MODIStsp: A Tool for Automatic Preprocessing of MODIS Time Series - v1.3.9

Lorenzo Busetto (busetto.l@irea.cnr.it), Luigi Ranghetti (ranghetti.l@irea.cnr.it)

2020-05-10

Contents

1 Introduction

MODIStsp is a novel "R" package allowing to automatize the creation of time series of rasters derived from MODIS Land Products data. It allows to perform several preprocessing steps on MODIS data available within a given time period.

Development of MODIStsp started from modifications of the ModisDownload "R" script by Thomas Hengl (2010), and successive adaptations by Babak Naimi (2014). The basic functionalities for download and preprocessing of MODIS datasets provided by these scripts were gradually incremented with the aim of:

- developing a stand-alone application allowing to perform several preprocessing steps (e.g., download, mosaicing, reprojection and resize) on all available MODIS land products by exploiting a powerful and user-friendly GUI front-end;
- allowing the creation of time series of both MODIS original layers and additional Quality Indicators (e.g., data acquisition quality, cloud/snow presence, algorithm used for data production, etc.) extracted from the aggregated bit-field QA layers;
- allowing the automatic calculation and creation of time series of several additional Spectral Indexes starting form MODIS surface reflectance products.

All processing parameters can be easily set with a user-friendly GUI, although non-interactive execution exploiting a previously created Options File is possible. Stand-alone execution outside an "R" environment is also possible, allowing to use scheduled execution of MODIStsp to automatically update time series related to a MODIS product and extent whenever a new image is available.

Required MODIS HDF files are automatically downloaded from NASA servers and resized, reprojected, resampled and processed according to user's choices. For each desired output layer, outputs are saved as single-band rasters corresponding to each acquisition date available for the selected MODIS product within the specified time period. "R" *RasterStack* objects with temporal information as well as Virtual raster files (GDAL vrt and ENVI META files) facilitating access to the entire time series can be also created.

2 Installation

`MODIStsp` requires R v >= 3.2.1 and GDAL (Geospatial Data Abstraction Library) v >= 1.11.1 **with support for HDF4 raster format** to be installed in your system. Brief instructions for installing R and GDAL can be found HERE.

2.1 On Windows

You can install the stable version of `MODIStsp`, from CRAN:

```
install.packages("MODIStsp")
```

, or the development version (containing the latest improvements and bug fixes):

```
library(devtools)
install_github("ropensci/MODIStsp")
```

Note that **if the `GTK+` library is not already installed on your system, installation may fail**. In that case, please install and load the `gWidgetsRGtk2` library beforehand:

```
install.packages("gWidgetsRGtk2")
library(gWidgetsRGtk2)
```

Upon loading `gWidgetsRGtk2` , an error window will probably appear. This signals that library "GTK+" is not yet installed on your system or is not on your PATH. To install it press "OK". A new window dialog window will appear, asking if you want to install "GTK+". Select "Install GTK" and then "OK" . Windows will download and install the GTK+ library. When it finishes, the RSession should be restarted and you should be ready to go !

In case RStudio does not automatically restart or continuously asks to install GTK+ again, kill it form "Task Manager" (or restart the R session from RStudio "Session" menu), reload RStudio and the try to reload `gWidgetsRGtk2`. If it loads correctly, you should be ready to go.

If it still fails, try downloading the GTK+ bundle from:

http://ftp.gnome.org/pub/gnome/binaries/win64/gtk+/2.22/gtk+-bundle_2.22.1-20101229_win64.zip (OR http://ftp.gnome.org/pub/gnome/binaries/win32/gtk+/2.22/gtk+-bundle_2.22.1-20101227_win32.zip if on Win32)

, unzip the archive on a folder of your choice (e.g., `C:\\Program Files\\GTK+`), then add the path to its "bin" subfolder (e.g., `C:\\Program Files\\GTK+\\bin\\` to your system PATH environment variable.

Restart your system and try loading again `gWidgetsRGtk2`: if it loads ok, you should be ready to install `MODIStsp`

2.2 On Linux systems

To install `MODIStsp` on Linux, you have to first install the following required dependencies:

- `Cairo` >= 1.0.0, `ATK` >= 1.10.0, `Pango` >= 1.10.0, `GTK+` >= 2.8.0, `GLib` >= 2.8.0 (required by package `RGtk2`)
- `Curl` (required by package `curl`)
- `GDAL` >= 1.6.3, `PROJ.4` >= 4.4.9 (required by package `rgdal`)

On *Debian and Ubuntu-based* systems, to install those packages open a terminal and type:

```
sudo apt-get install r-cran-cairodevice r-cran-rgtk2 libcairo2-dev libatk1.0-dev libpango1.0-dev
libgtk2.0-dev libglib2.0-dev libcurl4-openssl-dev libgdal-dev libproj-dev
```

On *rpm-base systems*, to install packages open a terminal and type:

```
sudo yum install libcairo2-devel libatk1.0-devel libpango1.0-devel gtk2 gtk2-devel
glib2-devel libcurl-devel gdal-devel proj proj-devel proj-epsg proj-nad
```

Then, you can install the stable version of MODIStsp from CRAN:

```
install.packages("MODIStsp")
```

, or the development version (containing the latest improvements and bug fixes) from github;

```
library(devtools)
install_github("ropensci/MODIStsp")
```

2.3 On Mac OS

NOTE: The following installation notes should be valid for MODIStsp installation on R 3.4.0 and above with Mac OSX Sierra. They were mainly taken (i.e., blatantly copied...) from: https://zhiyzuo.github.io/installation-rattle/. Thanks to Zhiya Zuo for providing this!

To properly install `MODIStsp` you will need to first install package `RGTk2`. This is a somehow difficult operation. The following instructions should help:

1. Check your Mac OS X version and update if necessary:

Enter the following command in terminal to check your macOS version. Expected output is as below the dashed line —.

```
~$ sw_vers
-----------------------
ProductName:    Mac OS X
ProductVersion: 10.12.6
```

```
BuildVersion:   16G29
```

If your system is above 10.11, continue. Otherwise, upgrade it to Sierra.

Install homebrew if you do not have it already installed. homebrew is a very convenient package manager for macOS. To do so, open a terminal, copy the following command in it and hit Enter:

```
~$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Follow the instructions to get brew ready. When inserting your password, nothing will show up for security reasons. Just hit Enter when you are finished.

When brew is finished, copy the following command in terminal and hit Enter:

```
~$ touch ~/.bash_profile
~$ echo "export PATH=/usr/local/bin:$PATH
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:/usr/local/lib/pkgconfig/gtk+-2.0.pc:/opt/X11/lib/pkgco
~$ source ~/.bash_profile
```

2. Install the `cairo` library with x11 support. Enter the following into your terminal:

```
~$ brew uninstall --force cairo --ignore-dependencies
~$ brew cask install xquartz
~$ brew install --with-x11 cairo
```

3. Install the `gtk+` library:

To do so, you first have to change the way homebrew wants to install gtk+. In an editor, write:

```
~$ brew edit gtk+
```

A text editor will open. Look in the file, and find a section that begins with "def install". Substitute the current `args` section with the following text:

```
def install
 args = [
         "--disable-dependency-tracking",
         "--disable-silent-rules",
         "--prefix=#{prefix}",
         "--disable-glibtest",
         "--enable-introspection=yes",
         # "--disable-visibility",
         # "--with-gdktarget=quartz",
         "--with-gdktarget=x11",
         "--enable-x11-backend"
        ]
```

Save the modified file using `ctrl+x ctrl+c`, followed by `y` to quit emacs. Now install the library using:

```
~$ brew install --build-from-source --verbose gtk+
```

4. Update your path so that `gtk+` is recognized, using:

```
~$ export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:/usr/local/lib/pkgconfig/gtk+-2.0.pc:/opt/X11/lib/pk
```

5. **Install `RGtk2` from source**:

- **Download the newest source file for RGtk2** from https://CRAN.R-project.org/package=gWidgetsRGtk2.
- Assuming that the path to this file is ~/Downloads. Run the following in terminal (change the path if you did not download in ~/Downloads):

```
~$ cd ~/Downloads
~/Downloads$ R CMD INSTALL RGtk2_2.20.33.tar.gz
```

(Note that the name of the tar.gz file may vary depending on when you downloaded the file).

6. **Open R and run**:

```
library(RGtk2)
```

hopefully, `RGtk2` will load without errors! If so, you should be ready to go, and you can:

7. **Install MODIStsp** from CRAN:

```
install.packages("MODIStsp")
MODIStsp()
```

or the development version from GitHub:

```
library(devtools)
install_github("ropensci/MODIStsp", ref = "master")
MODIStsp()
```

Good luck!

3 Running the tool in Interactive Mode: the MODIStsp GUI

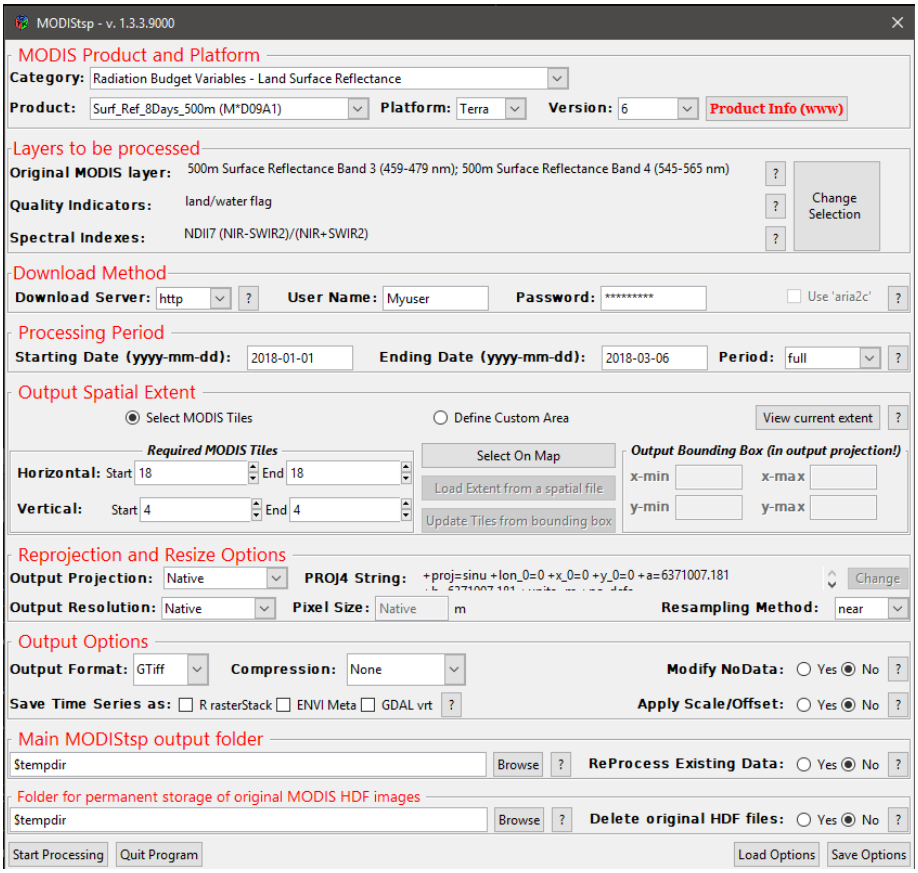
The easiest way to use `MODIStsp` is to use its powerful GUI (Graphical User Interface) for selection of processing options, and then run the processing.

To open the GUI, load the package and launch the MODIStsp function, with no parameters:

```
library(MODIStsp)
MODIStsp()
```

This **opens a GUI** from which processing options can be specified and eventually saved (or loaded) (Note 1: PCs with a small screen can fail to visualize the whole GUI; in this case, the user can add scroll bars with `MODIStsp(scrollWindow=TRUE)`); Note 2: At the first execution of `MODIStsp`, a Welcome screen will appear, signaling that `MODIStsp` is searching for a valid GDAL installation. Press "OK" and wait for GDAL to be found. If nothing happens for a long time (e.g., several minutes), `MODIStsp` (and in particular the gdalUtils package on which it relies) is not finding a valid GDAL installation in the more common locations. To solve the problem: 1. Ensure that GDAL is properly installed in your system 2. (On Windows) If it is installed, verify that GDAL is in your system PATH. and that the *GDAL_DATA* environment variable is correctly set (You can find simple instructions HERE) 3. If nothing works, signal it in the issues GitHub page of `MODIStsp` and we'll try to help!

The GUI allows selecting all processing options required for the creation of the desired MODIS time series. The main available processing options are described in detail in the following.



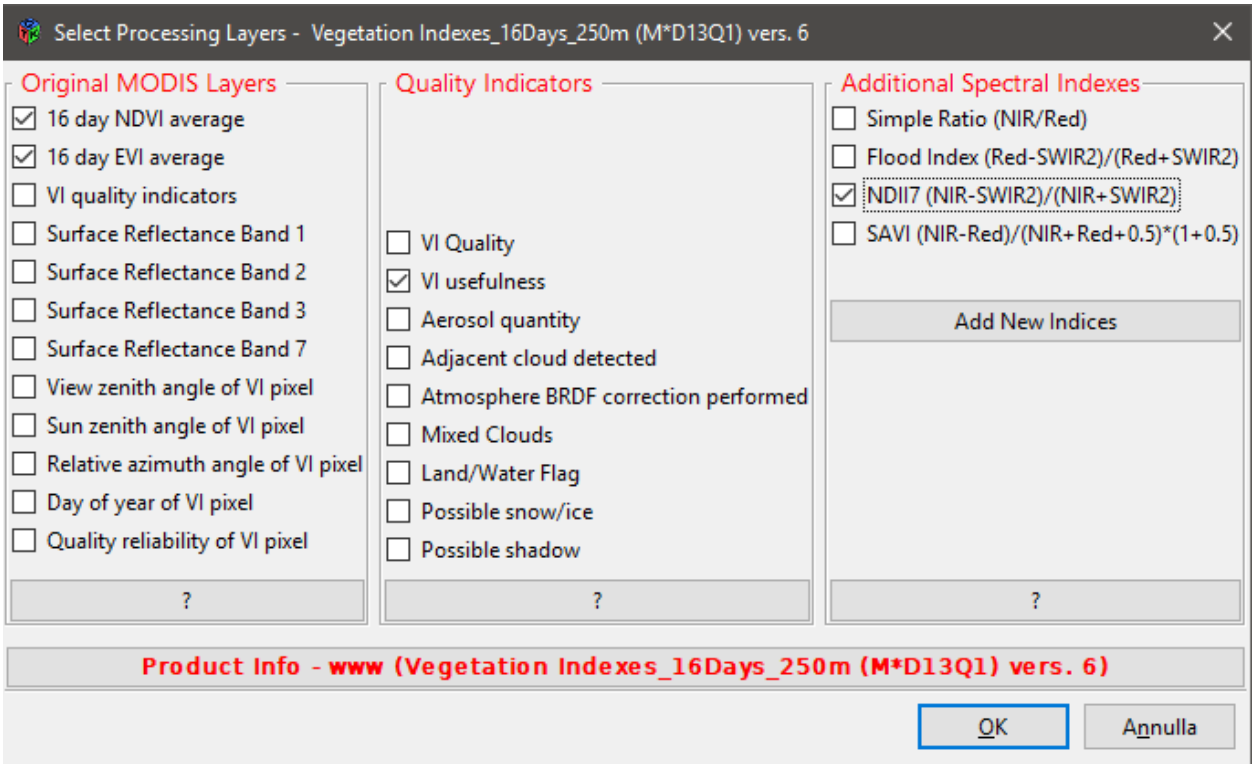
3.1 Selecting Processing Parameters

3.1.1 *MODIS Product, Platform and Layers:*

The top-most menus allow to specify details of the desired output time series:

1. **"Category"** and **"Product"**: Selects the MODIS product of interest;
2. **MODIS platform(s)**: Selects if only TERRA, only AQUA or Both MODIS platforms should be considered for download and creation of the time series;
3. **version**: Selects whether processing version 5 or 6 (when available) of MODIS products has to be processed

After selecting the product and version, clicking the **"Change Selection"** button opens the Select Processing Layers GUI panel, from which the user **must** select which MODIS original layers and/or derived Quality Indexes (QI) and Spectral Indexes (SI) layers should be processed:

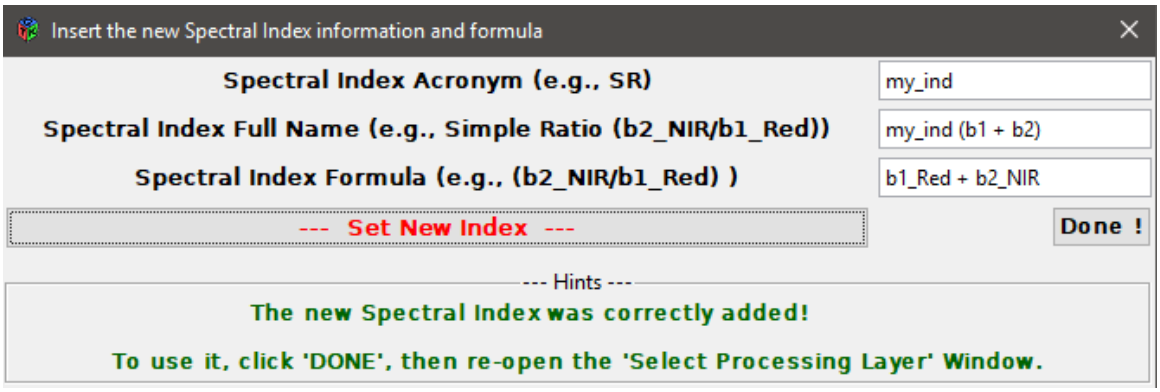


- The left-hand frame allows to select which *original MODIS layers* should be processed.
- The central frame allows to select which *Quality Indicators should be extracted* from the original MODIS Quality Assurance layers.
- For MODIS products containing surface reflectance data, the right-hand frame allows to select which additional *Spectral Indexes should be computed*. The following commonly used Spectral Indexes are available for computation by default:

Table II: List of default Spectral Indexes available in MODIStsp

Index Acronym	Index name and reference
NDVI	Normalized Difference Vegetation Index (Rouse et al. 1973)
EVI	Enhanced Vegetation Index (Huete et al. 2002)
SR	Simple Ratio (Tucker 1979)
NDFI	Normalized Difference Flood Index (Boschetti et al. 2014)
NDII7 (NDWI)	Normalized Difference Infrared Index - Band 7 (Hunt and Rock 1989)
SAVI	Soil Adjusted Vegetation Index (Huete 1988)
NDSI	Normalized Difference Snow Index (Hall et al. 2002)
NDII6	Normalized Difference Infrared Index - band 6 (Hunt and Rock 1989)
GNDVI	Green Normalized Difference Vegetation Index (Gitelson and Merzlyak 1998)
RGRI	Red Green Ratio Index (Gamon and Surfus 1999)
GRVI	Green-Red ratio Vegetation Index (Tucker 1979)

You can however **specify other SIs to be computed without modifying MODIStsp source code** by clicking on the ***"Add Custom Index"*** button, which allow to provide info related to the new desired SI using a simple GUI interface.



Provided information (e.g., correct band-names, computable formula, etc. . .) is automatically checked upon clicking "Set New Index". On success, the new index is added in the list of available ones for all products allowing its computation. Clicking "Done !" returns to the main.

NOTE All custom defined indexes can be removed by using the `MODIStsp_resetindexes()` function

3.1.2 *Download Method:*

Select the method to be used for download. Available choices are:

1. **http**: download through ftp from NASA lpdaac http archive (http://e4ftl01.cr.usgs.gov). This requires providing a user name and password, which can be obtained by registering an account at the address https://urs.earthdata.nasa.gov/profile;
2. **offline**: this option allows to process/reprocess HDF files already available on the user's PC without downloading from NASA – useful if the user already has an archive of HDF images, or to reprocess data already downloaded via MODIStsp to create time series for an additional layer (*It is fundamental that the HDFs are those directly downloaded from NASA servers* ! (See here for additional details).

Checking the **use_aria2c** option allows to accelerate the download from NASA archives. This requires however that that the "aria2c" software is installed in your system. To download and install it, see: https://aria2.github.io/

NOTE: The best performances are usually achieved using http*, though that may vary depending on network infrastructure.*

3.1.3 *Processing Period:*

Specify the starting and ending dates to be considered for the creation of the time in the series corresponding fields. Dates **must** be provided in the *yyyy–mm–dd* format (e.g., 2015-01-31)

The **Period** drop-down menu allows to choose between two options:

1. **full**: all available images between the starting and ending dates are downloaded and processed;
2. **seasonal**: data is downloaded only for one part of the year, but for multiple years. For example, if the starting date is 2005-03-01 and the ending is 2010-06-01, only the images of March, April and May for the years between 2005 and 2010 will be downloaded. This allows to easily process data concerning a particular season of interest.

3.1.4 *Spatial Extent:*

Allows to define the area of interest for the processing. Two main options are possible:

1. **Select MODIS Tiles**: specify which MODIS tiles need to be processed either by:

a. Using the "Start" and "End" horizontal and vertical sliders in the *Required MODIS Tiles* frame.

b. clicking the **"Select on Map"** button. A map will open in a browser window, allowing interactive selection of the required tiles

During processing, data from the different tiles is mosaiced, and a single file covering the total area is produced for each acquisition date

2. **Define Custom Area**: specify a custom spatial extent for the desired outputs either by:

   a. Manually inserting the coordinates of the Upper Left and Lower Right corners of the area of interest in the **Bounding Box** frame. *Coordinates of the corners must be provided in the coordinate system of the selected output projection*;

   b. clicking the **"Load Extent from a Spatial File" and selecting a raster or vector spatial file**. In this case, the bounding box of the selected file is retrieved, converted in the selected output projection, and shown in the "Bounding Box" frame. Required input MODIS tiles are also automatically retrieved from the output extent, and the tiles selection sliders modified accordingly.

   c. clicking the **"Select on Map"** button. A map will open in a browser window, allowing interactive selection of the spatial extent using the tools on the left.

Note: clicking the "show current extent" will open a browser window highlighting the currently selected spatial extent.

3.1.5 *Reprojection and Resize*

Specify the options to be used for reprojecting and resizing the MODIS images.

- **"Output Projection"**: select either the Native MODIS projection (Default) or specify a user-defined one. To specify a user selected projection, select "User Defined" and then insert a valid "Proj4" string in the pop-up window. Validity of the Proj4 string is automatically checked, and error messages issued if the check fails;
- **"Output Resolution"**, **"Pixel Size"** and **"Reprojection Method"**: specify whether output images should inherit their spatial resolution from the original MODIS files, or be resampled to a user-defined resolution. In the latter case, output spatial resolution must be specified in the measure units of the selected output projection. Resampling method can be chosen among "Nearest Neighbour" and "Mode" (Useful for down-sampling purposes). Other resampling methods (e.g., bilinear, cubic) are not currently supported since i) they cannot be used for resampling of categorical variables such as the QA and QI layers, and ii) using them on continuous variable (e.g., reflectance, VI values) without performing an a-priori data cleaning would risk to contaminate the values of high-quality observations with those of low-quality ones.

3.1.6 *Output Options*

Several processing options can be set using check-boxes:

- **Output Files Format**: Two of the most commonly formats used in remote sensing applications are available at the moment: ENVI binary and GeoTiff. If GeoTiff is selected, the type of file compression can be also specified among "None", "PACKBITS", "LZW" and "DEFLATE".
- **Save Time Series as**: Specify if virtual multitemporal files should be created. These virtual files allow access to the entire time series of images as a single file without the need of creating large multitemporal raster images. Available virtual files formats are "R" rasterStacks, ENVI meta-files and GDAL "vrt" files. In particular, "R" rasterStacks may be useful in order to easily access the preprocessed MODIS data within "R" scripts (see also https://docs.ropensci.org/MODIStsp/articles/output.html).
- **Modify No Data**: Specify if NoData values of MODIS layers should be kept at their original values, or changed to those specified within the "MODIStsp_Products_Opts" XML file. By selecting "Yes" in

the "Change Original NoData values" check-box, NoData of outputs are set to the largest integer value possible for the data type of the processed layer (e.g., for 8-bit unsigned integer layers, NoData is set always to 255, for 16-bit signed integer layers to 32767, and for 16-bit unsigned integer layers to 65535). Information about the new NoData values is stored both in the output rasters, and in the XML files associated with them. **NOTE:** Some MODIS layers have multiple NoData (a.k.a. *fill*) values. if *Modify No Data* is set to "Yes", `MODIStsp` will convert all *fill* values to a common output NoData value!

- **Apply Scale/Offset**: Specify if scale and offset values of the different MODIS layers should be applied. If selected, outputs are appropriately rescaled on the fly, and saved in the true "measure units" of the selected parameter (e.g., spectral indexes are saved as floating point values; Land Surface Temperature is saved in degrees Kelvin, etc.).

3.1.7 Main MODIStsp Output Folder

Select the main folder where the pre-processed time series data will be stored. All `MODIStsp` outputs **will be placed in specific sub-folders of this main folder** (see https://docs.ropensci.org/MODIStsp/articles/output.html for details on `MODIStsp` naming conventions)-.

The **"Reprocess Existing Data"** check-box allows to decide if images already available should be reprocessed if a new run of MODIStsp is launched with the same output folder. If set to "No", MODIStsp skips dates for which output files following the MODIStsp naming conventions are already present in the output folder. This allows to incrementally extend MODIS time series without reprocessing already available dates.

3.1.8 *Folder for permanent storage of original MODIS HDF images*

Select the folder where downloaded **original MODIS HDF files** downloaded from NASA servers will be stored.

The **"delete original HDF files"** check-box allows also to decide if the downloaded images should be deleted from the file system at the end of the processing. To avoid accidental file deletion, this is always set to "No" by default, and a warning is issued before execution whenever the selection is changed to "Yes".

3.2 Saving and Loading Processing Options

Specified processing parameters can be saved to a JSON file for later use by clicking on the ***Save Options*** button.

Previously saved options can be restored clicking on the ***Load Options*** button and navigating to the previously saved JSON file.

(Note that at launch, `MODIStsp` ***always reloads automatically the processing options used for its last successful run*** .

3.3 Starting the processing

Once you are happy with your choices, click on **Start Processing**. `MODIStsp` will start accessing NASA servers to download and process the MODIS data corresponding to your choices.

For each date of the specified time period, `MODIStp` downloads and preprocesses all hdf images required to cover the desired spatial extent. Informative messages concerning the status of the processing are provided on the console, as well as on a self-updating progress window.

The processed time series are saved in specific subfolders of the main selected output folder, as explained in detail HERE.

4 Non-Interactive Execution from within R

4.1 Specifying a saved "Options file"

`MODIStsp` can be launched in non-interactive mode within an `R` session by setting the optional `GUI` parameter to FALSE, and the `Options_File` parameter to the path of a previously saved JSON Options file. This allows to exploit `MODIStsp` functionalities within generic "R" processing scripts:

```
library(MODIStsp)

# **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!

# --> Specify the path to a valid options file saved in advance from MODIStsp GUI
# 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.

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

# --> Launch the processing
MODIStsp(gui = FALSE, options_file = options_file, verbose = FALSE)

# Outputs are in this case in subfolder "MODIStsp/VI_16Days_1Km_v6" of `base::tempdir()`:

out_fold <- file.path(tempdir(), "MODIStsp/VI_16Days_1Km_v6")
list.files(out_fold)
#> [1] "EVI"         "NDVI"        "QA_usef"     "Time_Series"

list.files(file.path(out_fold ,"EVI"))
#> [1] "MOD13A2_EVI_2016_161.tif" "MOD13A2_EVI_2016_177.tif"
```

4.2 Looping over different Options files

If you need to process different MODIS products, you can prepare beforehand different MODIStsp options files by using the GUI, and then loop over them like this:

```
opts_files <- c(system.file("testdata/test_MOD13A2.json", package = "MODIStsp"),
                system.file("testdata/test_MOD10A2.json", package = "MODIStsp"))

for (opts_file in opts_files) {
  MODIStsp(gui = FALSE, options_file = opts_file)
}

# MOD13A2 ouputs
out_fold <- file.path(tempdir(), "MODIStsp/VI_16Days_1Km_v6")
list.files(out_fold)
#> [1] "EVI"         "NDVI"        "QA_usef"     "Time_Series"
list.files(file.path(out_fold ,"EVI"))
#> [1] "MOD13A2_EVI_2016_161.tif" "MOD13A2_EVI_2016_177.tif"

# MOD10A2 ouputs
out_fold <- file.path(tempdir(), "MODIStsp/Surf_Temp_8Days_1Km_v6")
list.files(out_fold)
#> [1] "Days_Clear"    "LST_Day_1km"   "LST_Night_1km" "Time_Series"
```

```
list.files(file.path(out_fold ,"LST_Night_1km"))
#> [1] "MOD11A2_LST_Night_1km_2016_153.tif" "MOD11A2_LST_Night_1km_2016_161.tif"
#> [3] "MOD11A2_LST_Night_1km_2016_169.tif" "MOD11A2_LST_Night_1km_2016_177.tif"
```

4.3 Looping on different spatial extents

Specifying also the `spatial_file_path_` parameter overrides for example the output extent of the selected Options File. This allows to perform the same preprocessing on different extents using a single Options File. For example:

```
# Run 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
library(MODIStsp)
options_file <- system.file("testdata/test_MOD13A2.json", package = "MODIStsp")
spatial_file <- system.file("testdata/lakeshapes/garda_lake.shp", package = "MODIStsp")
MODIStsp(gui = FALSE, options_file = options_file,
         spatial_file_path = spatial_file, verbose = FALSE)

# --> Create a character array containing a list of shapefiles (or other spatial files)
extent_list = list.files(system.file("testdata/lakeshapes/", package = "MODIStsp"),
                         full.names = TRUE, "\\.shp$")
extent_list

#> [1] "D:/Documents/Source/git/MODIStsp/inst/testdata/lakeshapes/garda_lake.shp"
#> [2] "D:/Documents/Source/git/MODIStsp/inst/testdata/lakeshapes/iseo_lake.shp"

# --> Loop on the list of spatial files and run MODIStsp using each of them to
# automatically define the output extent (A separate output folder is created for
# each input spatial file).

for (single_shape in extent_list) {
  MODIStsp(gui = FALSE, options_file = options_file,
           spatial_file_path = single_shape )
}

# output files are placed in separate folders. For example here you can see how to
# access a list of produced files and plot one of them:
outfiles_garda <- list.files(
  file.path(tempdir(), "MODIStsp/garda_lake/VI_16Days_1Km_v6/EVI"),
  full.names = TRUE)
library(raster)
plot(raster(outfiles_garda[1]))

outfiles_iseo <- list.files(
  file.path(tempdir(), "MODIStsp/iseo_lake/VI_16Days_1Km_v6/EVI"),
  full.names = TRUE)
plot(raster(outfiles_iseo[1]))
```

5 Stand-alone execution and scheduled processing

5.1 Stand-alone execution

- MODIStsp can be also executed as a "stand-alone" application(i.e., without having to open R/RStudio); to do this, from R launch the function `MODIStsp_install_launcher()`.

In a Linux operating system this function creates a desktop entry (accessible from the menu in the sections "Science" and "Geography") and a symbolic link in a known path (default: /usr/bin/MODIStsp). In Windows, a link in the Start Menu and optionally a desktop shortcut are created. See `?install_MODIStsp_launcher` for details and path customization.

Double-clicking those files or launching them from a shell without parameters will launch `MODIStsp` in interactive mode. Non-interactive mode is triggered by adding the "-g" argument to the call, and specifying the path to a valid Options File as "-s" argument:

- Linux: `MODIStsp -g -s "/yourpath/youroptions.json"` (see `MODIStsp -h` for details).
- Windows:`your_r_library\MODIStsp\ExtData\Launcher\MODIStsp.bat -g -s "yourpath/youroptions.json"` (see `C:\Users\you\Desktop\MODIStsp -h` for details).

If you do not want to install any link, launchers can be found in the subdirectory "MODIStsp/ExtData/Launcher" of your library path.

5.2 Scheduled Processing

Stand-alone non-interactive execution can be used to periodically and automatically update the time series of a selected product over a given study area. To do that, you should simply:

1. Open the `MODIStsp` GUI, define the parameters of the processing specifying a date in the future as the "Ending Date" and save the processing options. Then quit the program.
2. Schedule non-interactive execution of the launcher installed as seen before (or located in the subdirectory "MODIStsp/ExtData/Launcher" of your library path) as windows scheduled task (or linux "cron" job) according to a specified time schedule, specifying the path of a previously saved Options file as additional argument:

5.2.0.1 On Linux

- edit your crontab by opening a terminal and type:

```
crontab -e
```

- add an entry for the launcher. For example, if you have installed it in /usr/bin and you want to run the tool every day at 23.00, add the following row:

```
0 23 * * * /bin/bash /usr/bin/MODIStsp -g -s "/yourpath/youroptions.json"
```

5.2.0.2 On Windows

- create a Task following these instructions; add the path of the MODIStsp.bat launcher as Action (point 6), and specifying `-g -s "X:/yourpath/youroptions.json"` as argument.

6 Outputs Format and Naming Conventions

6.1 Single-band outputs

Output raster files are saved in specific subfolders of the main output folder. In particular, **a separate subfolder** is created for each processed original MODIS layer, Quality Indicator or Spectral Index. Each subfolder contains one image for each processed date, created according to the following naming conventions:

`myoutfolder/"Layer"/"ProdCode"_"Layer"_"YYYY"_"DOY"."ext"`

(e.g., myoutfolder/NDVI/MOD13Q1_NDVI_2000_065.dat)

, where:

- ***Layer*** is a short name describing the dataset (e.g., b1_Red, NDII, UI);
- ***ProdCode*** is the code name of the MODIS product from which the image was derived (e.g., MOD13Q1);
- ***YYYY*** and ***DOY*** correspond to the year and DOY (Day of the Year) of acquisition of the original MODIS image;
- ***ext*** is the file extension (.tif for GTiff outputs, or .dat for ENVI outputs).

6.2 Virtual multi-band outputs

ENVI and/or GDAL virtual time series files and *RasterStack* RData objects are instead stored **in the "Time_Series" subfolder** if required.

Naming convention for these files is as follow:

`myoutfolder/Time_Series/"vrt_type"/"Sensor"/"Layer"/"ProdCode"_"Layer"_"StartDOY"_"StartYear_"EndDOY"_"`

(e.g., myoutfolder/Time_Series/RData/Terra/NDVI/MOD13Q1_MYD13Q1_NDVI_49_2000_353_2015_RData.RData)

, where:

- ***vrt_type*** indicates the type of virtual file ("RData", "GDAL" or "ENVI_META");
- ***Sensor*** indicates to which MODIS sensor the time series belongs ("Terra", "Aqua", "Mixed" or "Combined" (for MCD* products));
- ***Layer*** is a short name describing the dataset (e.g., b1_Red, NDII, UI);
- ***ProdCode*** is the code name of the MODIS product from which the image was derived (e.g., MOD13Q1);
- ***StartDOY***, ***StartYear***, ***EndDOY*** and ***EndYear*** indicate the temporal extent of the time serie created;
- ***suffix*** indicates the type of virtual file (ENVI, GDAL or RData);
- ***ext*** is the file extension (".vrt" for gdal virtual files, "META" for ENVI meta files or "RData" for `R` raster stacks).

7 Accessing the processed time series from R

Preprocessed MODIS data can be retrieved within `R` either by accessing the single-date raster files, or by loading the saved *RasterStack* objects.

Any single-date image can be accessed by simply opening it with a `raster` command:

```
library(raster)
modistsp_file <- "/my_outfolder/EVI/MOD13Q1_2005_137_EVI.tif"
my_raster <- raster(modistsp_file)
```

`rasterStack` time series containing all the processed data for a given parameter (saved in the "Time Series/RData" subfolder - see here for details) can be opened by:

```
in_virtual_file <- "/my_outfolder/Time_Series/RData/Terra/EVI/MOD13Q1_MYD13Q1_NDVI_49_2000_353_2015_RDa
indata          <- get(load(in_virtual_file))
```

This second option allows accessing the complete data stack and analyzing it using the functionalities for raster/raster time series analysis, extraction and plotting provided for example by the `raster` or `rasterVis` packages.

7.1 Extracting Time Series Data on Areas of Interest

`MODIStsp` provides an efficient function (`MODIStsp\_extract`) for extracting time series data at specific locations. The function takes as input a *RasterStack* virtual object created by `MODIStsp` (see above), the starting and ending dates for the extraction and a standard _Sp*_ object (or an ESRI shapefile name) specifying the locations (points, lines or polygons) of interest, and provides as output a `xts` object or `data.frame` containing time series data for those locations.

If the input is of class *SpatialPoints*, the output object contains one column for each point specified, and one row for each date. If it is of class *SpatialPolygons* (or *SpatialLines*), it contains one column for each polygon (or each line), with values obtained applying the function specified as the "FUN" argument (e.g., mean, standard deviation, etc.) on pixels belonging to the polygon (or touched by the line), and one row for each date.

As an example the following code:

```
#Set the input paths to raster and shape file
infile    <- 'myoutfolder/Time_Series/RData/Mixed/MOD13Q1_MYD13Q1_NDVI_49_2000_353_2015_RData.RData'
shpname   <- 'path_to_file/rois.shp'
#Set the start/end dates for extraction
startdate <- as.Date("2010-01-01")
enddate   <- as.Date("2014-12-31")
#Load the RasterStack
inrts     <- get(load(infile))
# Compute average and St.dev
dataavg   <- MODIStsp_extract(inrts, shpname, startdate, enddate, FUN = 'mean', na.rm = T)
datasd    <- MODIStsp_extract (inrts, shpname, startdate, enddate, FUN = 'sd', na.rm = T)
# Plot average time series for the polygons
plot.xts(dataavg)
```

loads a *RasterStack* object containing 8-days 250 m resolution time series for the 2000-2015 period and extracts time series of average and standard deviation values over the different polygons of a user's selected shapefile on the 2010-2014 period.

8 Problems and Issues

Solutions to some common **installation and processing problems** can be found in MODIStsp FAQ:

https://docs.ropensci.org/MODIStsp/articles/faq.html

- Please **report any issues** you may encounter in our issues page on GitHub:

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

9 Citation

To cite MODIStsp please use:

L. Busetto, L. Ranghetti (2016) MODIStsp: An R package for automatic preprocessing of MODIS Land Products time series, Computers & Geosciences, Volume 97, Pages 40-48, ISSN 0098-3004, https://doi.org/10.1016/j.cageo.2016.08.020, URL: https://github.com/ropensci/MODIStsp.

10 Installing R and GDAL

10.1 Installing R

10.1.1 Windows

Download and install the latest version of R which can be found here.

10.1.2 Linux

Please refer to the documentation which can be found here, opening the directory relative to your Linux distribution. The documentation provides instruction to add CRAN repositories and to install the latest R version. With Ubuntu 15.10 Wily (and newer) this step is not mandatory (although recommended), since packaged version of R is >= 3.2.1 (although not the latest); in this case, user can install R by simply typing in a terminal

```
sudo apt-get install r-base
```

10.2 Installing GDAL >= 2.3

10.2.1 Windows

Starting from MODIStsp 1.4.0, no external GDAL installation is needed - MODIStsp exploits the gdalUtilities package to access GDAL functionality through the sf::gdal_utils() function.

10.2.2 Debian and Ubuntu-based systems

1. Ensure that your repositories contain a version of `gdal-bin` >= 1.11.1. In particular, official repositories of Ubuntu 15.04 Vivid (or older) and Debian Jessie (or older) provide older versions of GDAL, so it is necessary to add UbuntuGIS-unstable repository before installing. To do this, follow instructions here). With Ubuntu 15.10 Wily (and newer) this step is not mandatory, although recommended in order to have updated version of GDAL installed.
2. To install GDAL, open a terminal and type
   `bash  sudo apt-get install gdal-bin`

10.2.3 ArchLinux

GDAL is maintained updated to the latest version as binary package within the community repository; although that, the support for HDF4 format is not included. To bypass this problem, ArchLinux users can install `gdal-hdf4` package from AUR (see here or here for the package installation from AUR). This package is updated manually after each release of `gdal` on the community repository, so a temporal shift between a new `gdal` release and the update of `gdal-hdf4` could happen. If you want to manually add the support for HDF4 in case `gdal-hdf4` is out-of-date, you can do it following these instructions.

10.2.4 Other Linux systems

Install the packaged binary of GDAL included in your specific distribution; if the version is older than 1.11.1, or if the support for HDF4 format is not included, you can manually install the HDF4 library and compile the source code by adding the parameter `--with-hdf4` to the `configure` instruction).

10.2.4.1 MacOs

1. Check if gdal is already installed and has hdf4 support. In a terminal, type:

```
~$ gdal-config --formats
```

- if gdal is installed, the command will list which drivers are installed. If the list includes “hdf4” you should be good to go.

- if gdal is not yet installed or hdf4 is not already supported, you can install/reinstall it following these notes. In short: open a terminal and run:

```
~$ brew install hdf4
~$ brew link --overwrite hdf4
~$ brew install gdal --complete --enable-unsupported --with-hdf4
```

then, check again to see if hdf4 is supported. The list should now include hdf4:

```
~$ gdal-config --formats
```

Good luck!

References

Boschetti, M., F. Nutini, G. Manfron, P. A. Brivio, and A. Nelson. 2014. "Comparative analysis of normalised difference spectral indices derived from MODIS for detecting surface water in flooded rice cropping systems." *PLoS ONE* 9 (2). https://doi.org/10.1371/journal.pone.0088741.

Gamon, J. A., and J. S. Surfus. 1999. "Assessing leaf pigment content and activity with a reflectometer." *New Phytologis* 143 (1): 105–17.

Gitelson, A. A., and M. N. Merzlyak. 1998. "Remote sensing of chlorophyll concentration in higher plant leaves." *Advances in Space Research* 22 (5): 689–92. https://doi.org/10.1016/S0273-1177(97)01133-2.

Hall, Dorothy K, George A Riggs, Vincent V Salomonson, Nicolo E DiGirolamo, and Klaus J Bayr. 2002. "MODIS snow-cover products." *Remote Sensing of Environment* 83 (1-2): 181–94. https://doi.org/10.1016/S0034-4257(02)00095-0.

Hengl, T. 2010. "Download and resampling of MODIS images." http://spatial-analyst.net/wiki/index.php?title=Download_and_resampling_of_MODIS_images.

Huete, A., K. Didan, T. Miura, E. P. Rodriguez, X. Gao, and L. G. Ferreira. 2002. "Overview of the radiometric and biophysical performance of the MODIS vegetation indices." *Remote Sensing of Environment* 83 (1-2): 195–213. https://doi.org/10.1016/S0034-4257(02)00096-2.

Huete, A. R. 1988. "A soil-adjusted vegetation index (SAVI)." *Remote Sensing of Environment* 25 (3): 295–309. https://doi.org/10.1016/0034-4257(88)90106-X.

Hunt, J. R., and B. Rock. 1989. "Detection of changes in leaf water content using Near- and Middle-Infrared reflectances." *Remote Sensing of Environment* 30 (1): 43–54. https://doi.org/10.1016/0034-4257(89)90046-1.

Naimi, B. 2014. "ModisDownload: an R function to download, mosaic, and reproject the MODIS images." http://r-gis.net/?q=ModisDownload.

Rouse, J. W. J., R. H. Haas, J. A. Schell, and D. W. Deering. 1973. "Monitoring vegetation systems in the Great Plains with ERTS. Third ERTS Symposium, NASA SP-351. U.S. Gov. Printing office." Edited by S. C. Freden, E. P. Mercanti, and M. A. Becker. NASA.

Tucker, C. J. 1979. "Red and photographic infrared linear combinations for monitoring vegetation." *Remote Sensing of Environment* 8 (2): 127–50. https://doi.org/10.1016/0034-4257(79)90013-0.