)

check_projection("example of invalid string", abort = FALSE)

proj_wkt <- sf::st_as_text(sf::st_crs(32632))
check_projection(proj_wkt)

## End(Not run)
```

get_mod_dates *Find MODIS dates included in selected processing period*

Description

Accessory function to find the folders corresponding to the requested dates period within the full list retrieved by get_moddirs

Usage

```
get_mod_dates(dates, date_dirs)
```

Arguments

dates	2- element string array specifying start/end dates (yyyy.mm.dd) for which the http addresses of folders in lpdaac should be retrieved (e.g., c("2015.1.1", "2015.12.31)
date_dirs	data frame full list of folders in lpdaac archive for product of interest

Value

array of folder names containing data for the MODIS product acquired in the period specified by "dates"

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2016)

Lorenzo Busetto, phD (2017)

get_mod_dirs *Get list of MODIS data folders from http server*

Description

Accessory function to get the full list of directories on the lpdaac http site containing data included in the time range selected for processing (modified after Barry Rowlingson function):

Usage

```
get_mod_dirs(
  http,
  download_server,
  user,
  password,
  yy,
  n_retries,
  gui,
  out_folder_mod
)
```

Arguments

`http`	`character` http site on lpdaac corresponding to the selected MODIS product
`download_server`	`character` `["http" | "offline"]` download service to be used; if NA, the script tries to download with http.
`user`	`character` username for earthdata http server
`password`	`character` password for earthdata http server
`yy`	`character` Year for which the folder containing HDF images are to be identified
`n_retries`	`numeric` number of times the access to the http server should be retried in case of error before quitting, Default: 20
`gui`	'logical" indicates if processing was called from the GUI environment or not. If not, processing messages are sent to a log file instead than to the console/GTK progress windows.
`out_folder_mod`	`character` output folder for MODIS HDF storage

Value

`character array` listing all available folders (a.k.a. dates) for the requested MODIS product on lpdaac http archive, for the years included in the time range selected for processing.

Note

License: GPL 3.0

Author(s)

Original code by Babak Naimi (`.getModisList`, in ModisDownload.R) modified to adapt it to MODIStsp scheme and to http archive (instead than old FTP) by:

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2016-2017)

get_mod_filenames *Find the names of MODIS images corresponding to the selected dates*

Description

Accessory function to find the names of HDF images corresponding to a given date and interval of spatial tiles within the lpdaac archive.

Usage

```
get_mod_filenames(
  http,
  used_server,
  user,
  password,
  n_retries,
  date_dir,
  v,
  h,
  tiled,
  out_folder_mod,
  gui
)
```

Arguments

http	character url of http site on lpdaac corresponding to a given MODIS product.
used_server	character can assume values "http"; it cannot be NA.
user	character username for earthdata server.
password	character password for earthdata server.
n_retries	numeric number of times the access to the http server should be retried in case of error before quitting, Default: 20.
date_dir	character array array of folder names corresponding to acquisition containing dates where MODIS files to be downloaded are to be identified (return array from get_mod_dates).
v	integer array containing a sequence of the vertical tiles of interest (e.g., c(18,19)).
h	integer array containing a sequence of the horizontal tiles of interest (e.g., c(3,4)).
tiled	numeric [0/1] indicates if the product to be downloaded is tiled or not tiled. 1 = tiled product; 0 = non-tiled product (resolution 0.05 deg).
out_folder_mod	character folder where hdf files have to be stored.
gui	logical indicates if processing was called within the GUI environment or not. If not, processing messages are redirected direct to the log file.

Value

character array containing names of HDF images corresponding to the requested tiles available for the product in the selected date

Note

License: GPL 3.0

Author(s)

Original code by Babak Naimi (.getModisList, in ModisDownload.R) modified to adapt it to MODIStsp scheme and to http archive (instead than old FTP) by:

Lorenzo Busetto, phD (2014-2016)

Luigi Ranghetti, phD (2016)

---

get_reqbands *Identify the MODIS original bands needed for a given processing run*

Description

Helper function used in MODIStsp_process to identify which MODIS hdf layers are required for the current process. The required layers include all MODIS original layers selected by the user, plus all those required to compute the Spectral Indexes and Quality Indicators selected by the user

Usage

```
get_reqbands(
  bands_indexes_matrix,
  indexes_bandsel,
  indexes_bandnames,
  quality_bandsel,
  quality_bandnames,
  out_prod_folder,
  file_prefix,
  yy,
  DOY,
  out_format,
  reprocess
)
```

Arguments

bands_indexes_matrix
: matrix built by set_bandind_matrix

indexes_bandsel
: character array Spectral Indexes to be computed starting from reflectance bands. You can get a list of available quality layers for a given product using function MODIStsp_get_prodlayers (e.g., MODIStsp_get_prodlayers("M*D13Q1")$indexes_bandnam
Default: NULL

indexes_bandnames
: names of all indexes available for the product being processed

quality_bandsel
: character array Quality Indicators to be computed starting from bit fields of original MODIS layers. You can get a list of available quality layers for a given product using function MODIStsp_get_prodlayers (e.g., MODIStsp_get_prodlayers("M*D13Q1")$qual
Default: NULL

quality_bandnames
: names of all quality indicators available for the product being processed

out_prod_folder
: character Main folder where the MODIStsp processed raster will be stored. Used to check if a given processed image already exists.

file_prefix
: File prefix corresponding to the MODIS product being processed. Used to check if a given processed image already exists.

yy
: Year corresponding to the image being processed. Used to check if a given processed image already exists.

DOY
: DOY corresponding to the image being processed. Used to check if a given processed image already exists.. Used to check if a given processed image already exists.

out_format
: character ["ENVI" | "GTiff"] Desired output format.

reprocess
: logical If TRUE, reprocess data for already existing dates.

Value

req_bands_indexes

Author(s)

Lorenzo Busetto, phD (2017)

---

get_yeardates *identify dates to be processed for a year*

Description

helper function needed to identify the ranges of dates to be processed for a given year as a function of download_range selection and starting/ending dates and years

Usage

```
get_yeardates(download_range, yy, start_year, end_year, start_date, end_date)
```

Arguments

download_range	character ["Full" | "Seasonal"] If "full", all the available images between the starting and the ending dates are downloaded; If "seasonal", only the images included in the season are downloaded (e.g: if the starting date is 2005-12-01 and the ending is 2010-02-31, only the images of December, January and February from 2005 to 2010 - excluding 2005-01, 2005-02 and 2010-12 - are downloaded), Default: Full
yy	numeric year for which the processing dates need to be identified
start_year	numeric start year of current MODIStsp_process run
end_year	numeric end year of current MODIStsp_process run.
start_date	character`` Start date for images download and preprocessing (yyyy.mm.dd) of current MO run.
end_date	character Start date for images download and preprocessing (yyyy.mm.dd) of current MODIStsp_process run.

Value

OUTPUT_DESCRIPTION

Author(s)

Lorenzo Busetto, phD (2017)

---

install_MODIStsp_launcher

Install a launcher for MODIStsp

Description

Function which allows to use MODIStsp in batch mode by creating links

Usage

```
install_MODIStsp_launcher(
  bin_dir = NA,
  rscript_dir = NA,
  desktop_dir = NA,
  desktop_shortcut = TRUE,
  sudo = FALSE
)
```

Arguments

bin_dir	• on Linux, directory in which the link to the bash script should be placed, Default: "/usr/bin" - use of a path included in the PATH environment variable is suggested; • on Windows, directory where to place the menu entry in the Start Menu, Default: Start Menu -> Programs -> MODIStsp.
rscript_dir	character in Windows only, the path of the directory in which Rscript is installed (usually is \"C:/Progra~1/R/R-version/bin/x64"). Edit this parameter if R is installed in a custom directory.
desktop_dir	character • on Linux, directory in which the desktop entry should be placed, Default: /usr/share/applications; • on Windows, directory where to place the desktop entry, Default: "Desktop" (Ignored if desktop_shortcut = FALSE).
desktop_shortcut	logical indicates if the desktop entry or the desktop shortcut should be created, Default: TRUE.
sudo	(Linux only) logical indicates if administrator rights have to be used to write within bin_dir and desktop_dir, If FALSE the root password is requested when launching the function. Note that using default values of bin_dir and desktop_dir requires to set this option to TRUE (or to launch the script in a root session of R), Default: FALSE

Details

MODIStsp can be used also as a stand-alone tool (i.e., without opening RStudio or R-GUI) by launching a bash/batch script, which is stored in the installation folder (/ExtData/Launcher) To allow to easily find it, this function creates a desktop entry and a symbolic link to the bash script (on Linux) or a link in the Start Menu to the batch script and a shortcut on the desktop (on Windows). **Note that**, if the packages MODIStsp is installed in a version-dependent directory (as the default one is), this function should be re-executed after an R upgrade, otherwise the links would continue to point to the old package version!

Value

The function is called for its side effects.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2015)

Examples

```
# Linux: common installation (script in /usr/bin,
# desktop entry in /usr/share/applications)
# (requires administrator permissions)
## Not run:
# the administrator password is asked interactively
install_MODIStsp_launcher(sudo = TRUE)

## End(Not run)

# Linux: installation in a directory which does not require administrator
# permissions
## Not run:
install_MODIStsp_launcher(bin_dir = "~/bin", desktop_dir = "~/Desktop")

## End(Not run)

# Windows: common installation
# (script in the Start Menu and shortcut on the desktop)
## Not run:
install_MODIStsp_launcher()

## End(Not run)
```

---

load_prodopts *Load characteristics of the different MODIS products*

Description

FUNCTION_DESCRIPTION

Usage

```
load_prodopts()
```

Details

Load characteristics of the different MODIS products from `prodopts_file`

Value

OUTPUT_DESCRIPTION

Author(s)

Lorenzo Busetto, phD (2017)

MODIStsp *MODIStsp main function*

Description

Main function for the MODIS Time Series Processing Tool (MODIStsp)

Usage

```
MODIStsp(
  ...,
  gui = TRUE,
  out_folder = NULL,
  out_folder_mod = NULL,
  opts_file = NULL,
  selprod = NULL,
  prod_version = NULL,
  bandsel = NULL,
  quality_bandsel = NULL,
  indexes_bandsel = NULL,
  sensor = NULL,
  download_server = NULL,
  downloader = NULL,
  user = NULL,
  password = NULL,
  download_range = NULL,
  start_date = NULL,
  end_date = NULL,
  spatmeth = NULL,
  start_x = NULL,
  end_x = NULL,
  start_y = NULL,
  end_y = NULL,
  bbox = NULL,
  spafile = NULL,
  out_projsel = NULL,
  output_proj = NULL,
  out_res_sel = NULL,
  out_res = NULL,
  resampling = NULL,
  reprocess = NULL,
  delete_hdf = NULL,
  nodata_change = NULL,
  scale_val = NULL,
  ts_format = NULL,
  out_format = NULL,
  compress = NULL,
```

```
  test = NULL,
  n_retries = 5,
  verbose = TRUE,
  parallel = TRUE
)
```

Arguments

...	not used for values, forces later arguments to bind by name
gui	logical if TRUE: the GUI is opened before processing. If FALSE: processing parameters are retrieved from the provided opts_file argument), Default: TRUE
out_folder	character Main output folder, default: NULL.
out_folder_mod	character Output folder for original HDF storage. If "$tempdir" (default), a temporary directory is used.
opts_file	character full path to a JSON file containing MODISTsp processing options saved from the GUI, Default: NULL
selprod	character Name of selected MODIS product (e.g., Vegetation Indexes_16Days_250m (M*D13Q1)). You can get a list of available product names using function MODISTsp_get_prodnames, Default: NULL
prod_version	Version of the selected MODIS product. Currently versions "061" and/or "006" can be chosen. Default value is "061" (version "006" was decommissioned by USGS on July 31, 2023, products of this version are being gradually removed). In case users would encounter an error in the encoding of bands or quality flags they are encouraged to report it by opening a new issue on GitHub at https://github.com/ropensci/MODISTsp/issues.
bandsel	character array Original MODIS layers to be processed. You can get a list of available layers for a given product using function MODISTsp_get_prodlayers (e.g., MODISTsp_get_prodlayers("M*D13Q1")$bandnames), Default: NULL
quality_bandsel	character array Quality Indicators to be computed starting from bit fields of original MODIS layers. You can get a list of available quality layers for a given product using function MODISTsp_get_prodlayers (e.g., MODISTsp_get_prodlayers("M*D13Q1")$qual Default: NULL
indexes_bandsel	character arraySpectral Indexes to be computed starting from reflectance bands. You can get a list of available quality layers for a given product using function MODISTsp_get_prodlayers (e.g., MODISTsp_get_prodlayers("M*D13Q1")$indexes_bandnam Default: NULL
sensor	character ["Terra"| "Aqua" | "Both"] MODIS platform to be considered. (Ignored for MCD* products). Default: "Both"
download_server	character ["http" | "offline"] service to be used for download. Default: "http"
downloader	download_server character ["http" | "aria2"] downloader to be used, Default: "http"

user	character Username for NASA http server. (urs.earthdata.nasa.gov/home).
password	character Password for NASA http server (urs.earthdata.nasa.gov/home).
download_range	character ["Full" | "Seasonal"] If "full", all the available images between the starting and the ending dates are downloaded; If "seasonal", only the images included in the season are downloaded (e.g: if the starting date is 2005-12-01 and the ending is 2010-02-31, only the images of December, January and February from 2005 to 2010 - excluding 2005-01, 2005-02 and 2010-12 - are downloaded), Default: Full
start_date	character Start date for images download and preprocessing (yyyy.mm.dd), Default: NULL
end_date	character End date for images download and preprocessing (yyyy.mm.dd), Default: NULL
spatmeth	character ["tiles" | "bbox" | "file"], indicates how the processing extent is retrieves. if "tiles", use the specified tiles (start_x....). If "file", retrieve extent from spatial file specifies in spafile. If "bbox", use the specified bounding box, Default: "tiles"
start_x	integer [0-35] Start MODIS horizontal tile defining spatial extent. Ignored if spatmeth != "tiles", Default: 18
end_x	integer [0-35] End MODIS horizontal tile defining spatial extent. Ignored if spatmeth != "tiles", Default: 18
start_y	integer [0-17] Start MODIS vertical tile defining spatial extent. Ignored if spatmeth != "tiles", Default: 4
end_y	integer [0-17] End MODIS vertical tile defining spatial extent. Ignored if spatmeth != "tiles", Default: 4
bbox	numeric(4) Output bounding box (xmin, ymin, xmax, ymax) in out_proj coordinate system. Ignored if spatmeth == "tiles", Default: NULL
spafile	character (optional) full path of a spatial file to use to derive the processing extent. If not NULL, the processing options which define the extent, the selected tiles and the "Full Tile / Custom" in the JSON options file are overwritten and new files are created on the extent of the provided spatial file. Ignored if spatmeth != "file", Default: NULL
out_projsel	character ["Native", "User Defined] If "Native", the outputs keep the original resolution of MODIS HDF images. Otherwise, the value set in "out_res" is used, Default:Native
output_proj	character either equal to "MODIS Sinusoidal", or to the code of a valid EPSG or to a WKT projection string. Ignored if outproj_sel == "Native", Default: NULL
out_res_sel	character ["Native", "User Defined]. If "Native", the outputs keep the original resolution of MODIS HDF images. Otherwise, the value set in "out_res" is used.
out_res	float Output resolution (in output projection measurement unit). Ignored if out_res_sel == "Native".
resampling	character ["near" | "bilinear" | "cubic" | "cubicspline", |lanczos"|, "average"|, "mode Resampling method to be used by gdalwarp.

`reprocess`	`logical` If TRUE, reprocess data for already existing dates.
`delete_hdf`	`logical` If TRUE, delete downloaded HDF files after completion.
`nodata_change`	`logical` if TRUE, NoData values are set to the max value of the datatype of the layer on the MODIStsp output rasters. NOTE: If multiple nodata values are reported for a layer, all are reset to the new value.
`scale_val`	`logical` If TRUE, scale and offset are applied to original MODIS layers, and Spectral Indexes are saved as floating point. If FALSE, no rescaling is done and Spectral Indexes are saved as integer, with a 10000 scaling factor.
`ts_format`	`character array including ["R RasterStack" | "ENVI Meta Files" | "GDAL VRT" | "ENVI and G` Selected virtual time series format.
`out_format`	`character ["ENVI" | "GTiff"]` Desired output format.
`compress`	`character ["None" | "PACKBITS" | "LZW" | "DEFLATE"]` Compression method for GTiff outputs (Ignored if `out_format == ENVI`)
`test`	`integer | character (e.g., "01a")` if set, MODIStsp is executed in "test mode", using a preset Options File instead than opening the GUI or accepting the `opts_file` parameter. This allows both to check correct installation on user's machines, and to implement unit testing.
`n_retries`	`numeric` maximum number of retries on download functions. In case any download function fails more than `n_retries` times consecutively, MODIStsp_process will abort, Default: 20
`verbose`	`logical` If FALSE, suppress processing messages, Default: TRUE
`parallel`	`logical` If TRUE (default), the function is run using parallel processing, to speed-up the computation for large rasters (with a maximum of 8 cores). The number of cores is automatically determined; specifying it is also possible (e.g. `parallel = 4`). In this case, more than 8 cores can be specified. If FALSE (default), single core processing is used.

Details

The function is used to:

- initialize the processing (folder names, packages, etc.);
- launch the GUI (`MODIStsp_GUI()`) on interactive execution, or load an options file to set processing arguments and/or retrieve CLI inputs and run processing on non-interactive execution;
- launch the routines for downloading and processing the requested datasets. (`MODIStsp_process()`)
- launching the function with GUI = FALSE and without specifying a opts_file initializes arguments with default values. This allows making a test run.

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015-2017)

See Also

MODIStsp_GUI(), MODIStsp_process()

Examples

```
#' # - Running the tool using the GUI
# Running the tool without any option will start the GUI with the default or
# last used settings, in interactive mode (i.e., with gui = TRUE).
if (interactive()) {
  MODIStsp()
}


#' # - Running the tool specifying processing arguments in the call

# **NOTE** Output files of examples are saved to file.path(tempdir(), "MODIStsp").

# Here we process layers __NDVI__ and __EVI__ and quality indicator __usefulness__
# of product __M*D13Q1__, considering both Terra and Aqua platforms, for dates
# comprised between 2020-06-01 and 2020-06-15 and saves output to R tempdir
# --> See name and available layers for product M*D13Q1.
# Note that this example (as well as the following ones) is run in single
# core to follow CRAN policies, by setting parallel = FALSE.
# Users can exploit multicore functionalities skipping to set this argument.

# The following check is performed in order not to provide errors
# running the examples if HDF4 is not supported.
is_hdf4_supported <- "HDF4" %in% sf::st_drivers("raster")$name

MODIStsp_get_prodlayers("M*D13A2")
if (is_hdf4_supported) {
  MODIStsp(
    gui = FALSE,
    out_folder = "$tempdir",
    selprod = "Vegetation_Indexes_16Days_1Km (M*D13A2)",
    bandsel = c("EVI", "NDVI"),
    quality_bandsel = "QA_usef",
    indexes_bandsel = "SR",
    user = "mstp_test" ,
    password = "MSTP_test_01",
    start_date = "2020.06.01",
    end_date = "2020.06.15",
    verbose = FALSE,
    parallel = FALSE
  )
}


#' # - Running the tool using the settings previously saved in a specific options file

# **NOTE** Output files of examples are saved to file.path(tempdir(), "MODIStsp").
```

```
# You can run the examples with `gui = TRUE` to set a different output folder!

# Here we use a test json file saved in MODIStsp installation folder which
# downloads and processed 3 MOD13A2 images over the Como Lake (Lombardy, Italy)
# and retrieves NDVI and EVI data, plus the Usefulness Index Quality Indicator.

opts_file <- system.file("testdata/test_MOD13A2.json", package = "MODIStsp")

if (is_hdf4_supported) {
  MODIStsp(gui = FALSE, opts_file = opts_file, verbose = TRUE, parallel = FALSE)
}


# Running the tool using the settings previously saved in a specific option file
# and specifying the extent from a spatial file allows to re-use the same
# processing settings to perform download and reprocessing on a different area

opts_file <- system.file("testdata/test_MOD13A2.json", package = "MODIStsp")
spatial_file <- system.file("testdata/lakeshapes/garda_lake.shp", package = "MODIStsp")
if (is_hdf4_supported) {
  MODIStsp(
    gui = FALSE,
    opts_file = opts_file,
    spatmeth = "file",
    spafile = spatial_file,
    verbose = TRUE,
    parallel = FALSE
  )
}


# Running the tool using the settings previously saved in a
# specific options file and specifying each time the extent from a different
# spatial file (e.g., to perform the same processing on several extents)
# Note that you can also put all your extent files in a specific folder and
# create the extent list using for example.

extent_list = list.files(
  system.file("testdata/lakeshapes/", package = "MODIStsp"),
  "\\.shp$",
  full.names = TRUE
)
extent_list
opts_file <- system.file("testdata/test_MOD13A2.json", package = "MODIStsp")

if (is_hdf4_supported) {
  for (single_shape in extent_list) {
    MODIStsp(
      gui = FALSE,
      opts_file = opts_file,
      spatmeth = "file",
      spafile = single_shape,
      verbose = TRUE,
```

```
      parallel = FALSE
    )
  }
}

# output files are placed in separate folders:
outfiles_garda <- list.files(
  file.path(tempdir(), "MODIStsp/garda_lake/VI_16Days_1Km_v61/NDVI"),
  full.names = TRUE
)
outfiles_garda
require(raster)
if (length(outfiles_garda) > 0) {
  plot(raster(outfiles_garda[1] ))
}

outfiles_iseo <- list.files(
  file.path(tempdir(), "MODIStsp/iseo_lake/VI_16Days_1Km_v61/NDVI"),
  full.names = TRUE
)
outfiles_iseo
if (length(outfiles_garda) > 0) {
  plot(raster(outfiles_iseo[1]))
}

# See also https://docs.ropensci.org/MODIStsp/articles/noninteractive_execution.html
```

`MODIStsp_addindex` *Add custom spectral indexes*

Description

Function used to add a user-defined Spectral Index to the default list of computable spectral indexes. Execution without the GUI (i.e., to add a new index from a script) is also possible (see examples).

Usage

```
MODIStsp_addindex(
  new_indexbandname = "",
  new_indexfullname = "",
  new_indexformula = "",
  new_indexnodata_out = "32767"
)
```

Arguments

`new_indexbandname`

`character` short name (acronym) of the new spectral index (Ignored if gui == TRUE), Default: NULL

`new_indexfullname`
: `character` extended name (acronym) of the new spectral index (Ignored if gui == TRUE), Default: NULL

`new_indexformula`
: `character` string containing the formula of the new spectral indexes (Ignored if gui == TRUE). Variables allowed in the formula are the names of the bands: b1_Red, b2_NIR, b3_Blue, b4_Green, b5_SWIR, b6_SWIR and b7_SWIR. Default: NULL

`new_indexnodata_out`
: `character` nodata value to use for rasters containing the new index

Details

- The function asks the user to provide the info related to the new desired Spectral Index, checks for correctness of provided information (e.g., correct bandnames, computable formula, etc...). If the index is legit, it modifies the MODIStsp_addindex.json file so to allow computation of the additional index within MODIStsp for all products containing the required reflectance bands.
- To remove all custom-added spectral indexes, run MODIStsp_resetindexes()

Value

The function is called for its side effects. On success, the MODIStsp_indexes.json is modified so to allow computation of the additional indexes.

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

See Also

MODIStsp_resetindexes

Examples

```
# Run the GUI to interactively define a new index
 ## Not run:
 MODIStsp_addindex()
## End(Not run)

# Define the new index in non-interactive execution

## Not run:
MODIStsp_addindex(new_indexbandname = "SSI",
  new_indexfullname = "Simple Useless Index",
```

```
  new_indexformula = "b2_NIR+b1_Red")

## End(Not run)
```

`MODIStsp_download` *MODIStsp download function*

Description

Internal function dealing with download of MODIS hdfs from http remote server for a given date.

Usage

```
MODIStsp_download(
  modislist,
  out_folder_mod,
  download_server,
  http,
  n_retries,
  use_aria,
  date_dir,
  year,
  DOY,
  user,
  password,
  sens_sel,
  date_name,
  gui,
  verbose
)
```

Arguments

`modislist`	`character array` List of MODIS images to be downloaded for the selected date (as returned from `get_mod_filenames`). Can be a single image, or a list of images in case different tiles are needed!
`out_folder_mod`	`character` Folder where the hdfs are to be stored
`download_server`	`character ["http"]` Server to be used.
`http`	`character` Address of the http server for the selected product.
`n_retries`	`numeric` Max number of retry attempts on download. If download fails more that n_retries times consecutively, abort
`use_aria`	`logical` if TRUE, download using aria2c
`date_dir`	`character array` Sub-folder where the different images can be found (element of the list returned from `get_mod_dirs`, used in case of http download to generate the download addresses).

year	character Acquisition year of the images to be downloaded
DOY	character array Acquisition doys of the images to be downloaded
user	character Username for http download
password	character Password for http download
sens_sel	character ["terra" | "aqua"] Selected sensor.
date_name	character Date of acquisition of the images to be downloaded.
gui	logical Indicates if on an interactive or non-interactive execution (only influences where the log messages are sent).
verbose	logical If FALSE, suppress processing messages, Default: TRUE

Value

The function is called for its side effects

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

`MODIStsp_extract` *Extract data from MODIStsp time series*

Description

function used to extract time series data from rts files created by MODIStsp on spatial locations provided in the form of "R" spatial objects (SpatialPoints, SpatialPolygons, etc.)

Usage

```
MODIStsp_extract(
  in_rts,
  sf_object,
  start_date = NULL,
  end_date = NULL,
  id_field = NULL,
  FUN = "mean",
  out_format = "xts",
  small = TRUE,
  small_method = "centroids",
  na.rm = TRUE,
  verbose = FALSE
)
```

Arguments

in_rts	A RasterStack bject created by MODIStsp (it MUST contain acquisition dates in the "Z" attribute)
sf_object	"sf" object OR name of an GDAL-readable vector file specifying the "area" from which data has to be extracted. • If sf_object represents lines, the output object contains one column for each line, containing values obtained applying the function specified as the FUN argument over all pixels touched by the line, and one line for each date. • If sf_object represents points, the output object contains one column for each point, containing values of the cells corresponding to the point, and one line for each date. • If sf_object represents polygons, the output object contains one column for each polygon, containing values obtained applying the function specified as the FUN argument over all pixels belonging to the polygon, and one line for each date
start_date	object of class Date, POSIXct or POSIXlt OR character coercible to Date class (format = "yyyy-mm-dd")Starting date of the period to be considered for data extraction . If not provided, the first date of the RasterStack is used.
end_date	object of class Date, POSIXct or POSIXlt OR character coercible to Date class (format = "yyyy-mm-dd"). Ending date of the period to be considered for data extraction . If not provided, the last date of the RasterStack is used.
id_field	character name of the column of the input sp object or shapefile to be used in the data extraction. Values contained in the column MUST be unique. The names of the columns of the output are taken from this column. If not provided, or an invalid value is provided, then the names of the columns of the output reflect the number of the feature in sf_object.
FUN	function to summarize the values (e.g. mean) on polygon data frames. The function should take a single numeric vector as argument and return a single value (e.g. mean, min or max), and accept a na.rm argument. Thus, standard R functions not including an na.rm argument must be wrapped as in this example: fun=function(x,...)length(x). Defaults to "mean"
out_format	character ["xts" | "dframe"] If dframe, the output is a data frame with dates in the first column and extracted data in the others, otherwise it is a xts object, Default: "xts"
small	logical If TRUE, and input is polygons, then values are returned also for polygons not covering at least one raster cell. "Included" cells in this case depend on the values of the "small_method" parameter.
small_method	character ["centroids" | "full"] If small == TRUE and input is polygons, controls which cells are "extracted" for small polygons. If set to "centroids" (default), then only the cells corresponding to polygon centroid are considered (faster, may have problems on strangely shaped polygons). If set to "full", then all cells intersected by the small polygon are extracted and used in calculations, Default: "centroids"

na.rm	logical If TRUE, and sf_object is a polygon, then na.rm = TRUE is used when applying FUN to the different pixels of the polygon, Default = TRUE.
verbose	logical If TRUE, messages on processing status are sent to the console. Default = TRUE.

Details

The function takes as input a RasterStack object containing time information in the "z" attribute (set by `raster::setZ`), a starting and ending date and a standard "R" spatial object, and returns the time series for the spatial locations specified in the spatial object in the form of a "R" xts object OR a plain data.frame with a "date" column in first position. If the input spatial object is a "point" or "line" one, the output object contains one column for each specified point, or for each cell intersecting the line, and one line for each date. If the input spatial object is a "polygon" one, the output object contains one column for each polygon, containing values obtained applying the function specified as the FUN argument over all pixels belonging to the polygon, and one line for each date.

Value

data.frame or xts object. Each column of data corresponds to one point or one polygon, each row to a date.

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2015 - 2017) email: busetto.l@irea.cnr.it

Examples

```
## Not run:
# Extract average and standard deviation values from a rts object created by
# MODIStsp for each polygon of a shapefile, for each date in the period
# between 2001-01-01 and 2014-12-31

# The example uses tif files in testdata/VI_16Days_500m_v61 to build
# a MODIStsp rasterStack corresponding to the 2016 time series of the NDVI index
# over the Como Lake (Italy). It then extracts data on polygons corresponding
# to different land cover classes saved in testdata/extract_polys.shp

# First, prepare the test dataset.
# __NOTE__ To avoid redownloading, here we copy some test data from MODIStsp
# installation folder to tempdir and use it to create a test time series.

test_zip <-  system.file("testdata/VI_16Days_500m_v6/NDVI.zip",
                         package = "MODIStsp")
dir.create(file.path(tempdir(), "MODIStsp/VI_16Days_500m_v61"),
           showWarnings = FALSE, recursive = TRUE)
utils::unzip(test_zip,
             exdir = file.path(tempdir(), "MODIStsp/VI_16Days_500m_v61"))
```

```
opts_file <- system.file("testdata/test_extract.json", package = "MODIStsp")
MODIStsp(opts_file = opts_file, gui = FALSE, verbose = FALSE)

# Now load the MODIStsp stack: This is a MODIS NDVI time series ranging between
# 2016-01-01 and 2016-12-18
# __NOTE__: MODIStsp rasterStack files are always saved in the "Time_Series\/RData"
# subfolder of your main output folder - see
# "https://docs.ropensci.org/MODIStsp/articles/output.html")

# Specify the filename of the RData RasterStack of interest
stack_file  <- file.path(tempdir(),
 "MODIStsp/VI_16Days_500m_v61/Time_Series/RData/Terra/NDVI",
  "MOD13A1_NDVI_1_2016_353_2016_RData.RData")
basename(stack_file)

ts_data <- get(load(stack_file))
ts_data

# Now load a shapefile containing polygons from which we want to extract data

polygons <- sf::st_read(system.file("testdata/extract_polys.shp",
                            package = "MODIStsp"), quiet = TRUE)
polygons

# Finally, extract the average values for each polygon and date and plot the
# results

out_dataavg <- suppressMessages(MODIStsp_extract(ts_data, polygons, id_field = "lc_type",
                              small = FALSE))
head(out_dataavg)

plot(out_dataavg, legend.loc = "topleft")

# use a different summarization function

out_datasd <- MODIStsp_extract(ts_data, polygons, id_field = "lc_type",
                               FUN = "sd", small = FALSE)
head(out_datasd)

# (See also https://docs.ropensci.org/MODIStsp/articles/Analyze.html for a
# worked-out example)

## End(Not run)
```

MODIStsp_get_prodlayers

Retrieve the names of MODIS layers for a product

Description

Function used to retrieve the names of original MODIS layers, quality layers and eventually available spectral indexes given a MODIS product code. It is useful to identify the names of the layers to be processed when launching MODIStsp from the CLI.

Usage

```
MODIStsp_get_prodlayers(prodname, prodver = "061")
```

Arguments

prodname	character containing the code of the desired MODIS product. NOTE: for products available separately for Terra and Aqua (e.g., MOD13Q1/MYD13Q1), use the code M*D_code_ (e.g., M*D13Q1)
prodver	character containing the version of the desired MODIS product. Currently versions "061" (default) and "006" can be chosen.

Value

list, containing the slots: prodname, bandnames, quality_bandnames and indexes_bandnames, band_fullnames, quality_fullnames, indexes_fullnames

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2020)

Luigi Ranghetti, phD (2021)

Examples

```
# Get layers of product M*13Q1 based on code
MODIStsp_get_prodlayers("M*13Q1")

# Get layers of product M*13Q1 based on full name
MODIStsp_get_prodlayers("Vegetation Indexes_16Days_250m (M*D13Q1)")

# Get indexes names of product M*13Q1 based on full name
MODIStsp_get_prodlayers("MCD43C4")$indexes_bandnames
```

MODIStsp_get_prodnames *Retrieve the names of all available product*

Description

Function used to retrieve the names of available MODIS products

Usage

```
MODIStsp_get_prodnames()
```

Value

character array of product names

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2020)

Examples

```
# Get MODIStsp product names
MODIStsp_get_prodnames()
```

MODIStsp_GUI *Build and manage the MODIStsp GUI*

Description

Function used to generate and handle the GUI used to allow selection of MODIStsp processing parameters.

Usage

```
MODIStsp_GUI()
```

Value

the function is called for its side effects - opening the GUI and allowing to set, save, load options and eventually launch the processing.

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

`MODIStsp_process` *MODIStsp main processing function*

Description

Main processing function of MODIStsp. Takes as input processing parameters specified by the user and performs all required processing.

Usage

```
MODIStsp_process(proc_opts, n_retries, verbose = TRUE, parallel = TRUE)
```

Arguments

`proc_opts`	`data.frame` containing all processing parameters, as passed from the MODIStsp GUI, or created in `MODIStsp` by joining explicitly passed arguments with a (not mandatory) options file.
`n_retries`	`numeric` maximum number of retries on download functions. In case any download function fails more than `n_retries` times consecutively, MODIStsp_process will abort, Default: 20
`verbose`	`logical` If FALSE, suppress processing messages, Default: TRUE
`parallel`	`logical` If TRUE (default), the function is run using parallel processing, to speed-up the computation for large rasters (with a maximum of 8 cores). The number of cores is automatically determined; specifying it is also possible (e.g. `parallel = 4`). In this case, more than 8 cores can be specified. If FALSE (default), single core processing is used.

Details

After retrieving the input processing options, the function

1. Accesses NASA http archive to determine the list of files to be downloaded/processed (or, in case of offline processing, get the list of already available hdf files present in `out_mod_folder`);
2. Performs all required processing steps on each date (download, reprojection, resize, mosaicing, Spectral Indexes and Quality indicators computation);
3. Creates virtual files of the processed time series.

Reprojection and resize is dealt with by accessing gdal routines through the gdalUtilities package. Extraction of bitfields from Quality layers is done though bitwise computation Checks are done in order to not re-download already existing HDF images, and not reprocess already processed dates (if the user did not specify that)

Value

The function is called for its side effects.

Note

Thanks Tomislav Hengl and Babak Naimi, whose scripts made the starting point for development of this function (ModisDownload; Download_and_resampling_of_MODIS_images)

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

MODIStsp_process_bands

MODIStsp helper for processing original HDF layers

Description

Internal function used to perform the required spatial processing on MODIS original hdf layers (reprojection, resizing, resampling, mosaicing, computation of scaling factors). The function is based on the use of gdal routines.

Usage

```
MODIStsp_process_bands(
  out_folder_mod,
  modislist,
  outproj_str,
  mod_proj_str,
  sens_sel,
  band,
  bandname,
  date_name,
  datatype,
  nodata_in,
  nodata_out,
  full_ext,
  bbox,
  scale_val,
```

```
  scale_factor,
  offset,
  out_format,
  outrep_file,
  compress,
  out_res_sel,
  out_res,
  resampling,
  nodata_change,
  gui,
  verbose,
  parallel
)
```

Arguments

out_folder_mod	character Output folder for original HDF storage. If "$tempdir" (default), a temporary directory is used.
modislist	character array List of MODIS images to be downloaded for the selected date (as returned from get_mod_filenames). Can be a single image, or a list of images in case different tiles are needed!
outproj_str	character EPSG or WKT of output projection.
mod_proj_str	character EPSG or WKT of MODIS projection.
sens_sel	character ["terra" | "aqua"] Selected sensor.
band	numeric band number corresponding to the HDF layer to be processed
bandname	character Name of the HDF layer to be processed.
date_name	character Date of acquisition of the images to be downloaded.
datatype	character Datatype to the HDF layer to be processed.
nodata_in	numeric Original nodata value to the HDF layer to be processed.
nodata_out	numeric Output nodata value to the HDF layer to be processed.
full_ext	logical If TRUE, process full tiles, if FALSE, process bbox
bbox	numeric(4) Output bounding box (xmin, ymin, xmax, ymax) in out_proj coordinate system. Ignored if spatmeth == "tiles", Default: NULL
scale_val	logical If TRUE, scale and offset are applied to original MODIS layers, and Spectral Indexes are saved as floating point. If FALSE, no rescaling is done and Spectral Indexes are saved as integer, with a 10000 scaling factor.
scale_factor	numeric Scale factor to be applied to the HDF layer to be processed (Ignored if scale_val == FALSE).
offset	numeric Offset to be applied to the HDF layer to be processed (Ignored if scale_val == FALSE).
out_format	character ["ENVI" | "GTiff"] Desired output format.
outrep_file	character Full path of the file where results of the processing are to be stored (created in MODIStsp_process)

compress	character ["None" | "PACKBITS" | "LZW" | "DEFLATE"] Compression method for GTiff outputs (Ignored if out_format == ENVI)
out_res_sel	character ["Native", "User Defined]. If "Native", the outputs keep the original resolution of MODIS HDF images. Otherwise, the value set in "out_res" is used.
out_res	float Output resolution (in output projection measurement unit). Ignored if out_res_sel == "Native".
resampling	character ["near" | "bilinear" | "cubic" | "cubicspline", |lanczos"|, "average"|, "mode Resampling method to be used by gdalwarp.
nodata_change	logical if TRUE, NoData values are set to the max value of the datatype of the layer on the MODIStsp output rasters. NOTE: If multiple nodata values are reported for a layer, all are reset to the new value.
gui	logical if TRUE: the GUI is opened before processing. If FALSE: processing parameters are retrieved from the provided opts_file argument), Default: TRUE
verbose	logical If FALSE, suppress processing messages, Default: TRUE
parallel	logical If TRUE, the function is run using parallel processing, to speed-up the computation for large rasters (with a maximum of 8 cores). The number of cores is automatically determined; specifying it is also possible (e.g. parallel = 4). In this case, more than 8 cores can be specified. If FALSE (default), single core processing is used.

Value

The function is called for its side effects

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

MODIStsp_process_indexes

MODIStsp helper for computing spectral indexes

Description

function used to compute spectral indexes, given the index formula

Usage

```
MODIStsp_process_indexes(
  out_filename,
  out_prod_folder,
  formula,
```

```
  bandnames,
  nodata_out,
  indexes_nodata_out,
  file_prefix,
  compress,
  yy,
  out_format,
  DOY,
  scale_val
)
```

Arguments

out_filename	character basename of the file in to which save results
out_prod_folder	character output folder for the product used to retrieve filenames of rasters of original bands to be used in computations
formula	character Index formula, as derived from XML file and stored in prod_opts within previous_file
bandnames	character array of names of original HDF layer. Used to identify the bands required for index computation
nodata_out	character array of NoData values of reflectance bands
indexes_nodata_out	character NoData value for resulting raster
file_prefix	character used to retrieve filenames of rasters of original bands to be used in computations
compress	character compression option for GTiff files
yy	character year string used to retrieve filenames of rasters of original bands to be used in computations
out_format	character string used to retrieve filenames of rasters of original bands to be used in computations
DOY	character doy string used to retrieve filenames of rasters of original bands to be used in computations
scale_val	character (Yes/No) if Yes, output values in are computed as float -1 - 1, otherwise integer -10000 - 10000

Details

the function parses the index formula to identify the required bands. On the basis of identified bands, it retrieves the reflectance bands required, gets the data into R raster objects, performs the computation and stores results in a GeoTiff or ENVI raster file

Value

NULL - new raster file saved in out_filename

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2017)

Luigi Ranghetti, phD (2017)

`MODIStsp_process_QA_bits` *MODIStsp helper function to compute Quality Indicators from HDF bit-field layers*

Description

function used to extract quality indicator from MODIS aggregated quality layers

Usage

```
MODIStsp_process_QA_bits(
  out_filename,
  in_source_file,
  bitN,
  out_format,
  nodata_source,
  nodata_qa_in,
  nodata_qa_out,
  compress
)
```

Arguments

`out_filename`	`character` file name of the output raster files containing QI values
`in_source_file`	`character` name of the file created by MODIStsp containing the data required to compute the quality indicator
`bitN`	`character` position of the bits corresponding to the quality indicator of interest (e.g., 0-1 = first two bits; 2-5: bits from 2 to 5, etc.)
`out_format`	output format (ENVI or GTiff)
`nodata_source`	`character` NoData values of the MODIS band containing data from which the bit field corresponding to the quality indicator must be extracted
`nodata_qa_in`	`character` in NoData for quality bands ("255")
`nodata_qa_out`	`character` out NoData for quality bands ("255")
`compress`	`character` compression option for GTiff files

Details

On the basis of the name of the image containing the aggregated quality information (`in_source_file``) and of the posit
the function extracts the correct information exploiting bitwise operators, and save the result in a new raster image

Note

License: GPL 3.0 Based on the `modis.qc.R` script by Yann Chemin (2008) (https://goo.gl/7Fhreo) license GPL 3.0

Author(s)

Lorenzo Busetto, phD (2017)

Luigi Ranghetti, phD (2017)

`MODIStsp_read_xml` *Read MODIS products characteristics from XML*

Description

function used to parse the XML file used to store the characteristics of MODIS Land Products and store them in the "prod_opts" data frame

Usage

```
MODIStsp_read_xml(prodopts_file, xml_file)
```

Arguments

`prodopts_file`	string filename of the RData in which to store the data parsed from the XML file
`xml_file`	string filename of the XML file containing the MODIS products characteristics

Details

The function parses the XML file product by product, stores data in a data frame and saves the data frame within the "MODIStsp_previous" RData file as a list of lists

Value

NULL - retrieved data are stored in the specified RData file

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

`MODIStsp_resetindexes` *Remove custom spectral indexes*

Description

Function used to remove all user-defined Spectral Indexes from MODIStsp, thus resetting the list of available indexes to the default ones.

Usage

```
MODIStsp_resetindexes()
```

Value

The function is called for its side effects. On success, the MODIStsp_indexes.json file is modified so to remove all previously custom-specified Spectral Indexes.

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017) <busetto.l@irea.cnr.it>

See Also

MODIStsp_addindex

Examples

```
## Not run:
# Remove all custom-defined spectral indexes from an options file

# Add a custom index for testing purposes
library(jsonlite)
opts_jsfile = system.file("testdata/test_addindex.json",
                          package = "MODIStsp")
 MODIStsp_addindex(
   opts_jsfile = opts_jsfile,
   gui = FALSE,
   new_indexbandname = paste0("Index_", as.character(sample(10000, 1))),
   new_indexformula = "b1_Red - b2_NIR",
   new_indexfullname = paste0("Index_", as.character(sample(10000, 1)))
```

```
  )

opts <- jsonlite::fromJSON(indexes_file)
opts$custom_indexes[1]

# Now remove all custom indexes
MODIStsp_resetindexes()
opts <- jsonlite::fromJSON(opts_jsfile)
opts$custom_indexes[1]

## End(Not run)
```

`MODIStsp_vrt_create` *Create MODIStsp virtual files*

Description

Function used to create virtual files from time series of single-band files corresponding to different acquisition dates. The function takes as input the folder in which the single-band files are stored, and creates a ENVI Meta file and/or a GDAL vrt file that allows access to the full time series as if it was a single physical file. Created virtual files are stored in the "Time Series" subfolder of 'out_prod_folder"

Usage

```
MODIStsp_vrt_create(
  sensor,
  out_prod_folder,
  bandnames,
  bandsel,
  nodata_out,
  indexes_bandnames,
  indexes_bandsel,
  indexes_nodata_out,
  quality_bandnames,
  quality_bandsel,
  quality_nodata_out,
  file_prefixes,
  ts_format,
  out_format,
  verbose
)
```

Arguments

`sensor`	`character ["Terra"| "Aqua"	"Both"]` MODIS platform to be considered. (Ignored for MCD* products). Default: "Both"

`out_prod_folder`
: `character` Main output folder.

`bandnames`
: names of all layers available for the product being processed

`bandsel`
: `character array` Original MODIS layers to be processed. You can get a list of available layers for a given product using function `MODIStsp_get_prodlayers` (e.g., MODIStsp_get_prodlayers("M*D13Q1")$bandnames), Default: NULL

`nodata_out`
: `numeric` Output nodata value to be used in vrt files

`indexes_bandnames`
: names of all indexes available for the product being processed

`indexes_bandsel`
: `character array`Spectral Indexes to be computed starting from reflectance bands. You can get a list of available quality layers for a given product using function `MODIStsp_get_prodlayers` (e.g., MODIStsp_get_prodlayers("M*D13Q1")$indexes_bandnam
Default: NULL

`indexes_nodata_out`
: nodata value for indexes vrts

`quality_bandnames`
: names of all quality indicators available for the product being processed

`quality_bandsel`
: `character array` Quality Indicators to be computed starting from bit fields of original MODIS layers. You can get a list of available quality layers for a given product using function `MODIStsp_get_prodlayers` (e.g., MODIStsp_get_prodlayers("M*D13Q1")$qual
Default: NULL

`quality_nodata_out`
: nodata value for quality vrts

`file_prefixes`
: `character array (2)` file_prefixes for TERRA and AQUA - used to identify the files corresponding to each sensor

`ts_format`
: `character` `["ENVI" | "GDAL" | "Both"]` Required output format for virtual file.

`out_format`
: `character` `["ENVI" | "GTiff"]` Format of images used as "input" for the vrt and contained in out_prod_folder/band folders.

`verbose`
: `logical` If FALSE, suppress processing messages, Default: TRUE

Value

NULL - the function is called for its side effects

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2014-2017)

Luigi Ranghetti, phD (2015)

process_message *Spawn processing update messages*

Description

helper function to provide processing messages

Usage

```
process_message(mess_text, verbose = TRUE)
```

Arguments

mess_text	character text to be shown in the processing windows and/or the console
verbose	logical If FALSE, suppress processing messages, Default: TRUE

Value

The function is called for its side effects

Author(s)

Lorenzo Busetto, phD (2017)

reproj_bbox *Reproject a bounding box*

Description

Helper function used to reproject bounding boxes; setting the parameter 'enlarge' allows to choose if the new one would be the one which completely includes the original extent in the output projection, or if is simply the one obtained by reprojecting the upper-left and the lower-right corners.

Usage

```
reproj_bbox(bbox, in_proj, out_proj, enlarge = TRUE)
```

Arguments

bbox	The input bounding box (it can be a matrix obtained from sp::bbox(), or a numeric vector in the format (xmin, ymin, xmax, ymax)).
in_proj	(crs | character) crs of the input projection, or string coercible to it using sf::st_crs() (e.g., WKT or numeric EPSG code)
out_proj	crs crs of the output projection, or string coercible to it using sf::st_crs() (e.g., WKT or numeric EPSG code)
enlarge	'logical" if TRUE, the reprojected bounding box is the one which completely include the original one; if FALSE, it is simply the one obtained by reprojecting the upper-left and the lower-right corners.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2015)

`set_bandind_matrix` *Helper function to determine the bands needed to compute SIs and QIs*

Description

FUNCTION_DESCRIPTION

Usage

```
set_bandind_matrix(
  bandnames,
  bandsel,
  indexes_bandnames,
  indexes_bandsel,
  indexes_formula,
  quality_bandnames,
  quality_bandsel,
  quality_source
)
```

Arguments

`bandnames`	names of all layers available for the product being processed
`bandsel`	`character array` Original MODIS layers to be processed. You can get a list of available layers for a given product using function `MODIStsp_get_prodlayers` (e.g., MODIStsp_get_prodlayers("M*D13Q1")$bandnames), Default: NULL
`indexes_bandnames`	names of all indexes available for the product being processed
`indexes_bandsel`	`character array`Spectral Indexes to be computed starting from reflectance bands. You can get a list of available quality layers for a given product using function `MODIStsp_get_prodlayers` (e.g., MODIStsp_get_prodlayers("M*D13Q1")$indexes_bandnam Default: NULL
`indexes_formula`	formulas of all indexes available for the product being processed
`quality_bandnames`	names of all quality indicators available for the product being processed

`quality_bandsel`
: `character array` Quality Indicators to be computed starting from bit fields of original MODIS layers. You can get a list of available quality layers for a given product using function `MODIStsp_get_prodlayers` (e.g., MODIStsp_get_prodlayers("M*D13Q1")$qual Default: NULL

`quality_source`
: sources of data (original layers) of all quality indicators available for the product being processed

Value

`matrix` containing info on which bands are needed for computing each available QI or SI

Author(s)

Lorenzo Busetto, phD (2017)

`split_nodata_values` *Split NODATA values or create matrix for reclassification*

Description

Internal functions: split_nodata_values splits the ranges of NODATA saved in the xml product file to a readable vector of NoData values; create_nodata_rcl creates the matrix for the reclassification of NODATA values to be used with raster::reclassify function.

Usage

```
split_nodata_values(nodata_in, take_all = TRUE)

create_nodata_rcl(nodata_in, nodata_out)
```

Arguments

`nodata_in`
: Character vector corresponding to input NoData values as saved in the xml product file (one or more values per band).

`take_all`
: Logical: if TRUE (default), all the NoData values are considered; if FALSE, only the last one is taken. See "details" for the meaning of this parameter.

`nodata_out`
: Character vector corresponding to output NoData values as saved in the xml product file (one single value per band).

Details

MODIS products can have more than one NoData values (sometimes with different meanings, e.g. 255 = "fill" and 254 = "detector saturated" in MOD09A1 product). By setting "Change NoData values" to "Yes" in the GUI, all the NoData values are coerced to one single new NoData value; conversely, setting it to "No" only one value is assumed to be NoData. The parameter `take_all` is

assumed to be used in this way, by using this function with `take_all = TRUE` with "Change NoData values" = "Yes" and `take_all = FALSE` with "Change NoData values" = "No".

In the xml product file, NoData ranges are set as:

- `x` for products with single NoData values;
- `x,y,z` for products with a vector of NoData values;
- `x:y` for products with a range of NoData values;
- `x:y,z` for a combination of NoData ranges and/or values.

In split_nodata_values *NoData values are assumed to be integer*: this means that intervals are splitted in integer values (e.g. "250:255" becomes "250 251 252 253 254 255"). Conversely, function create_nodata_rcl creates intervals, so it can also manage float values (in practice, this should not make difference within MODIS products, since NoData values are always integer values).

This function interprets these strings and convert them in vectors with single values. Notice that the first NoData value is the only one which is considered if 'Change NoData values' was set to 'No'.

Value

split_nodata_values returns a list with the same length of `nodata_in` vector, in which each element is a vector with all the NoData values.

create_nodata_rcl returns a list of matrices in the format specified for parameter `rcl` in raster::reclassify. The parameter `right` is intended to be used as `right = NA`.

Author(s)

Luigi Ranghetti, phD (2018)

Examples

```
MODIStsp:::create_nodata_rcl(c("255","250,254:255"), c("255","255"))
```

Index