Package 'RGISTools'

October 12, 2022

Type Package

Title Handling Multiplatform Satellite Images

Version 1.0.2

Author U Pérez - Goya [aut, cre] <unai.perez@unavarra.es>,
M Montesino - SanMartin [aut] <manuel.montesino@unavarra.es>,
A F Militino [aut] <militino@unavarra.es>,
M D Ugarte [aut] <lola@unavarra.es>

Maintainer U Perez - Goya <unai.perez@unavarra.es>

Description Downloading, customizing, and processing time series of satellite images for a region of interest. 'RGISTools' functions allow a unified access to multispectral images from Landsat, MODIS and Sentinel repositories. 'RGISTools' also offers capabilities for customizing satellite images, such as tile mosaicking, image cropping and new variables computation. Finally, 'RGISTools' covers the processing, including cloud masking, compositing and gap-filling/smoothing time series of images (Militino et al., 2018 <doi:10.3390/rs10030398> and Militino et al., 2019 <doi:10.1109/TGRS.2019.2904193>).

URL https://github.com/spatialstatisticsupna/RGISTools#RGISTools

BugReports https://github.com/spatialstatisticsupna/RGISTools/issues

Depends R (>= 3.5.0), raster, sf, tmap

Imports Rdpack, XML, xml2, utils, fields, urltools, rjson, rvest, tools, methods, curl, httr, mapview, sp, stars

Suggests rgdal

RdMacros Rdpack

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-05-20 14:20:06 UTC

R topics documented:

`RGISTools-package` *'RGISTools': Handling Multiplatform Satellite Images*

Description

This package enables you downloading, customizing, and processing time series of satellite images from Landsat, MODIS and Sentinel in a standardized way. Some functions download and convert automatically the platform-specific file formats into GTiff, so they can be loaded in 'R'. The customization functions support tile mosaicking, cropping, and deriving new variables of interest, such as the normalized difference vegetation index (NDVI), enhanced vegetation index (EVI), etc. Tile mosaicking is required when the region of interest extends over several tiles, so they can be combined into a single image. Cropping involves removing the pixels outside the region of interest, making any analysis more computationally and memory efficient. Processing involves cloud mosaicking, compositing and filling/smoothing satellite data. Cloud masking eliminates the pixel values corresponding to clouds. Cloud removal and (measurement or processing) errors trigger data gaps and outliers, decreasing the quality and quantity of measurements. Hence, the package includes a set of functions for filling and smoothing the satellite imagery. The combination of functions in 'RGISTools' results in a stack of satellite images ready-to-use. Due to the wide variety of procedures and sources of information being handled in 'RGISTools', the functions are divided into 7 categories, which are identified by the first 3 characters of the function names;

1. `mod` identifies MODIS Terra and Aqua satellite functions.
2. `sen` identifies Sentinel functions.
3. `ls7` identifies Landsat-7 functions.
4. `ls8` identifies Landsat-8 functions.
5. `ls` identifies both Landsat-7 and Landsat-8 functions.

6. gen identifies function for being used in any of the three platforms.
7. var identifies function for deriving variables in any of the three platforms.

Details

Below, there is a list of the most important functions grouped by satellite programs, and listed in operational order. These functions include searching, previewing, downloading, mosaicking, deriving new variables, compositing, cloud masking and filling/smoothing satellite imagery.

I. Landsat functions

The Landsat program is currently releasing imagery captured by two satellites; the Landsat-7 and Lansat-8. The functions for both satellites are separate due to discrepancies in their spectral coverage and data formats. To download Landsat imagery with the following functions, a USGS's 'EarthExplorer' account is required. Please, register here.

Landsat-7:

ls7LoadMetadata	Loads the Landsat-7 metadata file
ls7Search	Searches a time series of Landsat-7 images
lsPreview	Previews Landsat satellite images
lsDownload	Downloads a time series of Landsat images
lsCloudMask	Creates clouds masks for Landsat images
lsMosaic	Mosaics Landsat images
ls7FolderToVar	Computes new variables from Landsat-7 multispectral images
genSaveTSRData	Saves a time series of images

Landsat-8:

ls8LoadMetadata	Loads the Landsat-8 metadata file
ls8Search	Searches a time series of Landsat-8 images
lsPreview	Previews Landsat satellite images
lsDownload	Downloads a time series of Landsat images
lsCloudMask	Creates cloud masks for Landsat images
lsMosaic	Mosaics Landsat images
ls8FolderToVar	Computes new variables from Landsat-8 multispectral images
genSaveTSRData	Saves a time series of images

II. MODIS functions

Functions in 'RGISTools' download all land products from Terra and Aqua satellites. However, the processing focuses on the multispectral images. Be aware that an 'EarthData' account is required to use NASA's web service so, please, register here.

modSearch	Searches a time series of MODIS images

modPreview	Previews MODIS satellite images
modDownload	Downloads a time series of MODIS images
modMosaic	Mosaics MODIS images from the land products
modFolderToVar	Computes new variables from MODIS multispectral images
modCloudMask	Creates cloud masks for MODIS images
genSaveTSRData	Saves a time series of images

III. Sentinel functions

Sentinel archives provide a wide variety of products based on a 5-satellite constellation. The functions to download Sentinel images can cope with any product provided by ESA's 'SciHub' web service. However, image processing is focused on Sentinel-2 multispectal images. 'SciHub' credentials are required to download Sentinel imagery and can be obtained here.

senSearch	Searches a time series of Sentinel images
senPreview	Previews Sentinel images
senDownload	Downloads a time series of Sentinel images
senMosaic	Mosaics Sentinel-2 images
senCloudMask	Creates cloud masks for Sentinel-2 images
senFolderToVar	Computes new variables from Sentinel-2 multispectral images
genSaveTSRData	Saves a time series of images

IV. Important general functions

In addition to functions above, the package provides some general functions for a better data handling and processing:

genCompositions	Creates compositions of images from a time series of satellite images
genSmoothingIMA	Fills the gaps and smooths outliers in a time series of satellite images
genSmoothingCovIMA	Fills the gaps and smooths outliers in a time series of satellite images using covariates
genPlotGIS	Plots satellite images with a proper GIS format
genGetDates	Gets the capturing date of an image from the name of a `RasterLayer`

V. Remote sensing variables

New variables can be derived from multispectral images. The most common variables in the scientific literature are pre-programmed in 'RGISTools'. They can be identified by the prefix "var".

varEVI	Calculates the enhanced vegetation index (EVI)
varMSAVI2	Calculates the modified soil-adjusted vegetation index (MSAVI2)
varNBR	Calculates the normalized burn ratio (NBR)
varNBR2	Calculates the normalized burn ratio 2 (NBR2)
varNDMI	Calculates the normalized difference moisture index (NDMI)

varNDVI	Calculates the normalized difference vegetation index (NDVI)
varNDWI	Calculates the normalized difference water index (NDWI)
varRGB	Generates a Red-Green-Blue (RGB) image
varSAVI	Calculates the soil-adjusted vegetation index (SAVI)

`ex.dem.navarre` *A Digital Elevation Model (DEM) of the region of Navarre (Spain)*

Description

Geographically projected `RasterStack` with the digital elevation model (DEM) of the region of Navarre (Spain). The DEM was obtained from the National Center for Geographic Information of Spain. The DEM is used as a covariable in the Image Mean Anomaly (IMA) algorithm (`genSmoothingCovIMA`).

Format

The `RasterStack` contains 6 layers with the same DEM, one for every image in the time series of the `genSmoothingCovIMA` example. The `RasterStack` coordinates are in the Sinusoidal projection:

name layer names contain the capturing date of the corresponding image in the format "`YYYYJJJ`".

size 113 rows by 105 columns and 6 layers.

`ex.navarre` *A polygon with the border of Navarre (Spain)*

Description

Spatial feature (`sf`) representing the border of Navarre with coordinates in the longitude/latitude format.

`ex.ndvi.navarre` *A time series of NDVI of Navarre (Spain)*

Description

Geographically projected `RasterBrick` object of the normalized difference vegetation index (NDVI) of Navarre.

Format

The `RasterBrick` contains 6 images, from the 2nd to the 4th of August in 2017 and 2018. The `RasterBrick` coordinates are in the Sinusoidal projection:

name layer names contain the date of the image in the format "`YYYYJJJ`".

size each layer contains 113 rows and 105 columns.

genCompositions *Create image compositions from a time series of satellite images*

Description

genCompositions combines a series of satellite images to create compositions.

Usage

```
genCompositions(rstack, by, fun, n, by.days = FALSE, verbose = FALSE, ...)
```

Arguments

rstack	a RasterStack, where layer names contain the capturing date of an image in "YYYYJJJ" format.
by	character argument. Accepts "month" or "year" for creating monthly or yearly composites. Only required if n is provided.
fun	the function used to create the composite, such as max, min, mean, ...
n	number of images combined in the aggregation. Only required if by is not provided.
by.days	logical argument. If FALSE, n indicates the number of consucutive images being aggregated. If TRUE, the function aggregates the imagery within every n days. The aggregation requires at least one image avaiable.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions:

- AppRoot the path where the images will be saved in the GTiff format.

Details

The layer of the composite image takes its name from the first image used in the composition.

genCompositions reduces the number of images but improves the total quality of the time-series by removing cloulds and outliers. One widespread compositing technique is the maximum value composition (MVC). This technique allocates in each pixel of the composite the maximum value (fun = max) that the pixel reaches during a time period (n, by.days = TRUE).

Value

a RasterStack with the time series of the composite images.

Examples

```
# loading NDVI images of Navarre
data("ex.ndvi.navarre")
# Ploting the images: clouds are found
genPlotGIS(ex.ndvi.navarre)
# the first composite image is made with images 1, 2 and 3,
# and the second composite image is made with images 4, 5 and 6
composite.NDVI.a <- genCompositions(rstack = ex.ndvi.navarre,
                                    n = 3,
                                    fun = max)
genPlotGIS(composite.NDVI.a)
# when by.days=TRUE, the first composite image is made with images 1, 2 and 3,
# the second with image 4, and the third with images 5 and 6.
composite.NDVI.3a <- genCompositions(rstack = ex.ndvi.navarre,
                                     n = 3,
                                     by.days = TRUE,
                                     fun = max)
# Check that the clouds were removed
genPlotGIS(composite.NDVI.3a)
```

---

genFilterStack *Subset a RasterStack given a range of dates*

Description

genFilterStack filters the RasterLayers within a range of dates.

Usage

```
genFilterStack(r, ...)
```

Arguments

r	the RasterStack to be filtered.
...	arguments for nested functions:

- startDate a Date class object with the starting date of the study period.
- endDate a Date class object with the ending date of the study period.
- AppRoot the path where the RData will be saved.

Details

This is a helper function used by other functions in this package.

Value

a RasterStack with images within the specified dates.

Examples

```
# generate random images
img <- matrix(1:16, ncol = 4, byrow = TRUE)
r <- raster(img)
r <- stack(r, r, r, r, r, r)
names(r) <- paste0("RandomImage_201803", 1:6)
# print the names and dates of the random images
print(names(r))
genGetDates(names(r))
# example of filtering the raster stack
r2 <- genFilterStack(r = r,
                     startDate = as.Date("2018-02-02", "%Y-%m-%d"),
                     endDate = as.Date("2018-02-04", "%Y-%m-%d"))
# print the names and the number of layers of the filtered stack
genGetDates(names(r2))
nlayers(r2)
```

---

genGetDates *Return the capturing date from the name of a raster layer*

Description

genGetDates extracts the date of one or several images when the name of the layer includes the date in the "YYYYJJJ" format.

Usage

```
genGetDates(str, ...)
```

Arguments

str	character containing the date as "YYYYJJJ", where Y and J are year and julian day digits.
...	arguments for nested functions:

- format the format of the date being returned.

Details

The function reads a date from a character class object in year-julian ("YYYYJJJ") format and returns a Date class object.

Value

a Date class object with the date of the image or character class, if format argument is used.

Examples

```
img <- matrix(1:16, ncol = 4, byrow = TRUE)
r <- raster(img)
names(r) <- c("RandomImage_2018034")

spplot(r)
genGetDates(names(r), format = "%Y%j")
```

---

genMosaicList *Mosaic a list of raster images*

Description

genMosaicList makes a single mosaic from a list of raster images.

Usage

```
genMosaicList(imageList, fun = "max", verbose = TRUE)
```

Arguments

imageList	list of RasterLayers.
fun	the function being applied to pixels that overlap.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.

Details

This is a helper function used by other functions in this package. It combines a list of raster images with different geolocations. If images overlap, the function applies fun to calculate the new values for the overlapping pixels. If not specified, fun = max.

Value

a raster with the mosaicked images.

Examples

```
# create simulated rasters
img <- matrix(1:16, ncol = 4, byrow = TRUE)
r1 <- raster(img)
r2 <- r1
r3 <- r1
extent(r2) <- extent(1, 2, 1, 2)
extent(r3) <- extent(1, 2, 0, 1)
imageList <- list(r1, r2, r3)
# mosaic simulated rasters
mr <- genMosaicList(imageList)
spplot(mr)
```

genPlotGIS *Plot satellite images with a GIS format*

Description

This function displays satellite images with the usual format in geographic information systems (GIS), i.e., adding a scale, north arrow, and the border of the region of interest (optional).

Usage

```
genPlotGIS(
  r,
  region,
  breaks,
  labels,
  zlim,
  layout,
  proj,
  nbreaks = 40,
  nlabels = 10,
  as.grid = TRUE,
  compass.rm = FALSE,
  scale.bar.rm = FALSE,
  ...
)
```

Arguments

r	a Raster* class object with an image or stack of images to be plotted. If r is a list of RasterStack, genPlotGIS treates the stacks as RGB images.
region	a Spatial*, projected raster*, or sf class object defining the area of interest.
breaks	a numeric vector defining the color breaks of the legend.
labels	a character vector defining the labels in the breaks of the legend.
zlim	a numeric vector defining the maximun and minimun pixel values to be mapped.
layout	a numeric vector defining rows and columns to divide the plotting area.
proj	a character or 'CRS' class object defining the coordinate reference system of the plot.
nbreaks	a numeric argument defining the default number of breaks.
nlabels	a numeric argument defining the default number of labels in the legend.
as.grid	a logical argument. If TRUE, removes the space between plotted layers.
compass.rm	a logical argument to remove the compass from the plot. FALSE by default.
scale.bar.rm	a logical argument to remove the scale bat from the plot. FALSE by default.
...	argument for nested functions:

- tm_layout any argument accepted by the tm_layout function.
- tm.graticules any argument accepted by the tm_graticules function. The arguments are defined as tm.graticules.arg, where arg is the tm_graticules argument name. For example, the labels.size of tm_graticules is defined as tm.graticules.labels.size.
- tm.compass any argument accepted by the tm_compass function. The arguments are defined as tm.compass.arg, where arg is the tm_compass argument name. For example, the type of tm_compass is defined as tm.compass.type.
- tm.scale.bar any argument accepted by the tm_scale_bar function. The arguments are defined as tm.scale.bar.arg, where arg is the tm_scale_bar argument name. For example, the text.size of tm_scale_bar is defined as tm.scale.bar.text.size.
- tm.shape and tm.polygon refer to the region argument. Any argument accepted by the tm_shape and tm_polygon functions. The arguments are defined as tm.shape.region.arg or tm.polygon.region.arg, where arg is the tm_shape and tm_polygon argument name respectively. For example, the lwd of tm_polygon is defined as tm.polygon.region.lwd.
- tm.shape and tm.raster refer to the r argument. Any argument accepted by the tm_shape and tm_raster functions. The arguments are defined as tm.shape.r.arg or tm.raster.r.arg, where arg is the tm_shape and tm_raster argument name respectively. For example, the legend.reverse of tm_raster is defined as tm.raster.r.legend.reverse.
- tm.tmap.arrange any argument accepted by the tm_tmap_arrange function. The arguments are defined as tm.tmap.arrange.arg, where arg is the tm_tmap_arrange argument name. For example, the asp of tm_tmap_arrange is defined as tm.tmap.arrange.asp. This arguments are only accepted when plotting a list of stack images to plot as RGB.

Details

This is a wrapper function of tmap and hence displays any Raster* object and accepts all of its parameters. The function adds a scale, a north arrow and a polygon in the area of interest. If necessary, the function automatically reprojects the polygon to match the projection of the raster. The projection of the map can be changed by modifying the proj argument. For futher help on tmap arguements, please go the tmap reference manual.

Value

tmap class containing the plot.

Examples

```
## Not run:
# Simple plot of NDVI in Navarre
genPlotGIS(ex.ndvi.navarre,
           ex.navarre)

# Using tm arguments
genPlotGIS(ex.ndvi.navarre,
```

```
            ex.navarre,
            tm.compass.size=1,
            tm.compass.type="rose",
            tm.scale.bar.text.size=0.8,
            tm.polygon.region.lwd=6,
            tm.polygon.region.border.col="#000000",
            tm.raster.r.palette=rev(terrain.colors(40)),
            tm.raster.r.title="NDVI",
            as.grid = TRUE,
            tm.graticules.labels.size=1,
            tm.graticules.n.x=3,
            tm.graticules.n.y=3)
# Using the view mode of tmap for ploting the images in the viewer
tmap_mode("view")
genPlotGIS(ex.ndvi.navarre,
           ex.navarre,
           tm.raster.r.palette=rev(terrain.colors(40)))+
           tm_facets(as.layers=TRUE)

# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the red, blue and NIR bands
img.mod.red <- raster(files.mod[1])
img.mod.blue <- raster(files.mod[3])
img.mod.green <- raster(files.mod[4])
img.mod.rgb<-varRGB(img.mod.red,img.mod.green,img.mod.blue)
genPlotGIS(ex.ndvi.navarre,
           ex.navarre)+
           tm_facets(as.layers = TRUE)+
genPlotGIS(list(img.mod.rgb),
           ex.navarre)

## End(Not run)
```

---

genSaveTSRData *Saves a time series of images as an RData*

Description

genSaveTSRData imports a time series of images from a folder (GTiff format), builds a RasterStack and saves it in an RData.

Usage

```
genSaveTSRData(
```

```
  src,
  AppRoot = NULL,
  ts.name = "TS.Name",
  startDate = NULL,
  endDate = NULL,
  dextent = FALSE,
  recursive = FALSE
)
```

Arguments

src	path to the folder where the time series of images is located.
AppRoot	the path where the RData is saved.
ts.name	the name of the RasterStack in the RData.
startDate	a Date class object with the starting date of the study period.
endDate	a Date class object with the ending date of the study period.
dextent	a logical argument. If TRUE, the function expands the extent of the RasterStack to embrace the extents of all GTiff images.
recursive	logical argument. If TRUE, reads folders recursively, searching for GTiff images.

Details

The function reads all the images inside the folder specified in src. Images files must be GTiffs. The src can take the path created by other functions of this package, such as senMosaic, modMosaic, senFolderToVar, etc. The images are imported into 'R' to build a RasterStack. The name of the RasterStack is specified in ts.name. The RasterStack is saved in an RData file in the AppRoot directory.

Value

a RasterStack when the AppRoot argument is not defined. The function does not return anything otherwise.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# set the download folder
s.start <- Sys.time()
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download the images
modDownSearch(product = "MOD09GA",
            startDate = as.Date("30-07-2018", "%d-%m-%Y"),
            endDate = as.Date("06-08-2018", "%d-%m-%Y"),
            username = "username",
            password = "password",
            AppRoot = wdir,
```

```
            extract.tif = TRUE,
            collection = 6,
            extent = ex.navarre)
# set folder path where MOD09GA images will be saved
wdir.mod <- file.path(wdir,"Modis","MOD09GA")
# set the tif folder path
wdir.mod.tif <- file.path(wdir.mod,"tif")
# mosaic and cut navarre region
modMosaic(wdir.mod.tif,
          AppRoot = wdir.mod,
          out.name = "Navarre",
          extent = ex.navarre)
# change src to navarre folder
wdir.mod.navarre <- file.path(wdir.mod,"Navarre")
# calculate NDVI from navarre folder
modFolderToVar(wdir.mod.navarre,
               fun = varNDVI,
               AppRoot = dirname(wdir.mod.navarre),
               overwrite = TRUE)
# change src TS_sample
wdir.mod.ndvi <- file.path(dirname(wdir.mod.navarre),"NDVI")
# create the Rdata
tiles.mod.ndvi<-genSaveTSRData(wdir.mod.ndvi, ts.name = "ModisNDVI")
# remove values out of 0-1 range
tiles.mod.ndvi.lim <- clamp(tiles.mod.ndvi,lower=0,upper=1)
# plot the ndvi images
spplot(tiles.mod.ndvi.lim)
s.end <- Sys.time()
s.end - s.start

## End(Not run)
```

`genSmoothingCovIMA` *Fill data gaps and smooth outliers in a time series of satellite images using covariates*

Description

`genSmoothingCovIMA` runs the image mean anomaly (IMA) algorithm with covariates (Militino et al. 2018).

Usage

```
genSmoothingCovIMA(
  rStack,
  cStack,
  Img2Process = NULL,
  nDays = 3,
  nYears = 1,
```

```
  r.dates,
  fact = 5,
  fun = mean,
  aFilter = c(0.05, 0.95),
  snow.mode = FALSE,
  out.name = "out",
  ...
)
```

Arguments

rStack	a RasterStack class argument containing a time series of satellite images. Layer names should contain the date of the image in "YYYYJJJ" format.
cStack	a RasterStack class argument containing a time series of covariates.
Img2Process	a vector class argument defining the images to be filled/smoothed.
nDays	a numeric argument with the number of previous and subsequent days that define the temporal neighborhood.
nYears	a numeric argument with the number of previous and subsequent years that define the temporal neighborhood.
r.dates	a vector argument containing the dates of the layers in rstack
fact	a numeric argument with an aggregation factor of the anomalies carried out before the interpolation.
fun	a function used to aggregate the image of anomalies. Both mean(default) or median are acceptted.
aFilter	a vector with the lower and upper quantiles that define the outliers of the anomalies. Ex. c(0.05,0.95).
snow.mode	logical argument. If TRUE, the filling process will be parallelized using the 'raster' package.
out.name	the name of the folder containing the filled/smoothed images when saved in the Hard Disk Drive (HDD).
...	arguments for nested functions: • AppRoot the path where the filled/smoothed time series of images are saved as GTiff.

Details

This filling/smoothing method was developed by Militino et al. (2018). This technique decomposes a time series of images into a new series of mean and anomaly images. The procedure applies the filling/smoothing algorithm with covariates over the anomaly images. The procedure requires a proper definition of a temporal neighbourhood for the target image and aggregation factor.

Value

a RasterStack with the filled/smoothed images.

References

Militino AF, Ugarte MD, Perez-Goya U (2018). "Improving the Quality of Satellite Imagery Based on Ground-Truth Data from Rain Gauge Stations." *Remote Sensing. (Open-Access)*, **10**(398), 1–16. http://dx.doi.org/10.3390/rs10030398.

Examples

```
## Not run:
set.seed(0)
# load example ndvi and dem data of Navarre
data(ex.ndvi.navarre)
data(ex.dem.navarre)
# plot example data
genPlotGIS(ex.ndvi.navarre)
genPlotGIS(ex.dem.navarre)

# distorts 5% of the original ndvi data by
# altering 50% its values
for(x in c(2,5)){
  aux <- sampleRandom(ex.ndvi.navarre[[x]],
                      ncell(ex.ndvi.navarre) * 0.05,
                      cells = TRUE,
                      na.rm = TRUE)
  ex.ndvi.navarre[[x]][aux[,1]] <- aux[,2] * 1.5
}
genPlotGIS(ex.ndvi.navarre)

# smoothing the image using the DEM as covariate
smth.ndvi <- genSmoothingCovIMA(rStack = ex.ndvi.navarre,
                                cStack = ex.dem.navarre,
                                Img2Process = c(2))
# plot the distorted 1, smoothed 1,
# distorted 5, smoothed 5 images
plot(stack(ex.ndvi.navarre[[2]],
                 smth.ndvi[[1]],
                 ex.ndvi.navarre[[5]],
                 smth.ndvi[[2]]))

## End(Not run)
```

---

genSmoothingIMA *Fill data gaps and smooth outliers in a time series of satellite images*

Description

genSmoothingIMA is the implementation of a spatio temporal method called image mean anomaly (IMA) for gap filling and smoothing satellite data (Militino et al. 2019).

Usage

```
genSmoothingIMA(
  rStack,
  Img2Fill = NULL,
  nDays = 3,
  nYears = 1,
  fact = 5,
  fun = mean,
  r.dates,
  aFilter = c(0.05, 0.95),
  only.na = FALSE,
  factSE = 8,
  predictSE = FALSE,
  snow.mode = FALSE,
  out.name = "outname",
  ...
)
```

Arguments

rStack	a RasterStack class argument containing a time series of satellite images. Layer names should contain the date of the image in "YYYYJJJ" format.
Img2Fill	a vector argument defining the images to be filled/smoothed.
nDays	a numeric argument with the number of previous and subsequent days that define the temporal neighborhood.
nYears	a numeric argument with the number of previous and subsequent years that define the temporal neighborhood.
fact	a numeric argument with an aggregation factor of the anomalies before the interpolation.
fun	a function used to aggregate the image of anomalies. Both mean (default) or median are acceptted.
r.dates	a vector argument containing the dates of the layers in rstack
aFilter	a vector with the lower and upper quantiles that define the outliers of the anomalies. Ex. c(0.05,0.95).
only.na	logical argument. If TRUE only fills the NA values. FALSE by default.
factSE	the fact used in the standard error prediction.
predictSE	calculate the standard error instead the prediction.
snow.mode	logical argument. If TRUE, the filling process will be parallelized using the 'raster' package.
out.name	the name of the folder containing the smoothed/filled images when saved in the Hard Disk Device (HDD).
...	arguments for nested functions: • AppRoot the path where the filled/smoothed time series of images will be saved in GTiff format.

Details

This filling/smoothing method was developed by Militino et al. (2019). This technique decomposes a time series of images into a new series of mean and anomaly images. The procedure applies the smoothing algorithm over the anomaly images. The procedure requires a proper definition of a temporal neighbourhood for the target image and aggregation factor.

Value

a `RasterStack` with the filled/smoothed images.

References

Militino AF, Ugarte MD, Perez-Goya U, Genton MG (2019). "Interpolation of the Mean Anomalies for Cloud-Filling in Land Surface Temperature (LST) and Normalized Difference Vegetation Index (NDVI)." *IEEE Transactions on Geoscience and Remote Sensing. (Open-Access).* http://dx.doi.org/10.1109/TGRS.2019.2904193.

Examples

```
## Not run:
# load an example of NDVI time series in Navarre
data(ex.ndvi.navarre)
# the 2 images to be filled and the neighbourhood
genPlotGIS(ex.ndvi.navarre)

# filled images
tiles.mod.ndvi.filled  <- genSmoothingIMA(ex.ndvi.navarre,
                               Img2Fill = c(1),
                               only.na=TRUE)
# show the filled images
genPlotGIS(tiles.mod.ndvi.filled)
# plot comparison of the cloud and the filled images
tiles.mod.ndvi.comp <- stack(ex.ndvi.navarre[[1]], tiles.mod.ndvi.filled[[1]],
                             ex.ndvi.navarre[[2]], tiles.mod.ndvi.filled[[2]])
genPlotGIS(tiles.mod.ndvi.comp, layout=c(2, 2))

## End(Not run)
```

---

getRGISToolsOpt *Get the default value of an RGISTools option*

Description

`getRGISToolsOpt` gets the current value of an 'RGISTools' configuration variable. This function can be jointly used with `setRGISToolsOpt` and `showRGISToolsOpt`.

Usage

```
getRGISToolsOpt(opt, env = optEnv)
```

Arguments

opt	a character with the name of the option.
env	the environment where the 'RGISTools' option are saved.

Value

an option of 'RGISTools' configuration variable.

Examples

```
# list avaliable options names
showRGISToolsOpt()
# list the Sentinel-2 bands
getRGISToolsOpt("SEN2BANDS")
# list the Landsat-8 bands
getRGISToolsOpt("LS8BANDS")
# list the MODIS09 bands
getRGISToolsOpt("MOD09BANDS")
```

---

ls7FolderToVar *Compute a remote sensing index from a time series of Landsat-7 images*

Description

ls7FolderToVar computes a remote sensing index from the spectral bands of a time series of Landsat-7 images. The images are specified by the path to the folder that stores the imagery (resulting from the lsMosaic function). The function returns a RasterStack with the time series of images of the remote sensing index.

Usage

```
ls7FolderToVar(
  src,
  fun,
  AppRoot,
  getStack = FALSE,
  overwrite = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

src	the path to the folder with the Landsat-7 multispectral imagery.
fun	a function that computes the remote sensing index.
AppRoot	the directory of the outcoming time series.

getStack	logical argument. If TRUE, returns the time series as a RasterStack, otherwise the images are saved in the Hard Drive Device (HDD).
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The function requires the definition of the src and fun arguments. The src is usually the path resulting from lsMosaic. The fun argument can be any function from this package beginning with "var" (varNDVI, varEVI, etc.). Custom functions can also be implemented. Caution! It is mandatory to use level-2 products to get accurate derived variables.

Value

this function does not return anything, unless getStack = TRUE which then returns a RasterStack with the time series of with the index.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# main output directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download Landsat-7 images
lsDownSearch(satellite = "ls7",
             username = "username",
             password = "password",
             startDate = as.Date("01-01-2018", "%d-%m-%Y"),
             endDate = as.Date("20-01-2018", "%d-%m-%Y"),
             extent = ex.navarre,
             untar = TRUE,
             AppRoot = wdir)
# folder with the Landsat-7 untared images
wdir.ls7 <-file.path(wdir,"Landsat7")
wdir.ls7.untar <- file.path(wdir.ls7, "untar")
# mosaic the Landsat-7 images
lsMosaic(wdir.ls7.untar,
         AppRoot = wdir.ls7,
         out.name = "Navarre",
         extent = ex.navarre,
         gutils = TRUE)
# folder with the mosaicked images
wdir.ls7.navarre <- file.path(wdir.ls7, "Navarre")
# generate NDVI images of Navarre
wdir.ls7.var <- file.path(wdir.ls7, "Navarre_Variables")
```

```
dir.create(wdir.ls7.var)
ls7FolderToVar(wdir.ls7.navarre,
               fun = varNDVI,
               AppRoot = wdir.ls7.var,
               overwrite = TRUE)

files.ls7.ndvi <- list.files(file.path(wdir.ls7.var,"NDVI"),
                             pattern = "\\.tif$",
                             full.names = TRUE,
                             recursive = TRUE)
img.ls7.ndvi <- raster(files.ls7.ndvi[1])
spplot(img.ls7.ndvi)

## End(Not run)
```

`ls7LoadMetadata` *Load or update the Landsat-7 metadata file*

Description

`ls7LoadMetadata` loads a `data.frame` called "`.LS7MD`" with the names of the Landsat-7 images and their metadata. The metadata provides auxiliary information, such as image quality, acquisition date, cloud cover, etc. You can find a description of the metadata on the USGS's website.

Usage

```
ls7LoadMetadata(
  AppRoot,
  update = FALSE,
  verbose = TRUE,
  omit.question = TRUE,
  ...
)
```

Arguments

`AppRoot`	the directory where the metadata file should be located.
`update`	logical argument. If `TRUE`, updates the metadata file.
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.
`omit.question`	logical argument. If `TRUE`, the question about loading the metadata is omitted.
`...`	arguments for nested functions.

Details

All captures done by Landsat-7 are catalogued and documented in a unique csv file. The size of the file could be larger than 360MB. The function downloads and imports the metadata into 'R', which may take several minutes (roughly 15 minutes in a Intel Core i7-4790, 16Gb of RAM and Hard Drive Device). The function creates an RData file with the csv metadata. Thus, every time `ls7LoadMetadata` is called, this function loads the existing RData from the `AppRoot` directory, which aims to reduce the loading time of the metadata in the future.

Value

this function does not return anything, but loads the ".LS7MD" `data.frame` on the environment of the 'RGISTools' package.

Examples

```
## Not run:
# creates a MetaData folder and downloads the csv in the "Path_for_downloading_folder" directory
ls7LoadMetadata(AppRoot = file.path(tempdir(),"Path_for_downloading_folder"))

# update the metadata file
ls7LoadMetadata(AppRoot = file.path(tempdir(),"Path_for_downloading_folder"), update = TRUE)

# get metadata data frame
LS7MD <- getRGISToolsOpt("LS7METADATA")
head(LS7MD)

## End(Not run)
```

`ls7Search` *Search Landsat-7 images*

Description

`ls7Search` searches Landsat-7 images in the Landsat repository concerning a particular location and date interval. The function returns a `data.frame` with the names of the images and their metadata.

Usage

```
ls7Search(AppRoot, verbose = FALSE, precise = FALSE, ...)
```

Arguments

`AppRoot`	directory of the metadata file.
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.
`precise`	logical argument. If `TRUE`, conducts a thorough search, tile by tile (slower).
`...`	arguments for nested functions:

- dates a vector with the capturing dates being searched. This argument is mandatory if startDate and endDate are not defined.
- startDate a Date class object with the starting date of the study period. This argument is mandatory if dates is not defined.
- endDate a Date class object with the ending date of the study period. This argument is mandatory if dates is not defined.
- region a Spatial*, projected raster*, o r sf class object defining the area of interest. This argument is mandatory if pathrow, extent, or lonlat are not defined.
- pathrow a list of vectors with the path and row numbers of the tiles concerning the region of interest. This argument is mandatory if region, extent or lonlat are not provided. Ex. list(c(200,31),c(200,30)).
- lonlat a vector with the longitude/latitude coordinates of the point of interest. Ex. c(-1.64323,42.81687). This argument is mandatory if region, pathrow, or lonlat are not defined.
- extent an extent, Raster*, or Spatial* object representing the region of interest with longitude/latitude coordinates. This argument is mandatory if region, pathrow or lonlat are not defined.
- column names in the .LS7MD data.frame and their values.

Details

ls7Search searches images in the metadata file. If the metadata was downloaded before to the current directory, ls7Search will use this metadata by default. In case the metadata was not downloaded before, ls7Search will make that call for you. The function creates the following subfolders "Landsat-8/metadata", where the metadata file is located.

Landsat images are organized by tiles, which have a unique path and row numbers according to the Worldide Reference System. The fastest way to search an image in the metadata file is by path and row (pathrow). This method requires to know in advance the path and row numbers of the tile that is relevant for your region of interest. From the user's standpoint, the simplest way to search a time series of Landsat-7 images is by region, extent, or lonlat, since they do not require any prior knowledge about the tiles.

The function can screen the results by any other attribute in the metadata. For instance, to filter the imagery with an available preview, the browseAvaliable="Y" must be added as an argument of the function (see the examples).

Value

a data.frame with the name of the images and their metadata.

Examples

```
## Not run:
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
# search by path and row numbers of a tile
sres <- ls7Search(startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  pathrow = list(c(200,31),c(200,30)),
```

```
                  browseAvaliable = "Y",
                  AppRoot = wdir)

# search by point coordinates (long/lat coordinates)
sres <- ls7Search(startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  lonlat = c(-1.64323,42.81687),
                  browseAvaliable = "Y",
                  AppRoot = wdir)

# search by extent (long/lat coordinates)
# load a spatial polygon object of Navarre
data(ex.navarre)
sres <- ls7Search(startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  extent = ex.navarre,
                  precise = TRUE,
                  browseAvaliable = "Y",
                  AppRoot = wdir)

# search by extent (fast mode)
sres <- ls7Search(startDate = as.Date("01-01-2011","%d-%m-%Y"),
                  endDate = as.Date("31-12-2013","%d-%m-%Y"),
                  extent = ex.navarre,
                  precise = FALSE,
                  browseAvaliable = "Y",
                  AppRoot = wdir)
# remove metadata to free memory space
lsRemoveMetadata()

## End(Not run)
```

`ls8FolderToVar` *Compute a remote sensing index from a time series of Landsat-8 images*

Description

`ls8FolderToVar` computes a remote sensing index from the spectral bands of a time series of Landsat-8 images. The images are specified by the path to the folder that stores the imagery (resulting from the `lsMosaic` function). The function returns a `RasterStack` with a time series of images of the remote sensing index.

Usage

```
ls8FolderToVar(
  src,
  fun,
  AppRoot,
```

```
  getStack = FALSE,
  overwrite = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

src	path to the folder with the Landsat-8 multispectral image.
fun	a function that computes the remote sensing index.
AppRoot	the directory of the outcoming time series.
getStack	logical argument. If TRUE, returns the time series of images as a RasterStack, otherwise the images are saved in the Hard Drive Device (HDD).
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The function requires the definition of the src and fun arguments. The src is usually the path resulting from lsMosaic.The fun argument can be any function from this package beginning with "var" (varNDVI, varEVI, etc.). Custom functions can also be implemented. If fun = varRGB, then the argument getStack must be equal to FALSE and the red-green-blue (RGB) images must be imported afterwards. Caution! It is mandatory to use level-2 products to get accurate derived variables.

Value

this function does not return anything, unless getStack = TRUE which then returns a RasterStack with the time series of with the index.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# main output directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download Landsat-8 images
lsDownSearch(satellite = "ls8",
             username = "username",
             password = "password",
             startDate = as.Date("01-01-2018","%d-%m-%Y"),
             endDate = as.Date("18-01-2018","%d-%m-%Y"),
             pathrow = list(c(200, 31), c(200, 30)),
```

```
            untar = TRUE,
            AppRoot = wdir)
# folder with the Landsat-8 untared images
src.ls8 <-file.path(wdir,"Landsat8")
src.ls8.untar <- file.path(src.ls8, "untar")
# mosaic the Landsat-8 images
lsMosaic(src = src.ls8.untar,
         AppRoot = src.ls8,
         out.name = "Navarre",
         extent = ex.navarre,
         gutils = TRUE)
# path to the folder with mosaicked images
src.ls8.navarre <- file.path(src.ls8, "Navarre")
# generate NDVI images of Navarre
src.ls8.var <- file.path(src.ls8, "Navarre_Variables")
dir.create(src.ls8.var)
ls8FolderToVar(src.ls8.navarre,
               fun = varNDVI,
               AppRoot = src.ls8.var,
               overwrite = TRUE)

files.ls8.ndvi <- list.files(file.path(src.ls8.var,"NDVI"),
                             pattern = "\\.tif$",
                             full.names = TRUE,
                             recursive = TRUE)

img.ls8.ndvi <- raster(files.ls8.ndvi[1])
spplot(img.ls8.ndvi)

## End(Not run)
```

`ls8LoadMetadata` *Load or update the Landsat-8 metadata file*

Description

`ls8LoadMetadata` loads a `data.frame` called "`.LS8MD`" with the names of the Landsat-8 images and their metadata. The metadata provides auxiliary information, such as image quality, acquisition date, cloud cover, etc. You can find a description of the metadata on the USGS's website.

Usage

```
ls8LoadMetadata(
  AppRoot,
  update = FALSE,
  verbose = TRUE,
  omit.question = FALSE,
  ...
)
```

Arguments

AppRoot	the directory where the metadata file should be located.
update	logical argument. If TRUE, updates the metadata file.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
omit.question	logical argument. If TRUE, the question about loading the metadata is omitted.
...	arguments for nested functions.

Details

All captures done by Landsat-8 are catalogued and documented in a unique csv file. The size of the file could be larger than 210MB. The function downloads and imports the metadata into 'R', which may take several minutes (roughly 7 minutes in a Intel Core i7-4790, 16Gb of RAM and Hard Drive Device). The function creates an RData file with the csv metadata. Thus, every time ls8LoadMetadata is called, this function loads the existing RData from the AppRoot directory, which aims to reduce the loading time of the metadata in the future.

Value

this function does not return anything, but loads the ".LS8MD" data.frame on the 'RGISTools' package.

Examples

```
## Not run:

# creates a MetaData folder and downloads the csv in the "Path_for_downloading_folder" directory
ls8LoadMetadata(AppRoot = file.path(tempdir(),"Path_for_downloading_folder"))

# update the metadata file
ls8LoadMetadata(AppRoot = file.path(tempdir(),"Path_for_downloading_folder"), update = TRUE)

# get metadata data frame
LS8MD <- getRGISToolsOpt("LS8METADATA")
head(LS8MD)

## End(Not run)
```

---

ls8Search *Search Landsat-8 images*

Description

ls8Search searches Landsat-8 images in the Landsat repository concerning a particular location and date interval. The function returns a data.frame with the names of the images and their metadata.

Usage

```
ls8Search(AppRoot, verbose = FALSE, precise = FALSE, ...)
```

Arguments

AppRoot	directory of the metadata file.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
precise	logical argument. If TRUE, conducts a thorough search, tile by tile (slower).
...	arguments for nested functions:

- dates a vector with the capturing dates being searched. This argument is mandatory if startDate and endDate are not defined.
- startDate a Date class object with the starting date of the study period. This argument is mandatory if dates is not defined.
- endDate a Date class object with the ending date of the study period. This argument is mandatory if dates is not defined.
- region a Spatial*, projected raster*, or sf class object defining the area of interest. This argument is mandatory if pathrow, extent, or lonlat are not defined.
- pathrow a list of vectors with the path and row numbers of the tiles concerning the region of interest. This argument is mandatory if region, extent, or lonlat are not provided. Ex. list(c(200,31),c(200,30)).
- lonlat a vector with the longitude/latitude coordinates of the point of interest. Ex. c(-1.64323,42.81687).
- extent an extent, Raster*, or Spatial* object representing the region of interest with longitude/latitude coordinates. This argument is mandatory if region, pathrow, or lonlat are not defined.
- column names in the .LS8MD data.frame and their values.

Details

ls8Search searches images in the metadata file. If the metadata was downloaded before to the current directory, ls8Search will use this metadata by default. In case the metadata was not downloaded yet, ls8Search will make that call for you. The function creates the following subfolders "Landsat-8/metadata", where the metadata file is located.

Landsat images are organized by tiles, which have a unique path and row numbers according to the Worldide Reference System. The fastest way to search an image in the metadata file is by path and row (pathrow). This method requires to know in advance the path and row number of the tile that is relevant for your region of interest. From the user's standpoint, the simplest way to search a time series of Landsat-7 images is by region, extent, or lonlat since they do not require any prior knowledge about tiles.

The function can screen the results by any other attribute in the metadata. For instance, to filter the imagery with an available preview, the browseAvaliable="Y" must be added as an argument of the function (see the examples).

Value

a data.frame with the name of the images and their metadata.

Examples

```
## Not run:
# search by path and row numbers of a tile
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
sres <- ls8Search(startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  pathrow = list(c(200, 31), c(200, 30)),
                  browseAvaliable = "Y",
                  AppRoot = wdir)

# search by extent (long/lat coordinates)
# load a spatial polygon object of Navarre
data(ex.navarre)
sres <- ls8Search(startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  extent = ex.navarre,
                  precise = TRUE,
                  browseAvaliable = "Y",
                  AppRoot = wdir)

# search by extent (fast mode)
sres <- ls8Search(startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  extent = ex.navarre,
                  precise = FALSE,
                  browseAvaliable = "Y",
                  AppRoot = wdir)
# remove metadata to free memory space
lsRemoveMetadata()

## End(Not run)
```

`lsCloudMask` *Create cloud masks for Landsat images*

Description

`lsCloudMask` creates a cloud mask derived from the band for quality assurance (BQA) from Landsat-7 or Landsat-8 time series. The function is applied to untarred images, such as those resulting from `lsDownload` or `lsDownSearch`. The result is a new image band, called cloud (CLD), that is saved as separate GoeTiffs.

Usage

```
lsCloudMask(
  src,
  AppRoot,
  out.name,
  ls8 = TRUE,
```

```
  overwrite = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

src	the path to the folder with the untarred images from Landsat-7 or Landsat-8.
AppRoot	the directory where the cloud masks are saved.
out.name	the name of the folder that stores the outputs. If the arguemnt is not defined the folder will be named as "CloudMask".
ls8	logical argument. Defines the lansat satellite number. If TRUE clouds for landsat-8 are computed, otherwise, landsat 4-7 clouds are computed.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions.

- dates a vector with the dates being considered for creating cloud masks. This argument is optional.

Details

The valid range for the sensitivity threshold is 0-80000. By defualt, the argument is set to 28000. We recommend 6000 and 28000 for Landsat-7 and Landsat-8 respectively. The NA and 1 values of the mask represent cloudy and clear-sky pixels pixels respectively.

Value

this function does not return anything. It saves the cloud masks (CLD) as GTiff files in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)

# search and download images from Landsat-8 between
# 01-01-2018 and 20-01-2018 for the region of Navarre
lsDownSearch(satellite = "ls8",
             username = "username",
             password = "password",
             startDate = as.Date("01-01-2018", "%d-%m-%Y"),
             endDate = as.Date("20-01-2018", "%d-%m-%Y"),
             pathrow = list(c(200, 31), c(200, 30)),
             untar = TRUE,
             AppRoot = wdir)
```

```
# define the path where the GTiff images are located
wdir.ls8 <- file.path(wdir,"Landsat8")
wdir.ls8.untar <- file.path(wdir.ls8,"untar")

# mosaic and crop the imagery
lsMosaic(src = wdir.ls8.untar,
         AppRoot = wdir.ls8,
         out.name = "Navarre",
         extent = ex.navarre,
         gutils = TRUE, # using gdalUtils
         overwrite = TRUE) # overwrite

# generate the path where mosaicked images are located
wdir.ls8.navarre <- file.path(wdir.ls8, "Navarre")

# calculate the cloud mask from QC layer
lsCloudMask(src=wdir.ls8.navarre,
            overwrite=TRUE)

# load the B1 layer and calculate the CLD layer
files.ls8.navarre.path <- list.files(wdir.ls8.navarre,
                                     full.names = TRUE,
                                     recursive = TRUE,
                                     pattern = "\\.tif$")
tiles.ls8.cld <- files.ls8.navarre.path[grepl("CLD",files.ls8.navarre.path)]
tiles.ls8.b1 <- files.ls8.navarre.path[grepl("B1.tif",files.ls8.navarre.path)]
img.ls8.cld <- lapply(tiles.ls8.cld,raster)
img.ls8.b1 <- lapply(tiles.ls8.b1,raster)

# calculate cloud free b1 layers
img.ls8.b1.cloud.free <- img.ls8.b1[[1]] * img.ls8.cld[[1]]
spplot(img.ls8.b1.cloud.free)

## End(Not run)
```

`lsDownload` *Download Landsat-7 or Landsat-8 images from a search list*

Description

`lsDownload` downloads the results from `ls7Search` and `ls8Search` functions. The images are saved as GTiff files in the `AppRoot` directory.

Usage

```
lsDownload(
  searchres,
  username = NULL,
  password = NULL,
  AppRoot,
```

```
  lvl = 1,
  product = c("sr", "source_metadata"),
  verbose = FALSE,
  raw.rm = FALSE,
  untar = FALSE,
  overwrite = FALSE,
  nattempts = 5,
  l2rqname,
  ...
)
```

Arguments

searchres	the results from ls7Search or ls8Search.
username	USGS's 'EarthExplorer' username.
password	USGS's 'EarthExplorer' password.
AppRoot	the download directory.
lvl	a number specifying the processing level. Default value, 1.
product	character vector with the requested Level-2 products. By default c("sr","source_metadata").
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
raw.rm	logical argument. If TRUE, removes the raw images.
untar	logical argument. If TRUE, untars downloaded images.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
nattempts	the number of attempts to download an image in case it becomes corrupted.
l2rqname	character argument defining the name of the order for level 2 products.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for downloading.
- bFilter a vector with the bands to be extracted when untar=TRUE. If not supplied, all bands are extracted.

Details

lsDonwSearch downloads the list of URLs provided by ls7Search or ls8Search as a data.frame. The function requires an USGS's 'EarthExplorer' account, which can be obtained here.

The files from 'EarthExplorer' are compressed as 'tar.gz'. lsDownload decompresses the images and obtains the corresponding GTiffs. The GTiff files are saved in the AppRoot directory. To change this option, provide AppRoot = "full path". When untar = TRUE, the function decompresses the imagery. When only a subset of bands is required, band names can be provided through the bFilter argument. The band names are specified by string "band" and the band number (e.g., "band1"). Image decompression duplicates the information due to the presence of both, compressed and decompressed images. Set raw.rm = TRUE to remove former ones.

Value

this function does not return anything. It saves the imagery as 'tar.gz' (and GTiff files) in a folder called 'raw' ('untar') in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)

wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)

# search and download the images from Landsat-8 between
# 2011 and 2013 in the region of Navarre
sres <- ls8Search(startDate = as.Date("01-01-2018", "%d-%m-%Y"),
                  endDate = as.Date("20-01-2018", "%d-%m-%Y"),
                  extent = ex.navarre,
                  browseAvaliable = "Y",
                  AppRoot = wdir)

# download 1 image
lsDownload(searchres = sres[1,],
           username = "username",
           password = "password",
           AppRoot = wdir,
           untar = TRUE)
# download 4 images
lsDownload(searchres = sres[1:4,],
           username = "username",
           password = "password",
           AppRoot = wdir,
           untar = TRUE)
# download all the images
lsDownload(searchres = sres,
           username = "username",
           password = "password",
           AppRoot = wdir,
           untar = TRUE)

# search and download the images from Landsat-7 between
# 2011 and 2013 in the region of Navarre
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
sres <- ls7Search(startDate = as.Date("01-01-2018", "%d-%m-%Y"),
                  endDate = as.Date("20-01-2018", "%d-%m-%Y"),
                  extent = ex.navarre,
                  browseAvaliable = "Y",
                  AppRoot = wdir)
# download 1 image
lsDownload(searchres = sres[1,],
           username = "username",
           password = "password",
           untar = TRUE,
           AppRoot = wdir)
# download 4 images
lsDownload(searchres = sres[1:4,],
```

```
            username = "username",
            password = "password",
            untar = TRUE,
            AppRoot = wdir)
# download all the images
lsDownload(searchres = sres,
            username = "username",
            password = "password",
            untar = TRUE,
            AppRoot = wdir)

# removes the metadata to free memory space
lsRemoveMetadata()

# select Landsat-7 RGB bands
wdir.ls7 <- file.path(wdir,"Landsat7")
files.ls7 <- list.files(wdir.ls7,
                        pattern = "\\.TIF$",
                        full.names = TRUE,
                        recursive = TRUE)[c(6,5,4)]
files.ls7.rgb <- stack(files.ls7)
qrange <- c(0.001, 0.999)
img.ls7.rgb <- varRGB(files.ls7.rgb[[1]],
                      files.ls7.rgb[[2]],
                      files.ls7.rgb[[3]],
                      qrange)
plotRGB(img.ls7.rgb)

## End(Not run)
```

`lsDownSearch` *Search and download Landsat-7 or Landsat-8 images*

Description

`lsDownSearch` searches and downloads Landsat-7 or Landsat-8 images concerning a particular location and time interval from the 'EarthExplorer' repository. Images are saved as GTiff files in the `AppRoot` directory.

Usage

```
lsDownSearch(
  satellite,
  username,
  password,
  AppRoot,
  lvl = 1,
  product = c("sr", "source_metadata"),
  verbose = FALSE,
```

```
  untar = TRUE,
  raw.rm = FALSE,
  ...
)
```

Arguments

satellite	string containing the type of satellite ("ls7" or "ls8").
username	USGS's 'EarthExplorer' username.
password	USGS's 'EarthExplorer' password.
AppRoot	the download directory.
lvl	a number specifying the processing level. Default value, 1.
product	a character vector with the requested Level-2 products. By default c("sr", "source_metadata").
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
untar	logical argument. If TRUE, untars downloaded images.
raw.rm	logical argument. If TRUE, removes the raw images.
...	argumetns for nested functions:

- dates a vector with the capturing dates being searched. This argument is mandatory if startDate and endDate are not defined.
- startDate a Date class object with the starting date of the study period. This argument is mandatory if dates is not defined.
- endDate a Date class object with the ending date of the study period. This argument is mandatory if dates is not defined.
- region a Spatial*, projected raster*, or sf* class object defining the area of interest.
- any argument for ls8Search/ls7Search or lsDownload.

Details

lsDownSearch is a wrapper function of ls7Search, ls8Search, and lsDownload to search and download images in a single step. The function requires USGS's 'EarthExplorer' credentials, which can be obtained here.

The files from 'EarthExplorer' are compressed as 'tar.gz'. lsDownSearch decompresses the images and obtains the corresponding GTiffs. The GTiffs are saved in the AppRoot directory. To change this option, provide AppRoot = "full path". When untar=TRUE, the function untars the imagery in this location. Image decompression duplicates the information due to the presence of both, compressed and decompressed images. Set raw.rm = TRUE to remove the former ones.

Value

this function does not return anything. It saves the imagery as 'tar.gz' (and GTiff files) in a folder called 'raw' ('untar') in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)

wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# search and download the images from Landsat-8 between
# 01-01-2018 and 20-01-2018 for the region of Navarre
lsDownSearch(satellite = "ls8",
             username = "username",
             password = "password",
             startDate = as.Date("01-01-2018", "%d-%m-%Y"),
             endDate = as.Date("20-01-2018", "%d-%m-%Y"),
             extent = ex.navarre,
             AppRoot = wdir)

# remove metadata to free memory space
lsRemoveMetadata()

## End(Not run)
```

`lsEspaDownloadOrders` *Downloads the images that have been pre-processed by ESPA*

Description

`lsEspaDownloadOrders` downloads a set of images processed by the EROS Centre Science Processing Architecture (ESPA) through its application programming interface (API).

Usage

```
lsEspaDownloadOrders(
  orders,
  AppRoot,
  username = NULL,
  password = NULL,
  c.handle = NULL,
  verbose = FALSE,
  overwrite = FALSE,
  n.attempts = 5,
  untar = FALSE,
  ...
)
```

Arguments

`orders`	a list of the requested images as returned by `lsEspaGetOrderImages`.

AppRoot	the download directory.
username	USGS's 'EarthExplorer' username.
password	USGS's 'EarthExplorer' password.
c.handle	a curl handler created with the package 'curl' to stablish a connection with a preset password and username. This argument is mandatory if username and password are not defined.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
n.attempts	the number of attempts to download an image in case it becomes corrupted files.
untar	logical argument. If TRUE, untars the downloaded images.
...	argument for nested functions

Details

This function is part of a group of functions used to pre-process Landsat level-1 images. The pre-processing is carried out by ESPA on demand. lsEspaDownloadOrders downloads the images whose processing was completed according to lsEspaUpdateOrders. The function downloads and saves the imagery under the AppRoot directory. The function automatically creates two folders, called "raw" and "untar", to save the compressed and decompressed images respectively. The imagery is only decompressed when untar = TRUE.

Value

this function does not return anything. It saves the imagery as 'tar.gz' (and GTiff files) in a folder called 'raw' ('untar') in the AppRoot directory.

Examples

```
## Not run:
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# search Landsat-7 level-1
sres <- ls7Search(startDate = as.Date("01-01-2017", "%d-%m-%Y"),
                  endDate = as.Date("15-01-2017", "%d-%m-%Y"),
                  lonlat = c(-1.64323, 42.81687),
                  AppRoot = wdir)
# request to ESPA the prepocessing of level-1 images to get the surface reflectance
order <- lsEspaOrderImages(search.res = sres,
                           username = "username",
                           password = "password",
                           product = 'sr',
                           verbose = FALSE)
# get an ID for our request
orders <- lsEspaGetOrderImages(username = "username",
                               password = "password")
# follow up the status of the request
orders <- lsEspaUpdateOrders(orders = orders,
                             username = "username",
```

```
                              password = "password")
# saving directory
wdir.ls7.ESPA <- file.path(wdir,"Landsat7","ESPA")
dir.create(wdir.ls7.ESPA, recursive = TRUE)
# download when status says: complete
lsEspaDownloadOrders(orders = orders,
                     username = "username",
                     password = "password",
                     untar = TRUE,
                     AppRoot = wdir.ls7.ESPA)

## End(Not run)
```

`lsEspaGetOrderImages` *Gets a first response from ESPA regarding a recent request*

Description

`lsEspaGetOrderImages` obtains the identification number and the status of the request from the EROS Centre Science Processing Architecture (ESPA).

Usage

```
lsEspaGetOrderImages(
  username = NULL,
  password = NULL,
  c.handle = NULL,
  order.list = NULL,
  verbose = TRUE
)
```

Arguments

`username`	USGS's 'EarthExplorer' username.
`password`	USGS's 'EarthExplorer' password.
`c.handle`	a curl handler created with the package 'curl' to stablish a connection with a preset password and username. This argument is mandatory if `username` and `password` are not defined.
`order.list`	a list of orders from `lsEspaOrderImages`
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.

Details

This function is part of a group of functions used to pre-process Landsat level-1 images. The pre-processing is carried out by ESPA on demand. `lsEspaGetOrderImages` takes the identification (ID) number of a request carried out by `lsEspaOrderImages`. This ID is used to follow up the processing status with `lsEspaUpdateOrders`. All the status messages and their interpretation can be found in the ESPA's API User Guide.

Examples

```
## Not run:
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
# search Landsat 7 level-2
sres <- ls7Search(startDate = as.Date("01-01-2017", "%d-%m-%Y"),
                  endDate = as.Date("07-01-2017", "%d-%m-%Y"),
                  lonlat = c(-1.64323, 42.81687),
                  AppRoot = wdir)
# request to ESPA the pre-pocessing of level-2 images to
# get the surface reflectance
order <- lsEspaOrderImages(search.res = sres,
                           username = "username",
                           password = "password",
                           product = 'sr',
                           verbose = FALSE)
# get an ID for our request
lsEspaGetOrderImages(username = "username",
                     password = "password")

## End(Not run)
```

`lsEspaOrderImages` *Make a request to ESPA for pre-processing Landsat images*

Description

`lsEspaOrder` makes a request to the EROS Centre Science Processing Architecture (ESPA) to further process level-1 Landsat scenes.

Usage

```
lsEspaOrderImages(
  search.res,
  username,
  password,
  product = c("sr", "source_metadata"),
  verbose = FALSE
)
```

Arguments

`search.res`	the results from `ls7Search` or `ls8Search`.
`username`	USGS's 'EarthExplorer' username.
`password`	USGS's 'EarthExplorer' password.
`product`	the acronym of the requested product (see the details).
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.

Details

Landsat Level-1 images are pre-processed on demand by the EROS Centre Science Processing Architecture (ESPA). An order is placed to ESPA with the level-1 images that must be pre-processed (`search.res` and the requested final product (`product`). The products are identified by the following short-names:

- `toa`: for top of atmosphere reflectance.
- `bt`: for brightness temperature (thermal band TOA processing).
- `sr`: for surface reflectance.
- `sr_ndvi`: the normalized difference vegetation index from surface reflectance imagery.
- `sr_evi`: the enhanced vegetation index from surface reflectance imagery.
- `sr_savi`: the soil adjusted vegetation index from surface reflectance imagery.
- `sr_msavi`: the modified soil adjusted vegetation index from surface reflectance imagery.
- `sr_ndmi`: the normalized difference moisture index from surface reflectance imagery.
- `sr_nbr`: the normalized burn ratio from surface reflectance imagery.
- `sr_nbr2`: the normalized burn ratio 2 index from surface reflectance imagery.
- `prixel_qa`: for pixel quality assurance.

Value

this function does not return anything. It makes a petition to process level-1 images before doing the download.

Examples

```
## Not run:
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
# search Landsat 7 level-1
sres <- ls7Search(startDate = as.Date("01-01-2017", "%d-%m-%Y"),
                  endDate = as.Date("07-01-2017", "%d-%m-%Y"),
                  lonlat = c(-1.64323, 42.81687),
                  AppRoot = wdir)
# request to ESPA the pre-pocessing of level-1 images
# to get the surface reflectance
order <- lsEspaOrderImages(search.res = sres,
                           username = "username",
                           password = "password",
                           product = 'sr',
                           verbose = FALSE)

## End(Not run)
```

`lsEspaUpdateOrders` *Updates the status of a request made to ESPA*

Description

`lsEspaUpdateOrders` checks the current status of a request made to EROS Centre Science Processing Architecture (ESPA) to pre-process Landsat level-1 images

Usage

```
lsEspaUpdateOrders(
  orders,
  username = NULL,
  password = NULL,
  c.handle = NULL,
  verbose = FALSE
)
```

Arguments

`orders`	a list of the requested orders as returned by `lsEspaGetOrderImages`.
`username`	USGS's 'EarthExplorer' username.
`password`	USGS's 'EarthExplorer' password.
`c.handle`	a curl handler created with the package 'curl' to stablish a connection with a preset password and username. This argument is mandatory if `username` and `password` are not defined.
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.

Details

This function is part of a group of functions used to pre-process Landsat level 1 images. The pre-processing is carried out by ESPA on demand. `lsEspaUpdateOrders` uses the ID numbers gathered by `lsEspaGetOrderImages` regarding previous order requests to check the processing status. The function has to be run repeatedly until the status message states "complete". All the status messages and their interpretation can be found in the ESPA's API User Guide.

Value

this function returns a `dataframe` with the updated order information from ESPA.

Examples

```
## Not run:
src <- file.path(tempdir(),"Path_for_downloading_folder")
# search Landsat 7 level-1
search.res <- ls7Search(startDate = as.Date("01-01-2017", "%d-%m-%Y"),
                        endDate = as.Date("15-01-2017", "%d-%m-%Y"),
```

```
                        lonlat = c(-1.64323, 42.81687),
                        AppRoot = src)
# request to ESPA the prepocessing of level-1 images to get the surface reflectance
orders <- lsEspaOrderImages(search.res = search.res,
                            username = "username",
                            password = "password",
                            product = 'sr',
                            verbose = FALSE)
# get an ID for our request
orders <- lsEspaGetOrderImages(username = "username",
                               password = "password")
# follow up the status of the request
orders <- lsEspaUpdateOrders(orders = orders,
                             username = "username",
                             password = "password")

## End(Not run)
```

`lsGetDates` *Return the capturing dates of Landsat-7 or Landsat-8 images*

Description

`lsGetDates` reads the official name of a Landsat-7 or Landsat-8 image and returns the capturing date, as a `Date` class object.

Usage

```
lsGetDates(str, ...)
```

Arguments

`str`	the full path(s) or official name(s) of the Landsat-7 or Landsat-8 images from which the capturing date is retrieved.
`...`	arguments for nested functions: • `format` the format of the date being returned.

Details

The function works with file names (or their paths) regardless of their extension. The function accepts more than on file path, which can be passed as a vector of `characters`. Dates are returned as 'YYYY-mm-dd' by default. If another format is required, it can be modified through the argument `format`.

Value

a `Date` class object with the date of the Landsat image or `character` class, if the `format` argument is used.

Examples

```
# getting the capturing date from the name of a Landsat-8 image
file.ls8 <- "LC82000312017010LGN01.tar.gz"
date.ls8 <- lsGetDates(file.ls8)
print(date.ls8)
print(format(date.ls8, "%Y%j"))

# getting the capturing date from the name of a Landsat-7 and a Landsat-8
# image
file.ls7 <- c("LE72330822017009ASN01")
date.ls7 <- lsGetDates(file.ls7)
print(date.ls7)
```

`lsGetPathRow` *Return the pathrow of a tile of Landsat-7 or Landsat-8 images*

Description

`lsGetPathRow` reads the official name of a Landsat-7 or Landsa-8 image and returns the tile's path and row number, in "PPPRRR" format (Landsat naming convention).

Usage

```
lsGetPathRow(str)
```

Arguments

`str`	the full path(s) or official name(s) of the Landsat-7 or Landsa-8 images from which the tile's path and row numbers are retrieved.

Value

a string with the path and row in "PPPRRR" format.

Examples

```
# example of getting date from Landsat-8 image name
str <- c("LE72000302011066ASN00",
         "LE72000302011066ASN00")
pr <- lsGetPathRow(str)
print(pr)
```

lsMosaic *Mosaic a set of Landsat-7 or Landsat-8 images*

Description

lsMosaic merges the Landsat-7 or Landsat-8 imagery that covers a region of interest on the same dates.

Usage

```
lsMosaic(
  src,
  AppRoot,
  region,
  out.name = "outfile",
  verbose = FALSE,
  gutils = TRUE,
  overwrite = FALSE,
  ...
)
```

Arguments

src	the path to the folder with the Landsat-7 or Landsa-8 images in GTiff format.
AppRoot	the directory to save the mosaicked images.
region	a Spatial*, projected raster*, or sf class object defining the area of interest.
out.name	the name of the folder that stores the outputs. By default, "outfile" is assigned.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
gutils	logical argument. If TRUE, the function uses 'GDAL' utilities for mosaicking.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
...	arguments for nested functions: • pathrow a list of vectors with the path and row numbers of the tiles concerning the region of interest. This argument is mandatory if region is not provided. • bFilter a vector with the bands to be mosaicked. If not supplied, all bands are mosaicked. • dates a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The function mosaics the imagery in the src folder. The folder can hold GTiff images from several tiles, dates and bands. When only a subset of dates has to be mosaicked, the dates should be provided through the argument dateFilter. The dates must be provided as a Date class objects.

For further details about the bFilter argument go to lsDownload.Once mosaicked, the images can be cropped to fit the extent (optional). The extent can be defined in any coordinate reference system, since lsMosaic automatically reprojects the extent to match the projection of the image. The outputs are placed in the AppRoot directory, under the folder named as out.name. If no name is provided, the folder isn named "outfile". To use 'gutils = TRUE', a proper installation of 'GDAL' and the 'gdalUtils' library is required. This method is faster than native 'R' functions.

Value

this function does not return anything. It saves the imagery in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# main output directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download Landsat-8 images
lsDownSearch(satellite = "ls8",
             username = "username",
             password = "password",
             startDate = as.Date("01-01-2018", "%d-%m-%Y"),
             endDate = as.Date("20-01-2018", "%d-%m-%Y"),
             extent = ex.navarre,
             untar = TRUE,
             AppRoot = wdir)
# folder with the Landsat-8 untared images
wdir.ls8 <- file.path(wdir, "Landsat8")
tif.src <- file.path(wdir.ls8,"untar")
# mosaic the Landsat-8 images
lsMosaic(src = tif.src,
         AppRoot = wdir.ls8,
         out.name = "Navarre",
         extent = ex.navarre,
         gutils = TRUE, # using gdalUtils
         overwrite = TRUE) # overwrite

## End(Not run)
```

---

lsPreview *Preview Landsat-7 or Landsat-8 satellite images*

Description

lsPreview shows a preview of the n-th image from a set of search results on an interactive map.

Usage

```
lsPreview(
  searchres,
  n,
  dates,
  lpos = c(3, 2, 1),
  add.Layer = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

searchres	a data.frame with the results from ls7Search or ls8Search.
n	a numeric argument identifying the location of the image in searchres.
dates	a vector of Dates being considered for previewing. This argument is mandatory if n is not defined.
lpos	vector argument. Defines the position of the red-green-blue layers to enable false color visualization.
add.Layer	logical argument. If TRUE, the function plots the image on an existing map. Allows combinations of images on a map using senPreview and modPreview functions.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions: • arguments allowed by the viewRGB function from the mapview packages are valid arguments.

Details

The function shows a preview of the n-th output image from a search in the Landsat archives (ls7Search or ls8Search, with browseAvailable = "Y"). The preview is downloaded from USGS Bulk Metadata Service. Please, be aware that only some images may have a preview.

Value

this function does not return anything. It displays a preview of one of the search results.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
# retrieve jpg images covering Navarre between 2011 and 2013
sres <- ls7Search(startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  extent = ex.navarre,
```

```
                  precise = TRUE,
                  browseAvaliable = "Y",
                  AppRoot = wdir)
lsPreview(sres, 1)
# filter the images with less than 1% pixels covered by clouds
sres.cloud.free = subset(sres, sres$cloudCover < 1)
lsPreview(sres.cloud.free, 1)
lsPreview(sres.cloud.free, 2,add.Layer = TRUE)
# plot all the images in one date
lsPreview(sres.cloud.free,dates=as.Date("2013-09-04"))

## End(Not run)
```

`lsRemoveMetadata` *Remove the Landsat-7 or Lansat-8 metadata from the environment*

Description

`lsRemoveMetadata` removes Landsat-7 and/or Landsat-8 (`.LS7MD`/ `.LS8MD`) metadata from the environment in 'R'.

Usage

```
lsRemoveMetadata()
```

Details

The metadata file is loaded in 'R' with `ls7Search`, `ls8Search` and `lsDownSearch`. `lsRemoveMetadata` removes the metadata and frees up valuable RAM.

Examples

```
## Not run:
# creates a MetaData folder and downloads the csv file
# in the current working directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
ls8LoadMetadata(AppRoot = wdir)
lsRemoveMetadata()

## End(Not run)
```

lsSearch *Search Landsat 7-8 images using EarthExplorer API*

Description

lsSearch searches Landsat 7-8 images in the EarthExplorer API concerning a particular location and date interval. The function returns a data.frame with the names of the images and their metadata.

Usage

```
lsSearch(
  product,
  startDate,
  endDate,
  region,
  username,
  password,
  dates,
  logout = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

product	the name of the dataset. Avaliable names saved in 'RGISTools' (getRGISToolsOpt("EE.DataSets")).
startDate	a Date class object with the starting date of the study period. This argument is mandatory if dates is not defined.
endDate	a Date class object with the ending date of the study period. This argument is mandatory if dates is not defined.
region	a Spatial*, projected raster*, o r sf class object defining the area of interest. This argument is mandatory if pathrow, extent, or lonlat are not defined.
username	NASA's 'EarthData' username.
password	NASA's 'EarthData' password.
dates	a vector with the capturing dates being searched. This argument is mandatory if startDate and endDate are not defined.
logout	logical argument. If TRUE, logges out from EarthExplorer API
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions: • pathrow a list of vectors with the path and row numbers of the tiles concerning the region of interest. This argument is mandatory if region, extent or lonlat are not provided. Ex. list(c(200,31),c(200,30)).

- `lonlat` a vector with the longitude/latitude (EPSG:4326) coordinates of the point of interest. Ex. `c(-1.64323,42.81687)`. This argument is mandatory if `region`, `pathrow`, or `lonlat` are not defined.
- `extent` an `extent`, `Raster*`, or `Spatial*` object representing the region of interest with longitude/latitude (EPSG:4326) coordinates. This argument is mandatory if `region`, `pathrow` or `lonlat` are not defined.

Value

a `data.frame` with the name of the images and their metadata.

Examples

```
## Not run:
# search by path and row numbers of a tile
getRGISToolsOpt("EE.DataSets")
sres <- lsSearch(product = "LANDSAT_8_C1",
                 startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                 endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                 username = "username",
                 password = "password",
                 region = ex.navarre,
                 pathrow = list(c(200,31),c(200,30)))

sres <- lsSearch(product = "LANDSAT_8_C1",
                 startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                 endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                 username = "username",
                 password = "password",
                 lonlat = c(-1.64323,42.81687))


# search by extent (long/lat coordinates)
# load a spatial polygon object of Navarre
data(ex.navarre)
sres <- lsSearch(product = "LANDSAT_8_C1",
                 startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                 endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                 username = "username",
                 password = "password",
                 extent = ex.navarre)

## End(Not run)
```

---

lsUpdateEEDataSets *Update EarthExplorer dataset names*

Description

The 'EE.DataSets' option of 'RGISTools' contains all the dataset names supported by EarthExplorer API. If these names changes, `lsUpdateEEDataSets` updates these data.

Usage

```
lsUpdateEEDataSets(username, password, logout = TRUE, verbose = FALSE)
```

Arguments

username	USGS's 'EarthExplorer' username.
password	USGS's 'EarthExplorer' password.
logout	logical argument. If TRUE, logges out from EarthExplorer API
verbose	logical argument. If TRUE, the function prints the running steps and warnings.

Value

the dataset names in EarthExplorer API

Examples

```
## Not run:
setRGISToolsOpt("EE.DataSets",NULL)
getRGISToolsOpt("EE.DataSets")
datasetNames<-lsUpdateEEDataSets(username = "username",
                                 password = "password")

getRGISToolsOpt("EE.DataSets")

## End(Not run)
```

---

modCloudMask *Create cloud masks for MODIS images*

Description

modCloudMask creates cloud masks derived from the State Quality Assurance (State QA) band.

Usage

```
modCloudMask(src, AppRoot, out.name, overwrite = FALSE, ...)
```

Arguments

src	the path to the folder with the MODIS with state_1km images.
AppRoot	the directory where cloud masks are saved.
out.name	the name of the folder that stores the outputs. If the arguemnt is not defined the folder will be named as "CloudMask".
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
...	arguments for nested functions.

- dates a vector of dates being considered for creating cloud masks. This argument is optional.

Details

This function, interprets the State Quality Assurance (State QA) band to create cloud masks. The NA and 1 values of the mask represent cloudy and clear-sky pixels pixels respectively.

Value

this function does not return anything. It saves the cloud masks (CLD) as GTiff files in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)

# search and download images from MODIS between
# 01-01-2018 and 03-01-2018 for the region of Navarre
modDownSearch(product = "MOD09GA",
            startDate = as.Date("01-01-2017", "%d-%m-%Y"),
            endDate = as.Date("03-01-2017", "%d-%m-%Y"),
            username = "username",
            password = "password",
            AppRoot = wdir,
            extract.tif = TRUE,
            collection = 6,
            extent = ex.navarre)

# assign src1 as the output folder for modMosaic
wdir.mod <- file.path(wdir, "Modis")
wdir.mod.tiles <- file.path(wdir.mod, "MOD09GA")
wdir.mod.tif <- file.path(wdir.mod.tiles, "tif")
# mosaic the MODIS images
modMosaic(wdir.mod.tif, # the input folder
          AppRoot = wdir.mod.tiles, # the output folder
          out.name = "Navarre", # creates Navarre folder in AppRoot
          gutils = TRUE,
          extent = ex.navarre)


wdir.mod.navarre <- file.path(wdir.mod.tiles, "Navarre")
# generate the cloud masks
modCloudMask(src = wdir.mod.navarre,
             AppRoot = wdir.mod.tiles,
             overwrite = TRUE)

files.mod.cld <- file.path(wdir.mod.tiles,"CloudMask")
img.mod.cld <- stack(list.files(files.mod.cld, full.names=TRUE, pattern="CLD"))

# select b01
img.mod.navarre <- stack(list.files(wdir.mod.navarre,
```

```
                                    full.names=TRUE,
                                    recursive = TRUE,
                                    pattern="b01_1"))

# project to 500m
img.mod.cld.500 <- projectRaster(img.mod.cld,img.mod.navarre)

# plot the cloud free b01 layer
spplot(img.mod.navarre*img.mod.cld.500)

## End(Not run)
```

`modDownload` *Download MODIS images from a search list*

Description

`modDownload` downloads the images from a list of uniform resource locators (URLs) generated by the `modSearch` function from NASA's 'EartData' plataform. The images are saved as GTiff files in the `AppRoot` directory.

Usage

```
modDownload(
  searchres,
  AppRoot,
  username = NULL,
  password = NULL,
  nattempts = 5,
  verbose = FALSE,
  extract.tif = FALSE,
  overwrite = FALSE,
  raw.rm = FALSE,
  ...
)
```

Arguments

`searchres`	the output from the `modSearch` function.
`AppRoot`	the directory where the images will be saved.
`username`	NASA's 'EarthData' username.
`password`	NASA's 'EarthData' password.
`nattempts`	the number of attempts to download an image in case it becomes corrupted.
`verbose`	logical argument. If `TRUE`, the function prints running stages and warnings.
`extract.tif`	logical argument. If `TRUE`, extracts all the layers from hdf files and saves them as GTiff.

overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
raw.rm	logical argument. If TRUE, removes the raw images.
...	argument for nested functions: • dates a vector with the capturing dates being considered for downloading. • bFilter a vector with the bands to be extracted when extract.tif=TRUE. If not supplied, all bands are extracted.

Details

modDownload is able to download MODIS Terra and Aqua products. These products are published in the 'EarthData' Platform. The platform is supported by the Earth Observing System Data and Information System (EODIS) and managed NASA's Earth Science Data Systems (ESDS). modDownload requires credentianls from an 'EarthData' account to access the NASA's web data service, which can be obtained here.

When extract.tif = TRUE, the function decompresses the imagery. If only a subset of bands is required, band names can be provided through the bFilter argument. The band names are specified by "B" and the two-digit band number (e.g., "B01"). Image decompression duplicates the information due to the presence of both, compressed and decompressed images. Set raw.rm = TRUE to remove former ones.

Value

this function does not return anything. It saves the imagery as 'hdf' (and GTiff files) in a folder called 'raw' ('tif') in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
sres <- modSearch(product = "MYD13A2",
                  startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  collection = 6,
                  extent = ex.navarre)
head(sres)
# download the first image in sres
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
wdir.mod <- file.path(wdir,"Modis","MYD13A2")
wdir.mod.hdf <- file.path(wdir.mod,"hdf")
modDownload(mList[1],
            username = "username",
            password = "password",
            AppRoot = wdir.mod.hdf)
# download all images in mList
modDownload(sres,
            username = "username",
            password = "password",
            AppRoot = wdir.mod.hdf)
```

```
## End(Not run)
```

`modDownSearch` *Search and download MODIS images*

Description

`modDownSearch` searches and downloads MODIS images concerning a particular location and time interval from the 'EarthData' repository. Images are saved as GTiff files in the `AppRoot` directory.

Usage

```
modDownSearch(
  product,
  username,
  password,
  AppRoot,
  collection = 6,
  nattempts = 5,
  verbose = FALSE,
  extract.tif = FALSE,
  ...
)
```

Arguments

`product`	a `character` argument with the short name of the MODIS product.
`username`	NASA's 'EarthData' username.
`password`	NASA's 'EarthData' password.
`AppRoot`	the directory to save the outcoming time series.
`collection`	MODIS collection, by default 6.
`nattempts`	the number of attempts to download an image in case it becomes corrupted.
`verbose`	logical argument. If `TRUE`, the function prints running stages and warnings.
`extract.tif`	logical argument. If `TRUE`, extracts all the layers from hdf files and saves them as GTiff.
`...`	arguments for nested functions: • `dates` a vector with the capturing dates being searched. This argument is mandatory if `startDate` and `endDate` are not defined. • startDate a `Date` class object with the starting date of the study period. This argument is mandatory if `dates` is not defined. • endDate a `Date` class object with the ending date of the study period. This argument is mandatory if `dates` is not defined.

- `lonlat` a vector with the longitude/latitude coordinates of the point of interest. This argument is mandatory if `region` or `extent` are not defined.
- `extent` an `sf`, `Raster*`, or `Spatial*` object representing the region of interest with longitude/latitude coordinates. This argument is mandatory if `polygon` or `lonlat` are not defined.
- `region` A list of vectors defining the points of a polygon in longitude/latitude format. This argument is mandatory if `lonlat` or `extent` are not defined.
- Any argument in `modExtractHDF` function. Ex. `bFilter="b01_1"`.

Details

`modDownSearch` searches via NASA's Common Metadata Repository and downloads the imagert from the 'EarthData' web service to download the imagery. The catalogue of MODIS products can be found here. The catalogue shows detailed information about the products and their short names. By the time 'RGISTools' is released, NASA carries out the maintenance of its website on Wednesdays, which may cause an error when connecting to their server. You can get your 'EarthData' credentials here.

Value

this function does not return anything. It saves the imagery as 'tar.gz' (and GTiff files) in a folder called 'raw' ('tif') in the `AppRoot` directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(src)
modDownSearch(product = "MOD09GA",
              startDate = as.Date("01-01-2018", "%d-%m-%Y"),
              endDate = as.Date("03-01-2018", "%d-%m-%Y"),
              username = "username",
              password = "password",
              AppRoot = wdir,
              extract.tif = TRUE,
              collection = 6,
              extent = ex.navarre)
wdir.mod.tif <- file.path(wdir,"Modis","MOD09GA","tif")
files.mod <- list.files(wdir.mod.tif,
                        pattern = "\\.tif$",
                        full.names = TRUE,
                        recursive = TRUE)[c(16,19,18)]
img.mod <- stack(files.mod)
qrange <- c(0.001, 0.999)
img.mod.rgb <- varRGB(img.mod[[1]],
                      img.mod[[2]],
                      img.mod[[3]],
                      qrange)
plotRGB(img.mod.rgb)
```

```
## End(Not run)
```

---

modExtractHDF *Convert an HDF file into a set of GTiff files*

Description

modExtractHDF converts the original format of MODIS images (hierarchical data format or HDF) into GTiffs (one file for each layer).

Usage

```
modExtractHDF(
  filesHDF,
  AppRoot,
  overwrite = FALSE,
  bFilter = NULL,
  rm.band = NULL,
  region = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

filesHDF	the full path where the HDF files are located.
AppRoot	the directory where the extracted images are saved.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
bFilter	a vector containing the names of the bands to extract.
rm.band	a vector containing the names of the bands excluded from the extraction.
region	a Spatial*, projected raster*, or sf* class object defining the area of interest for image masking.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for downloading.

Details

HDF files cannot be directly loaded into 'R', so they must be converted into GTiffs. To acomplish this task, the function modExtractHDF borrows the gdalwarp and gdal_translate functions from the 'gdalUtils' package. Further details about these functions can be found in the corresponding package manual. 'GDAL' and 'gdalUtils' must be properly installed to use modExtractHDF. GTiffs can be loaded in 'R' using the 'raster' package.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)

wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
wdir.mod <- file.path(wdir, "Modis","MOD11A1")
sres <- modSearch(product = "MOD11A1",
                  startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  collection = 6,
                  extent = ex.navarre)

# download the images of the list
wdir.mod <- file.path(wdir, "Modis", "MOD11A1")
modDownload(searchres = sres,
            username = "username",
            password = "password",
            AppRoot = wdir.mod)

wdir.mod.tif<-file.path(wdir.mod,"tif")
wdir.mod.hdf <- file.path(wdir.mod, "hdf")

# extract the first image
files.mod.hdf <- list.files(wdir.mod.hdf,
                        full.names = TRUE,
                        pattern = "\\.hdf$")
files.mod.hdf.1 <- files.mod.hdf[1]
modExtractHDF(filesHDF = files.mod.hdf.1,
              AppRoot = wdir.mod.tif)

## End(Not run)
```

`modFolderToVar` *Compute a remote sensing index from a time series of MODIS multi-spectral images*

Description

`modFolderToVar` computes a remote sensing index from the spectral bands of a time series of MODIS images. The images are specified by the path to the folder that stores the imagery (resulting from the `modMosaic` function). The function returns a `RasterStack` with a time series of images of the remote sensing index.

Usage

```
modFolderToVar(
  src,
```

```
  AppRoot,
  fun,
  getStack = FALSE,
  overwrite = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

src	path to the folder with the MODIS multispectral images.
AppRoot	the directory of the outcoming time series.
fun	a function that computes the remote sensing index.
getStack	logical argument. If TRUE, returns the time series of images as a RasterStack, otherwise the images are saved in the Hard Drive Device (HDD).
overwrite	logical argument. If TRUE, it overwrites the existing images with the same name.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The function requires the definition of the src and fun arguments. The src is usually the path resulting from modMosaic. The fun argument can be any function from this package beginning with "var" (varNDVI, varEVI, etc.). Custom functions can also be implemented. If fun = varRGB, then the argument getStack must be equal to FALSE and the red-gree-blue (RGB) images must be imported afterwards.

Value

this function does not return anything, unless getStack = TRUE which then returns a RasterStack with the time series of with the index.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# main output directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download MOD09 images
modDownSearch(product = "MOD09GA",
              startDate = as.Date("01-01-2018", "%d-%m-%Y"),
              endDate = as.Date("03-01-2018", "%d-%m-%Y"),
              username = "username",
              password = "password",
```

```
              AppRoot = wdir, # output folder for tif images
              extract.tif = TRUE,
              collection = 6,
              extent = ex.navarre)
# assign wdir.mod as the output folder from modMosaic
wdir.mod <- file.path(wdir, "Modis", "MOD09GA") # output directory
wdir.mod.tif <- file.path(wdir.mod, "tif") # input directory
# mosaic the MODIS images
modMosaic(wdir.mod.tif,
          AppRoot = wdir.mod,
          out.name = "Navarre")
# path to the folder with the mosaicked images
wdir.mod.navarre <- file.path(wdir.mod, "Navarre")
# generate NDVI images of Navarre
wdir.mod.var <- file.path(wdir.mod, "Variables")
dir.create(wdir.mod.var)
modFolderToVar(src = wdir.mod.navarre,
               fun = varEVI,
               scfun = getRGISToolsOpt("MOD09SCL"),
               AppRoot = wdir.mod.var,
               overwrite = TRUE)
# import mosaicked images (.tif) to the environment in `R'
files.mod.evi <- list.files(file.path(wdir.mod.var,"EVI"),
                            pattern = "\\.tif$",
                            full.names = TRUE,
                            recursive = TRUE)

img.mod.evi <- lapply(files.mod.evi,raster)
spplot(img.mod.evi[[1]],at=seq(-1,2.5))

## End(Not run)
```

`modGetDates` *Return the capturing dates of MODIS images*

Description

`modGetDates` reads the official name of MODIS images and returns its capturing date, as `Date` class object.

Usage

```
modGetDates(str, ...)
```

Arguments

`str`	the full path(s) or official name(s) of MODIS images from which the capturing date is retrieved.
`...`	arguments for nested functions: • `format` the format of the date being returned.

Details

The function works with file names (or their paths) regardless of their extension. The function accepts more than on file path, which can be passed as a `list` of `characters`. Dates are returned as 'YYYY-mm-dd' by default. If another format is required, it can be modified through the argument `format`.

Value

a `Date` class object with the date of the MODIS image or `character` class, if `format` argument is used.

Examples

```
# getting the capturing date from the name of a MODIS image
file.mod <- 'MYD13A2.A2016361.h17v04.006.2017285133407.hdf'
modGetDates(file.mod)

# a list of the full file paths of MODIS images, mixing .hdf and .tif files
file.mod<-list('MYD13A2.A2013297.h17v04.006.2015269230726.hdf',
               'MYD13A2.A2013313.h17v04.006.2015271071143.tif')
modGetDates(file.mod, format = "%Y%j")
```

---

modGetPathRow *Return the pathrow of a tile of MODIS images*

Description

`modGetPathRow` reads the official name of a MODIS image and returns the tile's path and row number, in 'hXXvYY' format (MODIS naming convention).

Usage

```
modGetPathRow(str)
```

Arguments

`str`	the full path(s) or official name(s) of the MODIS images from which the tile's path and row numbers are retrieved.

Value

a string with the path and row in "hXXvYY" format.

Examples

```
# getting the path and row number of the tile of a Landsat-8 image
files.mod <- "MYD09GA.A2003136.h17v04.005.2008324054225"
pr.mod <- modGetPathRow(files.mod)
print(pr.mod)
```

modMosaic *Mosaic a set of MODIS images*

Description

modMosaic merges the MODIS imagery that covers a region of interest on the same dates.

Usage

```
modMosaic(
  src,
  AppRoot,
  region = NULL,
  out.name = "outfile",
  verbose = FALSE,
  gutils = TRUE,
  overwrite = FALSE,
  ...
)
```

Arguments

src	the path of the folder with the MODIS images in GTiff format.
AppRoot	the directory where the mosaicked images are saved.
region	a Spatial*, projected raster*, or sf class object defining the area of interest.
out.name	the name of the folder that stores the outputs. By default, "outfile" is assigned.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
gutils	logical argument. If TRUE, the function uses 'GDAL' utilities for mosaicking.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
...	arguments for nested functions: • pathrow a vector of character with the path and row numbers of the tiles concerning the region of interest in 'hXXvYY' format. This argument is mandatory if region is not provided. • bFilter a vector with the bands to be mosaicked. If not supplied, all bands are mosaicked. • dates a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The function mosaics the imagery in the src folder. The folder can hold GTiff images from several tiles, dates and bands. When only a subset dates has to be mosaicked, the dates should be provided through the argument dates. The dates must be provided as a Date class objects. For further details about the bFilter argument, go to the modDownload function. Once mosaicked, the images can

be cropped to fit the region (optional). The region can be defined in any coordinate reference system, since modMosaic automatically reproject the extent to match the projection of the image. The outputs will be placed in the AppRoot directory, under the folder named as out.name. If no name is provided, the folder is named "outfile".

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# main output directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download MODIS images
modDownSearch(product = "MOD09GA",
              startDate = as.Date("01-01-2018", "%d-%m-%Y"),
              endDate = as.Date("03-01-2018", "%d-%m-%Y"),
              username = "username",
              password = "password",
              AppRoot = wdir,
              extract.tif = TRUE,
              collection = 6,
              extent = ex.navarre)
# folder with the MODIS images extracted
wdir.mod <- file.path(wdir, "Modis", "MOD09GA")
wdir.mod.tif <- file.path(wdir.mod, "tif")
# mosaic the MODIS images
modMosaic(wdir.mod.tif,
          AppRoot = wdir.mod,
          out.name = "Navarre",
          gutils = TRUE,
          overwrite = TRUE,
          region = ex.navarre)

## End(Not run)
```

---

modPreview *Preview MODIS satellite images*

Description

modPreview shows a preview of the n-th image from a set of search results on an interactive map.

Usage

```
modPreview(
  searchres,
  n,
  dates,
  lpos = c(3, 2, 1),
```

```
  add.Layer = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

searchres	a vector with the results from modSearch.
n	a numeric argument identifying the location of the image in searchres.
dates	a vector with the dates being considered for previewing. This argument is mandatory if n is not defined.
lpos	vector argument. Defines the position of the red-green-blue layers to enable false color visualization.
add.Layer	logical argument. If TRUE, the function plots the image on an existing map. Allows combinations of images on a map using lsPreview and senPreview functions.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions: • arguments allowed by the viewRGB function from mapview packages are valid arguments.

Details

The function shows a preview of the n-th output image from a search in the MODIS archives (modSearch, with resType = "browseurl"). The preview is downloaded from the 'EarthData' Platform. Please, be aware that only some images may have a preview.

Value

this function does not return anything. It displays a preview of one of the search results.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# retrieve jpg images covering Navarre region between 2011 and 2013
sres <- modSearch(product = "MOD09GA",
                  startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  collection = 6,
                  extent = ex.navarre)

modPreview(sres,n=1)
modPreview(sres,2,add.Layer=T)

## End(Not run)
```

`modSearch` *Search MODIS images*

Description

`modSearch` searches MODIS images in the NASA Common Metadata Repository (CMR) concerning a particular location and date interval. The function returns a `character` vector with the names of the images and their uniform resource locators (URLs)

Usage

```
modSearch(product, collection = 6, verbose = FALSE, ...)
```

Arguments

`product`	the short name of the MODIS product.
`collection`	MODIS collection. By default, 6.
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.
`...`	arguments for nested functions:

- `dates` a vector with the capturing dates being searched. This argument is mandatory if `startDate` and `endDate` are not defined.
- `startDate` a `Date` class object with the starting date of the study period. This argument is mandatory if `dates` is not defined.
- `endDate` a `Date` class object with the ending date of the study period. This argument is mandatory if `dates` is not defined.
- `region` a `Spatial*`, projected `raster*`, or `sf` class object defining the area of interest. This argument is mandatory if `extent` or `lonlat` are not defined.
- `lonlat` a vector with the longitude/latitude coordinates of the point of interest. This argument is mandatory if `region` or `extent` are not defined.
- `extent` an `extent`, `Raster*`, or `Spatial*` object representing the region of interest with longitude/latitude coordinates. This argument is mandatory if `region` or `lonlat` are not defined.

Details

`modSearch` uses the NASA Common Metadata Repository (CMR) powered API. The catalogue of MODIS products can be found here. The catalogue shows the product short names and provides detailed information about the product. By the time 'RGISTools' is released, NASA carries out the maintenance of its website on Wednesdays, which may cause an error when connecting to their server. You can get your 'EarthData' credentials here.

The function can be used to retrieve the web address of the preview (`resType = "browseurl"`) or the actual image (`resType = "url"`). By default, the URL points towards the actual image.

Value

a vector with the url for image downloading.

Examples

```
## Not run:
# load a spatial polygon object of Navarre with longitude/latitude coordinates
data(ex.navarre)
# searching MODIS MYD13A2 images between 2011 and 2013 by longitude/latitude
# using a polygon class variable
sres <- modSearch(product = "MYD13A2",
                  startDate = as.Date("01-01-2011", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2013", "%d-%m-%Y"),
                  collection = 6,
                  extent = ex.navarre)
# region of interest: defined based on longitude/latitude extent
# searching MODIS MYD13A2 images in 2010 by longitude/latitude
# using a extent class variable defined by the user
aoi = extent(c(-2.49, -0.72, 41.91, 43.31))
sres <- modSearch(product = "MYD13A2",
                  startDate = as.Date("01-01-2010", "%d-%m-%Y"),
                  endDate = as.Date("31-12-2010", "%d-%m-%Y"),
                  collection = 6,
                  extent = aoi)
head(sres)

## End(Not run)
```

---

senCloudMask *Create cloud masks for Sentinel-2 images*

Description

senCloudMask creates cloud masks derived from the cloud probability band (CLDPRB) band from the "S2MSI2A" product.

Usage

```
senCloudMask(
  src,
  AppRoot,
  out.name,
  resbands,
  sensitivity = 50,
  overwrite = FALSE,
  ...
)
```

Arguments

src	the path to the folder with the "S2MSI2A" images.
AppRoot	the directory where the cloud masks are saved.
out.name	the name of the folder that stores the outputs. If the arguemnt is not defined the folder will be named as "CloudMask".
resbands	a character vector argument. Defines the band resolution used to create the cloud mask. Ex "20m" or "60m".
sensitivity	a numeric argument. Defines the sensitivity of the cloud detection method.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The valid threshold range for sensitivity is 0-100. By default, the argument is set to 50.

Value

this function does not return anything. It saves the cloud masks (CLD) as GTiff files in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# Download S2MSI1C products sensed by Sentinel-2
# between the julian days 210 and 218, 2018
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
senDownSearch(startDate = as.Date("2018210", "%Y%j"),
              endDate = as.Date("2018218", "%Y%j"),
              platform = "Sentinel-2",
              extent = ex.navarre,
              product = "S2MSI2A",
              pathrow = c("R094"),
              username = "username",
              password = "password",
              AppRoot = wdir)
# define the paths to the Sentinle-2 images and the
# folder with the unzipped images
wdir.sen <- file.path(wdir, "Sentinel-2")
wdir.sen.unzip <- file.path(wdir.sen, "unzip")
# mosaic the Sentinel-2 images
senMosaic(wdir.sen.unzip,
          AppRoot = wdir.sen,
          gutils = TRUE,
          out.name = "Navarre")
```

```
# calculate the cloud mask
wdir.sen.navarre <- file.path(wdir.sen, "Navarre")
senCloudMask(src = wdir.sen.navarre,
             resbands = "60m",
             overwrite = TRUE,
             sensitivity = 98,
             AppRoot = wdir.sen)

# define the path for the Sentinel-2 cloud mask
wdir.sen.cloud <- file.path(wdir.sen, "CloudMask")

# select B02 images of 60 meters
tiles.sen.navarre <- list.files(wdir.sen.navarre,
                            full.names = TRUE,
                            recursive = TRUE,
                           pattern = "\\.tif$")
tiles.sen.navarre.b2 <- tiles.sen.navarre[grepl("B02",tiles.sen.navarre)]
tiles.sen.navarre.b2 <- tiles.sen.navarre.b2[grepl("60m",tiles.sen.navarre.b2)]

# generate a 60-meter resolution cloud mask
tiles.sen.cloud <- list.files(wdir.sen.cloud,
                              full.names = TRUE,
                              pattern = "\\.tif$")
tiles.sen.cloud.60 <- tiles.sen.cloud[grepl("60m",tiles.sen.cloud)]

# remove the cloud mask from b02 tiles
img.sen.navarre.b2 <- stack(tiles.sen.navarre.b2)
img.sen.cloud.60 <- stack(tiles.sen.cloud.60)
img.sen.navarre.b2.cloud.free <- img.sen.navarre.b2*img.sen.cloud.60
# plot b2 cloud free layers
spplot(img.sen.navarre.b2.cloud.free)

## End(Not run)
```

`senDownload` *Download Sentinel images from a search list*

Description

`senDownload` downloads the images from a list of uniform resource locators (URLs) generated by the `senSearch` function from ESA's 'SciHub' web service. The images are saved as GTiff files in the `AppRoot` directory.

Usage

```
senDownload(
  searchres,
  AppRoot,
  username = NULL,
```

```
  password = NULL,
  nattempts = 5,
  unzip = FALSE,
  overwrite = FALSE,
  omit.md5.error = FALSE,
  ...
)
```

Arguments

searchres	the output from the senSearch function.
AppRoot	the directory where the outcoming time series are saved.
username	ESA's 'SciHub' username.
password	ESA's 'SciHub' password.
nattempts	the number of attempts to download an image in case it becomes corrupted.
unzip	logical argument. If TRUE, unzips the images.
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
omit.md5.error	logical argument. If TRUE, omits md5 errors and do not removes the downloaded image.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for downloading.
- bFilter a vector with the bands to be extracted when unzip=TRUE. If not supplied, all bands are extracted.

Details

senDownload downloads the list of URLs provided by senSearch. In case the process is interrupted, the image file could be corrupted. The function detects the corrupted files and restarts the process. To prevent the computer from crashing, there is a maximum number of attempts to try to donwnload the image (nattempts). The default number of attempts is set to 3. The function requires an ESA's 'SciHub' account, which can be obtained here.

When unzip = TRUE, the function decompresses the imagery. If only a subset of bands is required, band names can be provided through the bFilter argument. The band names are specified by "B", followed by the two-digit band number, and the resolution of the band (either 10m, 20m, or 60m) (e.g, "B01_10m"). Image decompression duplicates the information due to the presence of both, compressed and decompressed images. Set raw.rm = TRUE to remove former ones.

Value

this function does not return anything. It saves the imagery as 'zip' (and JP2 files) in a folder called 'raw' ('unzip') in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# download S2MSI1C products sensed by Sentinel-2
# in July-August 2018
sres <- senSearch(startDate = as.Date("2018-07-29","%Y-%m-%d"),
                        endDate = as.Date("2018-08-06","%Y-%m-%d"),
                        platform = "Sentinel-2",
                        extent = ex.navarre,
                        product = "S2MSI1C",
                        username = "username",
                        password = "password")

# filtering the path R094 where Navarre is located
names(sres)
sres.sen.R094 <- sres[grepl("R094", names(sres))]
names(sres.sen.R094)
# list the dates in sres
senGetDates(names(sres.sen.R094),format="%Y%j")
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
# donwload the imagery
senDownload(searchres = sres.sen.R094,
            username = "username",
            password = "password",
            AppRoot = wdir,
            unzip = TRUE)
wdir.sen.unzip <- file.path(wdir,"Sentinel","unzip")
files.sen.unzip<-list.files(wdir.sen.unzip,
                            pattern = "\\TCI.jp2$",
                            full.names = TRUE,
                            recursive = TRUE)
img.sen.rgb<-stack(files.sen.unzip[1])
plotRGB(img.sen.rgb)

## End(Not run)
```

---

senDownSearch *Search and download Sentinel images*

Description

senDownSearch searches and downloads Sentinel images concerning a particular location and time interval from 'SciHub's' repository.

Usage

```
senDownSearch(username, password, AppRoot, verbose = FALSE, ...)
```

Arguments

username	ESA's 'SciHub' username.
password	ESA's 'SciHub' password.
AppRoot	the directory where the images are saved.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
...	arguments for nested functions:

- product the type of Sentinel product. Ex. "S2MSI1C", "S2MSI2A", "S2MSI2Ap", ...
- startDate a Date class object with the starting date of the study period. This argument is mandatory if dates is not defined.
- endDate a Date class object with the ending date of the study period. This argument is mandatory if dates is not defined.
- dates a vector with the capturing dates being considered for searching. This argument is mandatory if startDate and endDate are not defined.
- region a Spatial*, projected raster*, or sf class object defining the area of interest.
- extent an extent, Raster*, or Spatial* object representing the region of interest with longitude/latitude coordinates. This argument is mandatroy when region is not defined.
- platform the name of the Sentinel mission ("Sentinel-1", "Sentinel-2", ...).
- nattempts the number of attempts to download an image in case it becomes corrupted.
- unzip logical argument. If TRUE, unzips the images.
- verbose logical argument. If TRUE, the function prints the running steps and warnings.

Details

senDownSearch is a wrapper function of senSearch and senDownload to search and download images in a single step. The function requires ESA's 'SciHub' credentials, which can be obtained here.

Value

this function does not return anything. It saves the imagery as 'zip' (and JP2 files) in a folder called 'raw' ('unzip') in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# Download S2MSI1C products sensed by Sentinel-2
# between the julian dates 210 and 218, 2018
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
```

```
senDownSearch(startDate = as.Date("2018210", "%Y%j"),
              endDate = as.Date("2018218", "%Y%j"),
              platform = "Sentinel-2",
              extent = ex.navarre,
              product = "S2MSI1C",
              pathrow = c("R094"),
              username = "username",
              password = "password",
              AppRoot = wdir)

wdir.sen <- file.path(wdir, "Sentinel-2")
wdir.sen.unzip <- file.path(wdir.sen, "unzip")

files.sen.unzip <- list.files(wdir.sen.unzip,
                              pattern = "\\TCI.jp2$",
                              full.names = TRUE,
                              recursive = TRUE)
img.sen.rgb <- stack(files.sen.unzip[1])
plotRGB(img.sen.rgb)

## End(Not run)
```

`senFolderToVar` *Compute a remote sensing index from a time series of Sentinel-2 images*

Description

`senFolderToVar` computes a remote sensing index from the spectral bands of a time series of Sentinel-2 images. The images are specified by the path to the folder that stores the imagery (resulting from the `senMosaic` function). The function returns a `RasterStack` with a time series of images of the remote sensing index.

Usage

```
senFolderToVar(
  src,
  AppRoot,
  fun,
  getStack = FALSE,
  overwrite = FALSE,
  verbose = FALSE,
  resbands = c("10m", "20m", "60m"),
  ...
)
```

Arguments

src	the path to the folder with the Sentinel-2 multispectral images.
AppRoot	directory where the outcoming time series is saved.
fun	a function that computes the remote sensing index.
getStack	logical argument. If TRUE, returns the time series of images as a RasterStack, otherwise the images are saved in the Hard Drive Device (HDD).
overwrite	logical argument. If TRUE, overwrites the existing images with the same name.
verbose	logical argument. If TRUE, the function prints the running steps and warnings.
resbands	the resolution of the image being used to compute index, when the imagery comes from the Senintel-2 "S2MSI2A" product. By default, all resolutions (10m, 20m, and 60m) are used.
...	arguments for nested functions.

- dates a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The function requires the definition of the src and fun arguments. The src is usually the path resulting from senMosaic. The fun argument can be any function from this package beginning with "var" (varNDVI, varEVI, etc.). Custom functions can also be implemented. If fun = varRGB, then the argument getStack must be equal to FALSE and the red-green-blue (RGB) images must be imported afterwards. Caution! It is mandatory to use level-2 products to get accurate derived variables.

Value

this function does not return anything, unless getStack = TRUE which then returns a RasterStack with the time series of with the index.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# main output directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download Sentinel-2 images
senDownSearch(startDate = as.Date("2018210","%Y%j"),
              endDate = as.Date("2018218","%Y%j"),
              platform = "Sentinel-2",
              extent = ex.navarre,
              product = "S2MSI1C",
              pathrow = c("R094"),
              username = "username",
              password = "password",
              AppRoot = wdir)
# folder with the unzipped images from Sentinel-2
```

```
wdir.sen <- file.path(wdir,"Sentinel-2")
wdir.sen.unzip <- file.path(wdir.sen, "unzip")
# mosaic the Sentinel-2 images
senMosaic(wdir.sen.unzip,
          AppRoot = wdir.sen,
          gutils = TRUE,
          out.name = "Navarre")
# path to the folder with the mosaicked images
wdir.sen.navarre <- file.path(wdir.sen, "Navarre")
wdir.sen.var <- file.path(wdir.sen.navarre, "Navarre_Variables")
dir.create(wdir.sen.var)
# generate EVI images of Navarre
senFolderToVar(wdir.sen.navarre,
               fun = varEVI,
               resbands = c("60m"),
               AppRoot = wdir.sen.var)

files.sen.evi <- list.files(file.path(wdir.sen.var,"EVI"),
                            pattern = "\\.tif$",
                            full.names = TRUE,
                            recursive = TRUE)

img.sen.evi <- lapply(files.sen.evi, raster)
spplot(img.sen.evi[[1]])

## End(Not run)
```

`senGetDates` *Return the capturing dates of Sentinel-2 images*

Description

`senGetDates` reads the official name of a Sentinel-2 image and returns the capturing date, as a `Date` class object.

Usage

```
senGetDates(str, ...)
```

Arguments

`str`	the full path(s) or official name(s) of the Sentinel images from which the capturing date is retrieved.
`...`	arguments for nested functions: • `format` the format of the date being returned.

Details

The function works with file names (or their paths) regardless of their extension. The function accepts more than on file path, which can be passed as a vector of `characters`. Dates are returned as 'YYYY-mm-dd' by default. If another format is required, it can be modified through the argument `format`.

Value

a `Date` class object with the date of the Sentinel image or `character` class, if the `format` argument is used.

Examples

```
# getting the capturing date from the name of Sentinel-2 images
file.sen <- c("S2A_MSIL1C_20170102T111442_N0204_R137_T30TWN_20170102T111441.SAFE",
         "S2A_OPER_PRD_MSIL1C_PDMC_20160308T090616_R094_V20160305T110109_20160305T110109")
date.sen <- senGetDates(file.sen)
print(date.sen)
print(format(date.sen,"%Y%j"))
senGetDates(file.sen, format = "%Y%j")
```

`senGetOrbit` *Return the relative orbit of the Sentinel-2 satellite*

Description

`senGetOrbit` reads the official name of a Sentinel image and returns relative orbit, in "NXXXX_RYYY" or "RYYY" format (Sentinel namig convention).

Usage

```
senGetOrbit(str)
```

Arguments

`str`	the full path(s) or official name(s) of the Sentinel images from which the orbits are retrieved.

Details

Get information about the relative orbits here.

Value

an string with the relative orbit of the image in "NXXXX_RYYY" or "RYYY" format, depending on the version of name convention.

Examples

```
#example of getting date from Sentinel2 image name
files.sen <- c("S2A_MSIL1C_20170102T111442_N0204_R137_T30TWN_20170102T111441.SAFE",
          "S2A_OPER_PRD_MSIL1C_PDMC_20160308T090616_R094_V20160305T110109_20160305T110109")
tile.sen <- senGetOrbit(files.sen)
print(tile.sen)
```

`senGetTile` *Return the pathrow of a tile from Sentinel-2 images*

Description

`senGetTile` reads the official name of a Sentinel-2 image and returns the tile's path and row number, in "TTTSSS" format (Sentinel naming convention).

Usage

```
senGetTile(str)
```

Arguments

`str`	the full path(s) or official name(s) of the Sentinel-2 images from which the tile's path and row numbers are retrieved.

Details

Find more details about the Sentinel tiling system here.

Value

a string with the path and row in "TTTSSS" format.

Examples

```
# getting path and row numbers from a couple of Sentinel-2 images
files.sen <- c("S2A_MSIL1C_20170102T111442_N0204_R137_T30TWN_20170102T111441.SAFE",
          "S2A_OPER_PRD_MSIL1C_PDMC_20160308T090616_R094_V20160305T110109_20160305T110109")
pr.sen <- senGetTile(files.sen)
print(pr.sen)
```

`senMosaic` *Mosaic a set of Sentinel-2 images*

Description

`senMosaic` merges the Sentinel-2 imagery that covers a region of interest on the same dates.

Usage

```
senMosaic(
  src,
  AppRoot,
  region = NULL,
  out.name = "outfile",
  verbose = FALSE,
  gutils = TRUE,
  overwrite = FALSE,
  ...
)
```

Arguments

`src`	the path of the folder with the Sentinel images in GTiff format.
`AppRoot`	the directory to save the mosaicked images.
`region`	a `Spatial*`, projected `raster*`, or `sf` class object defining the area of interest.
`out.name`	the name of the folder that stores the outputs. By default, "outfile" is assigned.
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.
`gutils`	logical argument. If `TRUE`, the function uses 'GDAL' utilities for mosaicking.
`overwrite`	logical argument. If `TRUE`, overwrites the existing images with the same name.
`...`	arguments for nested functions: • `pathrow` a `list` of vectors with the path and row numbers of the tiles concerning the region of interest. This argument is mandatory if `region` is not defined. • `bFilter` a vector with the bands to be mosaicked. If not supplied, all bands are mosaicked. • `dates` a vector with the capturing dates being considered for mosaicking. If not supplied, all dates are mosaicked.

Details

The function mosaics the imagery in the `src` folder. The folder can hold GTiff images from several tiles, dates and bands. When only a subset dates has to be mosaicked, the dates should be provided through the argument `dates`. The dates must be provided as a `Date` class object.For further details about the `bFilter` argument, go to the `senDownload` function. Once mosaicked, the images can

be cropped to fit the region (optional). The region can be defined in any coordinate reference system, since senMosaic automatically reproject the extent to match the projection of the image. The outputs will be placed in the AppRoot directory, under the folder named as out.name. If no name is provided, the folder is named "outfile".

Value

this function does not return anything. It saves the imagery in the AppRoot directory.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# main output directory
wdir <- file.path(tempdir(),"Path_for_downloading_folder")
print(wdir)
# download Sentinel-2 images
senDownSearch(startDate = as.Date("2018210", "%Y%j"),
              endDate = as.Date("2018218", "%Y%j"),
              platform = "Sentinel-2",
              extent = ex.navarre,
              product = "S2MSI1C",
              pathrow = c("R094"),
              username = "username",
              password = "password",
              AppRoot = wdir)
# folder with the unzipped images
wdir.sen <- file.path(wdir, "Sentinel-2")
wdir.sen.unzip <- file.path(wdir.sen, "unzip")
# mosaic the Sentinel-2 images
senMosaic(wdir.sen.unzip,
          AppRoot = wdir.sen,
          gutils = TRUE,
          out.name = "Navarre")

wdir.sen <- file.path(wdir.sen,"Navarre")
# load and plot a Sentinel-2 image
files.sen <- list.files(wdir.sen, pattern = "\\.tif$", full.names = TRUE ,recursive = TRUE)
# print Sentinel-2 bands
getRGISToolsOpt("SEN2BANDS")
file.sen.rgb <- stack(files.sen[grepl("TCI",files.sen)][1])
plotRGB(file.sen.rgb)

## End(Not run)
```

---

senPreview *Preview Sentinel-2 satellite images*

Description

`senPreview` shows a preview of the `n`-th image from a set of search results on an interactive map.

Usage

```
senPreview(
  searchres,
  username,
  password,
  n,
  dates,
  lpos = c(3, 2, 1),
  add.Layer = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`searchres`	a vector with the results from `senSearch`.
`username`	ESA's 'SciHub' username.
`password`	ESA's 'SciHub' password.
`n`	a `numeric` argument identifying the row of the image in `searchres`.
`dates`	a vector with the dates being considered for previewing. This argument is mandatory if `n` is not defined.
`lpos`	vector argument. Defines the position of the red-green-blue layers to enable false color visualization.
`add.Layer`	logical argument. If `TRUE`, the function plots the image on an existing map. Allows combinations of images on a map using `lsPreview` and `modPreview` functions.
`verbose`	logical argument. If `TRUE`, the function prints the running steps and warnings.
`...`	arguments for nested functions: • arguments allowed by the `viewRGB` function from `mapview` packages are valid arguments

Details

The function shows a preview of the `n`-th output image from a search in Sentinel archives (`modSearch`). The preview is downloaded from 'SciHub's' website. Please, be aware that only some images may have a preview. Credentials from an ESA's 'SciHub' account are needed, which can be obtained here.

Value

this function does not return anything. It displays a preview of one of the search results.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# perform the search query
sres <- senSearch(startDate = as.Date("2018210","%Y%j"),
                  endDate = as.Date("2018218","%Y%j"),
                  platform = "Sentinel-2",
                  extent = ex.navarre,
                  product = "S2MSI1C",
                  username = "username",
                  password = "password")
# preview some images
senPreview(sres, username = "username", password = "password",n=1)
senPreview(sres, username = "username", password = "password",n=3, add.Layer =TRUE)

# show the dates in julian days
senGetDates(names(sres),format="%Y%j")

senPreview(sres,
           username = "username",
           password = "password",
           dates = senGetDates(names(sres[3])))

## End(Not run)
```

---

senSearch *Search Sentinel images*

Description

`senSearch` searches Sentinel images through ESA's powered application programming interface (API), called 'SciHub', that concern a particular location and date interval. The function returns a `data.frame` with the names of the images and their uniform resource locators (URLs).

Usage

```
senSearch(username, password, ...)
```

Arguments

`username`	ESA's 'SciHub' username.
`password`	ESA's 'SciHub' password.
`...`	arguments for nested functions:

- `product` the type of Sentinel product. Ex. "S2MSI1C", "S2MSI2A", "S2MSI2Ap", ...
- `dates` a vector with the capturing dates being searched. This argument is mandatory if `startDate` and `endDate` are not defined.

- `startDate` a `Date` class object with the starting date of the study period. This argument is mandatory if `dates` is not defined.
- `endDate` a `Date` class object with the ending date of the study period. This argument is mandatory if `dates` is not defined.
- `region` a `Spatial*`, projected `raster*`, or `sf` class object defining the area of interest. This argument is mandatory if `extent` or `lonlat` are not defined.
- `extent` an `extent`, `Raster*`, or `Spatial*` object representing the region of interest with longitude/latitude coordinates. This argument is mandatory if `region` or `lonlat` are not defined.
- `lonlat` a vector with the longitude/latitude coordinates of the point of interest. This argument is mandatory if `region` or `extent` are not defined.
- `platform` the name of the Sentinel mission ("Sentinel-1", "Sentinel-2", ...).
- `qformat` the format of the response.
- `verbose` logical argument. If `TRUE`, the function prints the running steps and warnings.

Details

`senSearch` uses the ESA's powered API ('SciHub').The catalogue of Sentinel-2 products can be found here. The function explores the images available for a specific location and time-span. Dates must be provided as `Date` class objects. Credentials from ESA's 'SciHub' are needed and they can be obtained here.

Examples

```
## Not run:
# load a spatial polygon object of Navarre
data(ex.navarre)
# perform the search query
sres <- senSearch(startDate = as.Date("2018210", "%Y%j"),
                  endDate = as.Date("2018218", "%Y%j"),
                  platform = "Sentinel-2",
                  region = ex.navarre,
                  product = "S2MSI1C",
                  username = "username",
                  password = "password")
head(sres)

## End(Not run)
```

`setRGISToolsOpt` *Change the default value of an RGISTools option*

Description

`setRGISToolsOpt` changes the default value of an 'RGISTools' configuration variable. This function can be jointly used with `showRGISToolsOpt` and `getRGISToolsOpt`.

Usage

```
setRGISToolsOpt(opt, value, env = optEnv)
```

Arguments

opt	a character with the name of the option.
value	the new value of the selected option.
env	the environment where the 'RGISTools' option is saved.

Value

this function does not return anything.

Examples

```
# list avaliable options names
showRGISToolsOpt()
# list the URL where the Landsat-7 metadata is located
getRGISToolsOpt("LS7META.dir")
# change the URL where the Landsat-7 metadata is located
setRGISToolsOpt("LS7META.dir", "NewMTDir")
# list the URL where the Landsat-7 metadata is located
getRGISToolsOpt("LS7META.dir")
```

---

showRGISToolsOpt *Print the name of all RGISTools configuration variables*

Description

showRGISToolsOpt prints the name of all options in the 'RGISTools' package. This function can be jointly used with setRGISToolsOpt and getRGISToolsOpt.

Usage

```
showRGISToolsOpt(env = optEnv)
```

Arguments

env	the environment where the 'RGISTools' option are saved.

Value

a character vector with the names of the configuration variables.

Examples

```
showRGISToolsOpt()
```

varEVI *Calculate the enhanced vegetation index (EVI)*

Description

varEVI computes the enhanced vegetation index (EVI) from the blue, near-infrared (NIR) and red bands.

Usage

```
varEVI(blue, red, nir, scfun = function(r) {
    r
})
```

Arguments

blue	a raster with the blue band of the capture.
red	a raster with the red band of the capture.
nir	a raster with the NIR band of the capture.
scfun	a function to re-scale the original pixel values into reflectance (0-1).

Details

The enhanced vegetation index (EVI) is a vegetation indicator that improves the sensitivity towards high biomass densities compared to NDVI (Huete et al. 2002) (See varNDVI). This function is used within ls7FolderToVar, ls8FolderToVar, modFolderToVar and senFolderToVar.

Value

An EVI image in raster format.

References

Huete A, Didan K, Miura T, Rodriguez EP, Gao X, Ferreira LG (2002). "Overview of the radiometric and biophysical performance of the MODIS vegetation indices." *Remote sensing of environment*, **83**(1-2), 195–213.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern = "\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")
scale.factor <- 0.0001

# select the red, blue and nir bands
```

```
img.mod.red <- raster(files.mod[1]) * scale.factor
img.mod.blue <- raster(files.mod[3]) * scale.factor
img.mod.nir <- raster(files.mod[2]) * scale.factor
# calculate the EVI without scale
img.mod.evi <- varEVI(img.mod.blue, img.mod.red, img.mod.nir)
# calculate the EVI scaling 0-1
img.mod.evi.2 <- varEVI(img.mod.blue, img.mod.red, img.mod.nir,scfun=getRGISToolsOpt("MOD09SCL"))
img.mod.evi.12 <- stack(img.mod.evi,img.mod.evi.2)
# plot the image
spplot(img.mod.evi.12,col.regions=rev(terrain.colors(20)),at = c(seq(0,1,0.05)))
```

`varMSAVI2` *Calculate the modified soil-adjusted vegetation index (MSAVI2)*

Description

`varMSAVI2` computes the modified soil-adjusted vegetation index 2 (MSAVI2) from the near-infrared (NIR) and red bands.

Usage

```
varMSAVI2(red, nir)
```

Arguments

`red`	a `raster` with the red band of the capture.
`nir`	a `raster` with the NIR band of the capture.

Details

The modified soil adjusted vegetation index 2 (MSAVI2) is a vegetation indicator that removes the effect from background variations (Qi et al. 1994). This function is used within `ls7FolderToVar`, `ls8FolderToVar`, `modFolderToVar` and `senFolderToVar`.

Value

A MSAVI2 image in `raster` format.

References

Qi J, Chehbouni A, Huerte A, Kerr Y, Sorooshian S (1994). "A modified soil adjusted vegetation index." *Remote Sensing Environment*, **48**, 119–126.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")
# select the red and NIR bands
img.mod.red <- raster(files.mod[1])
img.mod.nir <- raster(files.mod[2])
# calculate the MSAVI2 image
img.mod.msavi2 <- varMSAVI2(img.mod.red, img.mod.nir)
# plot the image
spplot(img.mod.msavi2,col.regions=rev(topo.colors(20)))
```

---

varNBR *Calculate the normalized burn ratio (NBR)*

Description

varNBR computes the normalized burn ratio (NBR) from the near-infrared (NIR) and shortwave-infrared 2 (SWIR2) bands.

Usage

```
varNBR(nir, swir2)
```

Arguments

nir	a raster with the NIR band of the capture.
swir2	a raster with the SWIR2 band of the capture.

Details

The normalized burn ratio (NBR) is an index that identifies burned areas by comparing its value before and after the fire event. It is calculated using the NIR and SWIR2 bands (Garcia and Caselles 1991).This function is used within ls7FolderToVar, ls8FolderToVar, modFolderToVar and senFolderToVar.

Value

A NBR image in raster format.

References

Garcia ML, Caselles V (1991). "Mapping burns and natural reforestation using Thematic Mapper data." *Geocarto International*, **6**(1), 31–37.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the NIR and SWIR2 bands
files.mod.nir <- raster(files.mod[2])
files.mod.swir2 <- raster(files.mod[7])
# calculate the NBR image
files.mod.nbr <- varNBR(files.mod.nir, files.mod.swir2)
# plot the image
spplot(files.mod.nbr,col.regions=rev(heat.colors(20)))
```

---

varNBR2 *Calculate the normalized burn ratio 2 (NBR2)*

Description

varNBR2 computes the NBR2 index from the shortwave infrared 1 (SWIR1) and shortwave infrared 2 (SWIR2).

Usage

```
varNBR2(swir1, swir2)
```

Arguments

swir1	a raster with the the SWIR1 band of the capture.
swir2	a raster with the the SWIR2 band of the capture.

Details

The normalized burn ratio 2 (NRB2) is an index to identify burned areas. In contrast to NBR, NRB2 highlights the sensitivity to water in vegetation (Lutes et al. 2006). This function is used within ls7FolderToVar, ls8FolderToVar, modFolderToVar and senFolderToVar.

Value

A NBR2 image in raster format.

References

Lutes DC, Keane RE, Caratti JF, Key CH, Benson NC, Sutherland S, Gangi LJ (2006). "FIREMON: Fire effects monitoring and inventory system." *Gen. Tech. Rep. RMRS-GTR-164. Fort Collins, CO: US Department of Agriculture, Forest Service, Rocky Mountain Research Station. 1 CD.*, **164**.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the SWIR1 and SWIR2 bands
img.mod.swir1 <- raster(files.mod[6])
img.mod.swir2 <- raster(files.mod[7])
# calculate the NBR2 image
img.mod.nbr2 <- varNBR2(img.mod.swir1,img.mod.swir2)
# plot the image
spplot(img.mod.nbr2,col.regions=rev(heat.colors(20)))
```

---

varNDMI *Calculate the normalized difference moisture (water) index (NDMI)*

Description

varNDMI computes the normalized difference moisture index (NDMI) from the near-infrared (NIR) and shortwave-infrared 1 (SWIR1) bands.

Usage

```
varNDMI(nir, swir1)
```

Arguments

nir	a raster with the nir band of the capture.
swir1	a raster with the swir1 band of the capture.

Details

The normalized difference moisture index (NDMI) is an index that represents the water stress levels of the canopy, using the NIR and SWIR (Gao 1995). This function is used within ls7FolderToVar, ls8FolderToVar, modFolderToVar and senFolderToVar.

Value

A NDMI image in raster format.

References

Gao B (1995). "Normalized difference water index for remote sensing of vegetation liquid water from space." In *Imaging Spectrometry*, volume 2480, 225–237. International Society for Optics and Photonics.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the NIR and SWIR1 bands
img.mod.nir <- raster(files.mod[2])
img.mod.swir1 <- raster(files.mod[6])
# calculate the NDMI image
img.mod.ndmi <- varNDMI(img.mod.nir,img.mod.swir1)
# plot the image
spplot(img.mod.ndmi)
```

---

varNDVI *Calculate the normalized difference vegetation index (NDVI)*

Description

`varNDVI` computes the normalized difference vegetation index (NDVI) from the red an near-infrared (NIR) bands.

Usage

```
varNDVI(red, nir)
```

Arguments

`red`	a `raster` with the red band of the capture.
`nir`	a `raster` with the NIR band of the capture.

Details

The normalized difference vegetation index (NDVI) is the most widely used index for monitoring vegetation dynamics. The NDVI reflects the vegetation vigour and it is closely related to the amount of photosynthetically active radiation absorbed (Rouse Jr 1972). This function is used within `ls7FolderToVar`, `ls8FolderToVar`, `modFolderToVar` and `senFolderToVar`.

Value

A NDVI image in `raster` format.

References

Rouse Jr JW (1972). "Monitoring the vernal advancement and retrogradation (green wave effect) of natural vegetation." *NASA technical Reports Server*. https://ntrs.nasa.gov/search.jsp?R=19730009607.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the red and NIR bands
img.mod.red <- raster(files.mod[1])
img.mod.nir <- raster(files.mod[2])
# calculate the NDVI image
img.mod.ndvi <- varNDVI(img.mod.red,img.mod.nir)
# plot the image
spplot(img.mod.ndvi,col.regions=rev(terrain.colors(20)))
```

---

varNDWI *Calculates the normalized difference water index (NDWI)*

Description

varNDWI Calculate the normalized difference water index (NDWI) from the green and near-infrared (NIR) bands.

Usage

```
varNDWI(green, nir)
```

Arguments

green	a raster with the green band of the capture.
nir	a raster with the NIR band of the capture.

Details

The normalized difference water index (NDWI) is a ratio between the green and near-infrared bands of the spectrum that was developed to detect open water areas and minimize the influence of the soil and vegetation variations (McFeeters 1996). This function is used within ls7FolderToVar, ls8FolderToVar, modFolderToVar and senFolderToVar.

Value

A NDWI image in raster format.

References

McFeeters SK (1996). "The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features." *International journal of remote sensing*, **17**(7), 1425–1432.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the green and NIR bands
img.mod.green <- raster(files.mod[4])
img.mod.nir <- raster(files.mod[2])
# calculate the NDWI image
img.mod.ndwi <- varNDWI(img.mod.green,img.mod.nir)
# plot the image
spplot(img.mod.ndwi,col.regions=rev(rainbow(20)))
```

---

varRGB *Generate an RGB image from 3 spectral bands*

Description

varRGB creates red-green-blue (RGB) images as a RasterStack by scaling the pixel values to 0-255 color range.

Usage

```
varRGB(red, green, blue, q.range = c(), rPath = NULL, region = NULL)
```

Arguments

red	a raster with the red band of the capture.
green	a raster with the green band of the capture.
blue	a raster with the blue band of the capture.
q.range	a vector with the minimum and maximum reflectance quantiles being considered.
rPath	the file path where the resulting RGB image is saved.
region	a Spatial*, projected raster*, or sf class object defining the area of interest for image masking.

Details

The function rescales the original reflectance values to a range of 0-255. The function re-arranges the RGB bands to create a stack ready to visualize with plotRGB. Bands may contain outliers which cause the image to look dark. Use the q.range argument to remove the outliers and get a better-looking image.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the red, blue and NIR bands
img.mod.red <- raster(files.mod[1])
img.mod.blue <- raster(files.mod[3])
img.mod.green <- raster(files.mod[4])

q.range=c(0.001,0.999)
img.mod.rgb<-varRGB(img.mod.red,img.mod.green,img.mod.blue,q.range)
print(plotRGB(img.mod.rgb))
```

---

varSAVI *Calculates the soil-adjusted vegetation index (SAVI)*

Description

`varSAVI` Calculate the soil-adjusted vegetation index (SAVI) from the red and near-infrared (NIR) bands.

Usage

```
varSAVI(red, nir, L = 0.5, scfun = function(r) {
    r
})
```

Arguments

`red`	a `raster` with the red band of the capture.
`nir`	a `raster` with the NIR band of the capture.
`L`	a constant to remove soil background effect. A value of 0.5 is recommended in the literature.
`scfun`	a function to re-scale the original pixel values into reflectance (0-1).

Details

The soil adjusted vegetation index (SAVI) is an indicator engineered to remove the influence of the soil background effect (Huete 1988). This function is used within `ls7FolderToVar`, `ls8FolderToVar`, `modFolderToVar` and `senFolderToVar`.

Value

A SAVI image in `raster` format.

References

Huete AR (1988). "A soil-adjusted vegetation index (SAVI)." *Remote sensing of environment*, **25**(3), 295–309.

Examples

```
# path to the cropped and cutted MODIS images for the region of Navarre
wdir <- system.file("ExNavarreVar", package = "RGISTools")
# list all the tif files
files.mod <- list.files(wdir, pattern="\\.tif$", recursive = TRUE, full.names = TRUE)
# print the MOD09 bands
getRGISToolsOpt("MOD09BANDS")

# select the red and NIR bands
img.mod.red <- raster(files.mod[1])
img.mod.nir <- raster(files.mod[2])
# calculate the SAVI image
img.mod.savi <- varSAVI(img.mod.red,img.mod.nir,scfun=getRGISToolsOpt("MOD09SCL"))
# plot the image
spplot(img.mod.savi,col.regions=rev(topo.colors(20)))
```

Index