Package 'MODIS'

July 3, 2017

Type Package

Title Acquisition and Processing of MODIS Products

Version 1.1.0

Date 2017-07-03

Description Download and processing functionalities for the Moderate Resolution Imaging Spectroradiometer (MODIS). The package provides automated access to the global online data archives (LPDAAC and LAADS) and processing capabilities such as file conversion, mosaicking, subsetting and time series filtering.

License GPL (>= 3) | file LICENSE

LazyData TRUE

Depends mapdata, R (>= 2.10), raster

Imports bitops, devtools, grDevices, graphics, mapedit, maps, methods, parallel, ptw, RCurl, rgdal, sf, sp, stats, utils, XML

URL https://github.com/MatMatt/MODIS

ByteCompile TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Author Matteo Mattiuzzi [aut],
Jan Verbesselt [ctb],
Tomislav Hengl [ctb],
Anja Klisch [ctb],
Forrest Stevens [ctb],
Steven Mosher [ctb],
Bradley Evans [ctb],
Agustin Lobo [ctb],
Koen Hufkens [ctb],
Florian Detsch [cre, aut]

Maintainer Florian Detsch <fdetsch@web.de>

Repository CRAN

Date/Publication 2017-07-03 13:35:41 UTC

R topics documented:

`MODIS-package` *MODIS Acquisition and Processing*

Description

MODIS Acquisition and Processing

Details

Download and processing functionalities for the Moderate Resolution Imaging Spectroradiometer (MODIS). The package provides automated access to the global online data archives (LPDAAC and LAADS) and processing capabilities such as file conversion, mosaicking, subsetting and time series filtering.

Author(s)

Matteo Mattiuzzi, Jan Verbesselt, Tomislav Hengl, Anja Klisch, Forrest Stevens, Steven Mosher, Bradley Evans, Agustin Lobo, Florian Detsch

Maintainer: Matteo Mattiuzzi <matteo@mattiuzzi.com>

---

addCollection *Add New Product to MODIS Collections*

Description

`addCollection` is a non-exported helper function to add a new product column to the product collections managed by **MODIS** (see `MODIS:::collections`). Once added, the specified product will be tracked and, if required, kept up-to-date by `getCollection`.

Usage

```
addCollection(product, collection = NA, path_ext = "inst/external",
  overwrite = FALSE, ...)
```

Arguments

`product`	Character. Name of the product that should be added to the 'collections' dataset, see `getCollection`.
`collection`	Numeric. Optional information about available collections. If not supplied, this defaults to 'NA' and the user is required to manually retrieve information about available collections via `getCollection(..., forceCheck = TRUE)`. Note that the latter operation requires the previous execution of `MODIS:::addProduct` and `MODIS:::addServer` to make the newly added product available to `getCollection`.
`path_ext`	Character. Path to folder containing file 'MODIS_Products.RData'. When working with RStudio projects (.Rproj), this usually defaults to 'inst/external'.
`overwrite`	Logical. If `TRUE`, the initial '.RData' file located in 'path_ext' will be overwritten.
`...`	Currently not used.

Value

A 'data.frame' which, for each product featured by **MODIS**, holds information about available collections.

Author(s)

Florian Detsch

See Also

`MODIS:::collections`.

Examples

```
## Not run:
## E.g., add collection of MODIS evapotranspiration product
MODIS:::addCollection(product = "MOD16A2", collection = 105)

## End(Not run)
```

---

addProduct *Add New Product to MODIS Inventory*

Description

addProduct is a non-exported helper function to add a new entry to the list of satellite products featured by **MODIS** (see MODIS:::MODIS_Products).

Usage

```
addProduct(product, sensor = "MODIS", platform = c("Terra", "Aqua"), pf1,
  pf2, topic, type = c("Tile", "Swath", "CMG"), res, temp_res,
  internalseparator = "\\.", server = c("LPDAAC", "LAADS"),
  path_ext = "inst/external", overwrite = FALSE, ...)
```

Arguments

product	Character. Name of the product that should be added to the inventory, see getProduct.
sensor	Character. Sensor type, defaults to 'MODIS'.
platform	Character. Satellite platform on which the specified 'sensor' is mounted, defaults to "Terra".
pf1, pf2	Character. Online server paths.
topic	Character. The official name of 'product'.
type	Character. Product type, defaults to 'Tile'.
res	Character. Spatial resolution of 'product', e.g. "1000m".
temp_res	Character. Temporal resolution of 'product', e.g. "8 Day".
internalseparator	Character. Separator string matching the product's naming convention, defaults to '\.' for MODIS products.
server	Character. Server to download the data from (more than one entry is possible).
path_ext	Character. Path to folder containing file 'MODIS_Products.RData'. When working with RStudio projects (.Rproj), this usually defaults to 'inst/external'.
overwrite	Logical. If TRUE, the initial '.RData' file located in 'path_ext' will be overwritten.
...	Currently not used.

Value

A 'list' holding the updated contents of file 'MODIS_Products.RData'.

Author(s)

Florian Detsch

See Also

MODIS:::MODIS_Products, getProduct.

Examples

```
## Not run:
## E.g., add MODIS evapotranspiration product
MODIS:::addProduct(product = "MOD16A2", sensor = "MODIS", platform = "Combined",
                   pf1 = "MOLT", pf2 = "MOD", res = "1000m", temp_res = "8 Day",
                   topic = "Global Terrestrial Evapotranspiration", server = "NTSG")

## End(Not run)
```

---

addServer *Add New Remote Server to MODIS Inventory*

Description

addServer is a non-exported helper function to add a new entry to the list of online (FTP) servers featured by **MODIS** (see MODIS:::MODIS_FTPinfo).

Usage

```
addServer(name, sensor = "MODIS", basepath, varpath, content = "images",
  path_ext = "inst/external", overwrite = FALSE, ...)
```

Arguments

name	Character. Name of the remote server that should be added to the inventory.
sensor	Character. Sensor type, defaults to 'MODIS'.
basepath	Character. Absolute online server path.
varpath	Character. Pattern of organizational structure on server.
content	Character. Content type, defaults to "images".
path_ext	Character. Path to folder containing file 'MODIS_FTPinfo.RData'. When working with RStudio projects (.Rproj), this usually defaults to 'inst/external'.
overwrite	Logical. If TRUE, the initial '.RData' file located in 'path_ext' will be overwritten.
...	Currently not used.

Value

A 'list' holding the updated contents of 'MODIS_FTPinfo.RData'.

Author(s)

Florian Detsch

See Also

`MODIS:::MODIS_FTPinfo`.

Examples

```
## Not run:
## E.g., add server of MODIS evapotranspiration product
MODIS:::addServer(name = "NTSG", sensor = "MODIS",
                  basepath = "ftp://ftp.ntsg.umt.edu/pub/MODIS/NTSG_Products/MOD16/",
                  varpath = "PRODUCT.CCC/YYYY/DDD/")

## End(Not run)
```

---

aggInterval *Create Periods for Temporal Composites*

Description

The creation of custom temporal aggregation levels (e.g., half-monthly, monthly) from native 16-day MODIS composites usually requires the definition of date sequences based on which the "composite_day_of_the_year" SDS is further processed. Complementing `transDate`, which returns the respective start and end date only, this function creates full-year (half-)monthly or annual composite periods from a user-defined temporal range.

Usage

```
aggInterval(x, interval = c("month", "year", "fortnight"))
```

Arguments

`x`	`Date` object, see eg default value of 'timeInfo' in `temporalComposite`.
`interval`	`character`. Time period for aggregation. Currently available options are "month" (default), "year" and "fortnight" (i.e., every 1st and 15th day of the month).

Value

A `list` with the following slots:

- `$begin`: The start date(s) of each (half-)monthly timestep as `Date` object.
- `$end`: Same for end date(s).
- `$beginDOY`: Similar to `$begin`, but with `character` objects in MODIS-style date format (i.e., "%Y%j"; see `strptime`).
- `$endDOY`: Same for end date(s).

Author(s)

Florian Detsch

See Also

`transDate`.

Examples

```
dates <- do.call("c", lapply(2015:2016, function(i) {
  start <- as.Date(paste0(i, "-01-01"))
  end <- as.Date(paste0(i, "-12-31"))
  seq(start, end, 16)
}))

intervals <- c("month", "year", "fortnight")
lst <- lapply(intervals, function(i) {
  aggInterval(dates, interval = i)
}); names(lst) <- intervals

print(lst)
```

---

arcStats *Get Summary of Local MODIS Data*

Description

In the same manner as `getHdf`, this function quantifies the availability of local MODIS hdf data and gives you an overview (plot or/and table) of locally available MODIS grid hdf files.

Usage

```
arcStats(product, collection = NULL, extent = "global",
  begin = "2000.01.01", end = format(Sys.time(), "%Y.%m.%d"),
  asMap = TRUE, outName = NULL, ...)
```

Arguments

product	character, see getProduct. MODIS grid product to be checked.
collection	character or integer, see getCollection. MODIS product version.
extent	Extent information, defaults to 'global'. See getTile.
begin	character. Begin date of MODIS time series, see transDate.
end	character. End date, defaults to 'Today' expressed in a function.
asMap	Controls output type. Possible options are TRUE (png), FALSE (csv) or 'both'.
outName	character. Name of output file, defaults to 'product.collection.YYYYMMDDHHMMSS.png' (or *.csv) of the function call or, if applicable, 'product.collection.extent.YYYYMMDDHHMMSS.png' (or *.csv).
...	Arguments passed to MODISoptions, most importantly outProj and outDirPath.

Value

An invisible NULL (provably this will change to a matrix-like object similar to the '*.csv' output). If asMap= TRUE, a 'table.csv' and a 'image.png' file(s) in outDirPath.

Author(s)

Matteo Mattiuzzi

Examples

```
## Not run:
# arcStats result on a webserver:
# "http://ivfl-info.boku.ac.at/index.php/eo-data-processing/status-of-the-local-archive"
#
# The following examples are expecting that you have some data stored locally!
############################################################
# generates 2 png's and 2 csv's one for TERRA one for AQUA
arcStats(product="M.D13Q1")

# generates 2 png's and 2 csv's one for TERRA one for AQUA with the specified countries.
arcStats(product="M.D13Q1",extent=c("austria","germany","italy"))

# generates 1 png and 1 csv for AQUA.
arcStats(product="MYD13Q1",begin="2005001",outName="MyDataStart2005")

# generates 1 png for AQUA for the selected area and plots it in 'Sinusoidal'.
arcStats(product="MYD13Q1",begin="2005001",asMap=TRUE, outName="InteractiveSelection2005",
         extent=getTile(), outProj="asIn")

# generates 1 png for AQUA for the selected area and plots it in 'Geographic' Coordinates.
arcStats(product="MYD13Q1",begin="2005001",asMap=TRUE, outName="InteractiveSelection2005",
         extent=getTile(), outProj="GEOGRAPHIC")

## End(Not run)
```

delHdf *Delete Local MODIS Grid Files*

Description

Delete MODIS grid files to reduce the local storage.

Usage

```
delHdf(product, collection = NULL, extent = "global", tileV = NULL,
  tileH = NULL, begin = NULL, end = NULL, ask = TRUE, ...)
```

Arguments

product	character, see getProduct.
collection	character or integer, see getCollection.
extent	Extent information, defaults to 'global'. See getTile.
tileV	numeric or character. Vertical tile number(s), see tileH.
tileH	numeric or character. Horizontal tile number, see getTile.
begin	character. Begin date of MODIS time series, see transDate for formatting.
end	Same for end date.
ask	logical. If TRUE (default), the user is being asked for deletion after checking.
...	Arguments passed to MODISoptions, particularly localArcPath.

Author(s)

Matteo Mattiuzzi

Examples

```
## Not run:

# YOU ARE ASKED TO CONFIM THE DELETION! BUT IF THE EXAMPLE THIS FOR YOU SENSITIVE DATA CHANGE IT!

# REMOVE "MYD11A2" from specific date range and area subset:
# delHdf(product="MYD11A2",begin="2010001",end="2010.02.01",extent="austria")
# or
# delHdf(product="MYD11A2",begin="2010001",end="2010.02.01",tileV=18:19,tileH=4)

# REMOVE "MOD11A2" and "MYD11A2" from specific date range but globaly:
# delHdf(product="M.D11A2",begin="2010001",end="2010.02.01")

# REMOVE ALL "MOD11A2" from local archive:
# delHdf(product="MOD11A2")

## End(Not run)
```

detectBitInfo *List MODIS Quality Information*

Description

This function returns MODIS QA information for a specific product. It gets the information from an internal database and not all products are available.

Usage

```
detectBitInfo(product, what = "all", warn = TRUE)
```

Arguments

product	character, see getProduct.
what	character. Parameter name, i.e. https://lpdaac.usgs.gov/dataset_discovery/modis/modis_products_table/mod13q1, (TABLE 2: MOD13Q1 VI Quality; Long Name).
warn	logical, whether or not to throw warning messages.

Value

If what = "all" a data.frame, else a list.

Author(s)

Matteo Mattiuzzi

Examples

```
detectBitInfo("MOD13Q1")
detectBitInfo("MOD13Q1", "VI usefulness")

detectBitInfo("MYD17A2")
```

extractBits *Extract Bit-Encoded Information and Create Weights Raster*

Description

This function applies bitAnd and bitShiftR from **bitops** to convert bit-encoded information. It is also possible to convert this information to a scale from 0 to 1 in order to use it as weighting information in functions like whittaker.raster or smooth.spline.raster.

Usage

```
extractBits(x, bitShift = 2, bitMask = 15, filename = "", ...)

makeWeights(x, bitShift = 2, bitMask = 15, threshold = NULL,
  filename = "", decodeOnly = FALSE, ...)

maskWater(X, bitShift = NULL, bitMask = NULL, keep = NULL,
  datatype = "INT1U", ...)
```

Arguments

x	matrix, vector or Raster* object.
bitShift	integer. Bit starting point, see examples and detectBitInfo.
bitMask	integer. Bit mask size, see examples and detectBitInfo.
filename	character passed to writeRaster. If not specified, output is written to a temporary file.
...	Other arguments passed to writeRaster.
threshold	integer. Threshold for valid quality.
decodeOnly	logical. If FALSE (default), convert bits to weights from 0 (not used) to 1 (best quality). If TRUE, only extract selected bits and convert to decimal system.
X	Raster* object.
keep	If NULL (default), bits are only encoded, else an integer vector of values you want to keep (becomes TRUE), the rest becomes NA. See examples.
datatype	character. Output datatype, see writeRaster.

Value

A Raster* object.

Functions

- extractBits: Extract bit-encoded information from Raster* file
- maskWater: Masks water (additional information required)

Note

makeWeights and extractBits are identical with the only difference that makeWeights does additionally convert the data into weighting information.

Author(s)

Matteo Mattiuzzi

See Also

detectBitInfo.

Examples

```
## Not run:

# example MOD13Q1 see https://lpdaac.usgs.gov/products/modis_products_table/mod13q1
# enter in Layers
# See in TABLE 2: MOD13Q1 VI Quality
# column 1 (bit) row 2 VI usefulness
bitShift = 2
bitMask = 15 # ('15' is the decimal of the binary '1111')
# or try to use
detectBitInfo("MOD13Q1") # not all products are available!
viu <- detectBitInfo("MOD13Q1","VI usefulness") # not all products are available!
viu

# simulate bit info
bit <- round(runif(10*10,1,65536))

# extract from vector
makeWeights(bit,bitShift,bitMask,decodeOnly=TRUE)
# the same as
extractBits(bit,bitShift,bitMask)

# create a Raster object
VIqual <- raster(ncol=10,nrow=10)
VIqual[] <- bit

# extract from Raster object
a <- makeWeights(VIqual,bitShift,bitMask,decodeOnly=TRUE)

# linear conversion of 0 (0000) to 15 (1111) to 1 fo 0
b <- makeWeights(VIqual,bitShift,bitMask,decodeOnly=FALSE)

threshold=6 # every thing < threshold becomes a weight = 0
c <- makeWeights(VIqual,bitShift,bitMask,threshold,decodeOnly=FALSE)

res <- round(cbind(a[],b[],c[]),2)
colnames(res) <- c("ORIG","Weight","WeightThreshold")
res

#####
# water mask
runGdal(product="MOD13A2",begin="2009001",end="2009001", extent=extent(c(-9,-3 ,54,58)),
        SDSstring="001",outDirPath="~/",job="delme") # 6.4 MB
x <- raster("~/delme/MOD13A2.A2009001.1_km_16_days_VI_Quality.tif")

res1 <- maskWater(x)
plot(res1)

res2 <- maskWater(x,keep=1) # 1 = Land (nothing else)
x11()
plot(res2)
```

```
# Land (Nothing else but land) + Ocean coastlines and lake shorelines + shallow inland Water,
# the rest becomes NA
x11()
res3 <- maskWater(x,keep=c(1,2,3))
plot(res3)

# unlink("~/delme",recursive=TRUE)

###############

# as on Linux you can read HDF4 directly you can also do:
if(.Platform$OS.type=="unix")
{
  x <- getHdf(HdfName="MOD13A2.A2009001.h17v03.005.2009020044109.hdf", wait=0) # 6.4 MB

  detectBitInfo(x) # just info
  getSds(x) # just info

  x <- getSds(x)$SDS4gdal[3] # you need 'VI Quality'
  x  <- raster(x)
  # plot(x)
  # ex <- drawExtent()
  ex <- extent(c(-580779,-200911,5974929,6529959))
  x  <- crop(x,ex) # just for speed-up

  res1 <- maskWater(x)
  plot(res1)

  res2 <- maskWater(x,keep=1) # 1 = Land (Nothing else but land), the rest becomes NA
  x11()
  plot(res2)

 # Land (Nothing else but land) + Ocean coastlines and lake shorelines + shallow inland Water,
  # the rest becomes NA
  res3 <- maskWater(x,keep=c(1,2,3))
  x11()
  plot(res3)
}

## End(Not run)
```

`extractDate` *Extract Dates from (MODIS) Files*

Description

This function helps to extract dates from a vector of files.

Usage

```
extractDate(files, pos1 = 10, pos2 = 16, asDate = FALSE,
  format = "%Y%j")
```

Arguments

files	character vector of filenames from which to extract dates.
pos1	integer, start of date string in files.
pos2	integer, end of date string.
asDate	logical. If TRUE, the result is converted to a Date object.
format	character, date format. Used only if asDate = TRUE. Defaults to MODIS date style (i.e., "%Y%j" for year and julian day). See strptime for modifications.

Value

A list with the following entries: 'inputLayerDates', 'pos1', 'pos2', 'asDate' and, optionally, 'format'. If asDate = FALSE, 'inputLayerDates' are represented as character, else as Date.

Author(s)

Matteo Mattiuzzi

Examples

```
## Not run:
# example on HDF files
files <- c("MOD13Q1.A2010209.h18v03.005.2010239071130.hdf",
           "MOD13Q1.A2010225.h18v03.005.2010254043849.hdf")
extractDate(files)
extractDate(files,asDate=TRUE)

# on any other file
files <- c("Myfile_20010101.XXX","Myfile_20010115.XXX","Myfile_20010204.XXX")
extractDate(files,pos1=8,pos2=15)
extractDate(files,pos1=8,pos2=15,asDate=TRUE,format="\%Y\%m\%d")

## End(Not run)
```

---

fileSize *Get Size of File(s)*

Description

Function for getting size of any files.

Usage

```
fileSize(file, units = "B")
```

Arguments

file	character vector of file(s) with path.
units	character, defaults to "B". Currently available options are "B", "KB", "MB", "GB" or "TB" for bites, kilo-, mega-, giga- and terabytes.

Value

numeric vector of the same length as file (in units). Note that directories are excluded.

Author(s)

Matteo Mattiuzzi

Examples

```
## Not run:
fileSize(list.files("./"))

## End(Not run)
```

---

genTile *Generate Global Tiling System*

Description

This function generates a matrix with bounding box information for a global tiling system (based on Lat/Lon).

Usage

```
genTile(tileSize = 1, offset = 0, StartNameFrom = c(0, 0),
  extent = list(xmin = -180, xmax = 180, ymin = -90, ymax = 90))
```

Arguments

tileSize	numeric, size of a single tile in degrees (EPSG:4326).
offset	numeric, shifts the tiling system in upper-left direction.
StartNameFrom	numeric. c(Lat-Direction,Lon-Direction) start number in the naming of the tiles.
extent	list. Tile system extent information, basically the coverage of the data on server.

Value

A matrix.

Author(s)

Matteo Mattiuzzi

See Also

getTile.

Examples

```
# 1x1 degree tiling system
e1 <- genTile()
head(e1)

# 10x10 degree tiling system with offset to be aligned to Geoland2 Dataset
e2 <- genTile(tileSize = 10, offset = (1/112) / 2)
head(e2)

# Tiling system for SRTMv4 data (CGIAR-CSI)
e3 <- genTile(tileSize = 5, StartNameFrom = c(1, 1),
              extent = list(xmin = -180, xmax = 180, ymin = -60,ymax = 60))
head(e3)
```

---

getCollection *Get Available Collections of MODIS Product(s)*

Description

Checks and retrieves available MODIS collection(s) for a given product.

Usage

```
getCollection(product, collection = NULL, newest = TRUE,
  forceCheck = FALSE, as = "character", quiet = TRUE, ...)
```

Arguments

product	character. MODIS grid product to check for existing collections, see getProduct.
collection	character or integer. If provided, the function only checks if the specified collection exists and returns the collection number formatted based on the as parameter or FALSE if it doesn't exists. The check is performed on the primary data source, 'LP DAAC' https://lpdaac.usgs.gov/.
newest	logical. If TRUE (default), return only the newest collection, else return all available collections.

forceCheck	logical, defaults to FALSE. If TRUE, connect to the 'LP DAAC' FTP server and get available collections, of which an updated version is permanently stored in MODIS:::combineOptions()$auxPath.
as	character, defaults to 'character' which returns the typical 3-digit collection number (i.e., "005"). as = 'numeric' returns the result as numeric (i.e., 5).
quiet	logical, defaults to TRUE.
...	Additional arguments passed to MODIS:::combineOptions.

Value

A 3-digit character or numeric object (depending on 'as') or, if length(product) > 1, a list of such objects with each slot corresponding to the collection available for a certain product. Additionally, a text file in a hidden folder located in getOption("MODIS_localArcPath") as database for future calls. If 'collection' is provided, only the (formatted) collection (or FALSE if it could not be found) is returned.

Author(s)

Matteo Mattiuzzi

See Also

getProduct.

Examples

```
## Not run:

# update or get collections for MOD11C3 and MYD11C3
getCollection(product="M.D11C3")
getCollection(product="M.D11C3",newest=FALSE)

getCollection(product="M.D11C3",collection=3)
getCollection(product="M.D11C3",collection=41)
getCollection(product="M.D11C3",collection="041")
getCollection(product="M.D11C3",forceCheck=TRUE)

## End(Not run)
```

---

getHdf *Create or Update Local Subset of Online MODIS Data Pool*

Description

Create or update a local user-defined subset of the global MODIS grid data archive. Based on user-specific parameters the function checks in the local archive for available data and downloads missing data from the online MODIS data pool. When run in a schedule job, the function manage the continuous update of the local MODIS data archive.

Usage

```
## S4 method for signature 'character'
getHdf(product, HdfName, begin = NULL, end = NULL,
  tileH = NULL, tileV = NULL, extent = NULL, collection = NULL,
  checkIntegrity = TRUE, forceDownload = TRUE, ...)

## S4 method for signature 'missing'
getHdf(HdfName, checkIntegrity = TRUE, ...)
```

Arguments

product	character. MODIS grid product to be downloaded, see getProduct. Use dot notation to address Terra and Aqua products (e.g. M.D13Q1).
HdfName	character vector or list. Full HDF file name(s) to download a small set of files. If specified, other file-related parameters (i.e., begin, end, collection, etc.) are ignored.
begin	character. Begin date of MODIS time series, see transDate for formatting.
end	character. End date, compatible with future dates for continuous updates via scheduled jobs.
tileH	numeric or character. Horizontal tile number(s), see getTile.
tileV	numeric or character. Vertical tile number(s), see tileH.
extent	See Details in getTile.
collection	character or integer. Desired MODIS product collection, see MODIS pages or getCollection for more information.
checkIntegrity	logical. If TRUE (default), the size of each downloaded file is checked. In case of inconsistencies, the function tries to re-download broken files.
forceDownload	logical. If TRUE (default), try to download data irrespective of whether online information could be retrieved via MODIS:::getStruc or not.
...	Further arguments passed to MODISoptions, eg 'wait'.

Value

An invisible vector of downloaded data and paths.

Author(s)

Matteo Mattiuzzi

References

MODIS data is obtained through the online Data Pool at the NASA Land Processes Distributed Active Archive Center (LP DAAC), USGS/Earth Resources Observation and Science (EROS) Center, Sioux Falls, South Dakota https://lpdaac.usgs.gov/get_data.

Examples

```
## Not run:
# One or more specific file (no regular erpression allowed here)
a <- getHdf(HdfName = c("MYD11A1.A2009001.h18v04.006.2015363221538.hdf",
                        "MYD11A1.A2009009.h18v04.006.2015364055036.hdf",
                        "MYD11A1.A2009017.h18v04.006.2015364115403.hdf"))
a

# Get all MODIS Terra and Aqua M*D11A1 data from 1 December 2016 up to today
# (can be ran in a sceduled job for daily archive update)
b1 <- getHdf(product = "M.D11A1", begin = "2016.12.01",
             tileH = 18:19, tileV = 4)
b1

# Same tiles with a 'list' extent
Austria <- list(xmax = 17.47, xmin = 9.2, ymin = 46.12, ymax = 49.3)
b2 <- getHdf(product = "M.D11A1", begin = "2016336", extent = Austria)
b2

# Using country boarders from 'mapdata' package
c <- getHdf(product = "M.D11A1", begin = "2016306", end = "2016335",
            extent = "Luxembourg")
c

# Interactive selection of spatial extent, see getTile()
d <- getHdf(product = "M.D11A1", begin = "2016306", end = "2016307")
d

## End(Not run)
```

`getProduct` *Check and Create Product-Related Information*

Description

On user side, it is a funtion to find the desidered product. On package site, it generates central internal information to hande files.

Usage

```
getProduct(x = NULL, quiet = FALSE)
```

Arguments

`x`	`character`. MODIS filename, product name or regular expression (for the latter, see argument `pattern` in `grep` for details). If not specified, all available products are returned.
`quiet`	`logical`, defaults to `FALSE`.

Value

An invisible `list` with usable information for other functions, see examples.

Author(s)

Matteo Mattiuzzi

Examples

```
getProduct() # list available products

# or use regular expression style
getProduct("M.D11C3")
getProduct("M*D11C")

# or get information about specific product
internal_info <- getProduct("MOD11C3", quiet = TRUE)
internal_info
```

---

getSds *List SDS Layers in an .HDF File*

Description

This function lists the names of all scientific datasets (SDS) contained in a specified MODIS grid HDF file.

Usage

```
getSds(HdfName, SDSstring = NULL, method = "gdal")
```

Arguments

`HdfName`	`character`. (Absolute) filename from which to extract SDS names.
`SDSstring`	`character`, see Value.
`method`	`character`, defaults to `"gdal"`. Caution: on Windows, the default 'GDAL' installation doesn't support HDF4 files. Install 'FWTools' or use `method = "mrt"` instead.

Value

A `list` or `character`. If `SDSstring` is provided, the function reports extracted SDS and a formatted SDSsting (e.g., "11101"). If not provided, the SDS names in `HdfName` are returned. Consult the MRT manual for details.

Author(s)

Matteo Mattiuzzi

Examples

```
## Not run:
getSds(HdfName="XXX.hdf")
getSds(HdfName="/path/XXX.hdf",method="gdal") # require GDAL (FWTools on Windows)
getSds(HdfName="/path/XXX.hdf",method="mrt") # require MRTool

## End(Not run)
```

---

getTile *Get MODIS Tile ID(s)*

Description

Get MODIS tile ID(s) for a specific geographic area.

Usage

```
getTile(x = NULL, tileH = NULL, tileV = NULL)
```

Arguments

x	Extent information, see Details. Ignored if tileH and tileV are specified.
tileH, tileV	numeric or character. Horizontal and vertical tile number(s) of the MODIS Sinusoidal grid (e.g., tileH = 1:5). If specified, no cropping is performed and the full tile(s) (if more than one then also mosaicked) is (are) processed!

Details

If x is of class (see Examples for use cases)

missing:
: If 'tileH,tileV' are specified, 'x' will be ignored. If no such tile indices are provided and 'x' is missing, a viewe

character:
: Either the country name of a map object (see map) or a valid file path of a raster image or ESRI shapefile (shp). '

Raster*:
: Spatial extent, resolution, and projection of the specified Raster* are determined automatically. This informati

Extent:
: Boundary coordinates from Extent objects are assumed to be in EPSG:4326 as well as such objects have no pr

Other:
: Spatial, sf, or map object.

Value

A `MODISextent` object.

Note

MODIS does no longer support the tile identification and automated download of MERIS and SRTM data. At least as far as the latter is concerned, easy data access is granted through `getData`.

Author(s)

Matteo Mattiuzzi, Florian Detsch

See Also

`extent`, `map`, `search4map`.

Examples

```
## Not run:
# ex 1 ############
# interactive tile selection
getTile()

# ex 2: Spatial (taken from ?rgdal::readOGR) ############
dsn <- system.file("vectors/Up.tab", package = "rgdal")[1]
Up <- rgdal::readOGR(dsn, "Up")
getTile(Up)

# ex 3: sf ############
library(mapview)
getTile(franconia)

# ex 4: tileH,tileV ############
getTile(tileH = 18:19, tileV = 4)

# ex 5: Raster* with valid CRS ############
rst1 <- raster(xmn = 9.2, xmx = 17.47, ymn = 46.12, ymx = 49.3)
getTile(rst1)

# this also works for projected data
rst3 <- projectExtent(rst1, crs = "+init=epsg:32633")
getTile(rst3)

# ex 6: Raster* without CRS or, alternatively, Extent -> treated as EPSG:4326 ############
mat2 <- matrix(seq(180 * 360), byrow = TRUE, ncol = 360)
rst2 <- raster(mat2)
getTile(rst2)
getTile(extent(rst1))

# ex 7: map names as returned by search4map() ############
getTile("Austria")
getTile(c("Austria", "Germany"))
```

```
# or search for specific map name patterns (use with caution):
m1 <- search4map("Per")
getTile(m1)

# or use 'map' objects directly (remember to use map(..., fill = TRUE)):
m2 <- map("state", region = "new jersey", fill = TRUE)
getTile(m2)

## End(Not run)
```

`lpdaacLogin` *Create File with Earthdata Login Credentials*

Description

Create a hidden .netrc file with Earthdata login credentials in your home directory. If your priority server for MODIS file download is LPDAAC (see also `MODISoptions`), these are subsequently used to automatically login to `urs.earthdata.nasa.gov` and download required files.

Usage

```
lpdaacLogin(server = "LPDAAC")
```

Arguments

`server`	`character`. MODIS file server, defaults to `"LPDAAC"` which is currently the only option available.

Value

`invisible()`.

Author(s)

Matteo Mattiuzzi and Florian Detsch

See Also

- `http://docs.opendap.org/index.php/DAP_Clients_-_Authentication#LDAP` (section 2.2)
- `https://github.com/MatMatt/MODIS/issues/10`

Examples

```
## Not run:
lpdaacLogin()

## End(Not run)
```

minorFuns *Minor MODIS Package Functions*

Description

Compendium of minor **MODIS** package-related functions.

Usage

```
search4map(pattern = "", database = "worldHires", plot = FALSE)
```

Arguments

pattern	Regular expression passed to grep.
database	character. Defaults to "worldHires", see map for available options.
plot	logical, defaults to FALSE. If TRUE, search results are displayed.

Value

A list of length 2. The first entry is the call to create the given map, whereas the second entry represents the names of areas within the search.

Functions

- search4map: Simplifies search for **mapdata**-based extents

Author(s)

Matteo Mattiuzzi

See Also

getTile, map, grep.

Examples

```
## Not run:
search4map()

search4map(pattern="USA",plot=TRUE)
search4map(database="state",plot=TRUE)?map

search4map(database="italy",pattern="Bolz",plot=TRUE)

search4map(pattern="Sicily",plot=TRUE)

## End(Not run)
```

MODISoptions *Set or Retrieve Permanent MODIS Package Options*

Description

Set or retrieve persistant **MODIS** package options (per user or systemwide). Changes here will persist through sessions and updates.

Usage

```
MODISoptions(localArcPath, outDirPath, pixelSize, outProj, resamplingType,
  dataFormat, gdalPath, MODISserverOrder, dlmethod, stubbornness, wait, quiet,
  systemwide = FALSE, save = TRUE, checkTools = TRUE)
```

Arguments

localArcPath	character, defaults to "~/MODIS_ARC". Target folder for downloaded MODIS HDF files.
outDirPath	character, defaults to "~/MODIS_ARC/PROCESSED". Target folder for results of runGdal and runMrt.
pixelSize	Output pixel size (in target reference system units) passed to runGdal and runMrt, defaults to "asIn".
outProj	Target reference system passed to runGdal and runMrt. runGdal requires a valid CRS. As for runMrt, please consult the MRT manual. Since the two pocessing methods do not have common methods, it is suggested to stick with the default settings (see Details).
resamplingType	Defaults to "NN" (Nearest Neightbour). MRT and GDAL both support c('NN', 'CC', 'BIL'). In addition, GDAL supports cubicspline and lanczos and, from GDAL >= 1.10.0 onwards, also mode and average.
dataFormat	character, defaults to "GTiff". One of getOption("MODIS_gdalOutDriver") (column 'name').
gdalPath	character. Path to gdal bin directory and more relevant for Windows users. Use MODIS:::checkTools("GDAL") to try to detect it automatically.
MODISserverOrder	character. Possible options are "LAADS" (default) and "LPDAAC" (see 'dlmethod' and 'Details'). If only one server is selected, all efforts to download data from the second server available are inhibited.
dlmethod	character, defaults to auto. See 'method' in download.file. On Unix (also Mac?), it is suggested to use "wget" or, if installed, "aria2". In order to download MODIS files from LPDAAC, please note that either wget (default) or curl must be installed and made available through the PATH environmental variable.
stubbornness	numeric. The number of retries after the target server has refused a connection. Higher values increase the chance of getting the file, but also lead to hanging functions if the server is down.

wait	numeric waiting time (in seconds) inserted after each internal online download call via download.file or getURL. Reduces the chance of FTP connection errors that frequently occur after many requests.
quiet	logical passed eg to download.file which is called from inside getHdf.
systemwide	logical. If FALSE (default), 'user'-wide settings are saved to path.expand("~/.MODIS_Opts.R"). If TRUE, write settings to 'systemwide', presumed you have write access to paste(R.home(component="etc"), '/', '.MODIS_opts.R', sep='').
save	logical. If TRUE (default), settings are permanent.
checkTools	logical, defaults to TRUE. Check if external tools (i.e., GDAL and MRT) are installed and reachable through R.

Details

These settings are permanent, easy to change and take effect immediately!

If you change default values, consider that your settings have to be valid for any MODIS product, layer and area!

It is not recommended to use

- a coordinate reference system that is not applicable globally as default for 'outProj',
- or a fixed 'pixelSize' for different products,
- or a 'resamplingType' that is not "NN".

'localArcPath' and 'outDirPath' should be changed, expecially on a Windows OS, as '~/MODIS_ARC/...' is normally on the 'C:/...' drive. You may also specify a shared network drive if you have a central MODIS data server.

On Windows, you have to set 'gdalPath' to the location of GDAL executables (i.e., the '.../GDAL../bin' directory). On Unix-alikes, this should not be required unless you want to specify a non-default GDAL installation.

On an unixoid OS, it is suggested to use dlmethod = 'wget' because it is a reliable tool and, after the change of the 'LP DAAC' datapool from FTP to HTTP (May 2013), dlmethod = 'auto' seems not to work properly. On Windows, on the other hand, dlmethod = 'auto' seems to work fine.

Please note that in order to download MODIS files from LPDAAC, you are required to register for an Earthdata Login Profile (https://urs.earthdata.nasa.gov/users/new) and create a read-only .netrc file in your home directory containing the Earthdata server address as well as your login credentials. An automated solution for the creation of a workable .netrc file is provided through lpdaacLogin.

Value

The most relevant **MODIS** options are printed to the console. Use capture.output to prevent this behavior.

Author(s)

Matteo Mattiuzzi, Steven Mosher and Florian Detsch

Examples

```
## Not run:
## get options
MODISoptions()

## set options
MODISoptions(localArcPath="/another/path/than/default")

## End(Not run)
```

---

orgStruc *Reorganise MODIS Files in Local Data Archive*

Description

Re-organise the storage structure of your MODIS archive according to the settings in `options("MODIS_arcStructure")`. Depending on the specified `'source'` , you can also use this function to gather all MODIS grid files on you computer and organise them according to '~/.MODIS_opts.R'. The main purpose is to organise the archive, but it is also possible to copy a subset of files to a desidered location!

Usage

```
orgStruc(from, to, structure, pattern, move = FALSE, quiet = FALSE)
```

Arguments

`from`	`character`. Local path to look for MODIS files, defaults to `options("MODIS_localArcPath")` (see `MODISoptions`).
`to`	`character`. Target folder to move (or copy) MODIS files to, defaults to `options("MODIS_localArcPath`
`structure`	`character`. Storage structure, defaults to `options("MODIS_arcStructure")` (see Examples).
`pattern`	Regular expression passed to `list.files`. Insert a pattern if you want to extract specific files from your archiv.
`move`	`logical`. If `TRUE` (default), files are moved and duplicated files are deleted. If `FALSE`, files are just copied and thus remain in the origin folder. Note that the copying process performs rather slowly when dealing with large files, e.g. 250-m 'MOD13Q1'.
`quiet`	`logical`, defaults to `FALSE`.

Value

If `quiet = FALSE`, information on how many files have been moved (or copied) and deleted is printed to the console.

Author(s)

Matteo Mattiuzzi

Examples

```
## Not run:
# MOVE all MODIS grid data to the directory and structure as defined in
# options("MODIS_localArcPath", "MODIS_arcStructure")
orgStruc(move = TRUE)

# COPY all MOD13Q1 from 2001 to folder "~/MyFiles/MOD13Q1.collection/"
orgStruc(pattern="MOD13Q1.A2001*.",to="~/MyFiles",structure="PRODUCT.CCC")

# COPY all MOD13Q1 to folder "~/MyFiles/"
orgStruc(pattern="MOD13Q1.*.",to="~/MyFiles",structure="")

## End(Not run)
```

---

orgTime *Handle Input and Output Dates Used for Filtering*

Description

This function lets you define the period to be filtered, the output temporal resolution, and select the required data from your input 'files'.

Usage

```
## S4 method for signature 'character'
orgTime(files, nDays = "asIn", begin = NULL,
  end = NULL, pillow = 75, pos1 = 10, pos2 = 16, format = "%Y%j")

## S4 method for signature 'Date'
orgTime(files, nDays = "asIn", begin = NULL, end = NULL,
  pillow = 75)

## S4 method for signature 'Raster'
orgTime(files, nDays = "asIn", begin = NULL,
  end = NULL, pillow = 75, pos1 = 10, pos2 = 16, format = "%Y%j")
```

Arguments

files	A character, Date, or Raster* object. Typically MODIS filenames created e.g. from runGdal or runMrt, but any other filenames holding date information are supported as well. If a Raster* object is supplied, make sure to adjust 'pos1', 'pos2', and 'format' according to its layer names.

nDays	Time interval for output layers. Defaults to "asIn" that includes the exact input dates within the period selected using begin and end. Can also be nDays = "1 month" or "1 week", see seq.Date and Examples.
begin	character. Output begin date, defaults to the earliest input dataset.
end	character. Output end date, defaults to the latest input dataset. Note that the exact end date depends on begin and nDays.
pillow	integer. Number of days added to the beginning and end of a time series.
pos1	integer, start of date string in files.
pos2	integer, end of date string.
format	character, see extractDate.

Value

A list with the following slots (to be completed):

- $inSeq
- $outSeq
- $inDoys
- $inputLayerDates
- $outputLayerDates
- $call

Author(s)

Matteo Mattiuzzi, Florian Detsch

See Also

seq.Date.

Examples

```
# Using MODIS files
files <- c("MOD13A2.A2010353.1_km_16_days_composite_day_of_the_year.tif",
           "MOD13A2.A2011001.1_km_16_days_composite_day_of_the_year.tif",
           "MYD13A2.A2010361.1_km_16_days_composite_day_of_the_year.tif",
           "MYD13A2.A2011009.1_km_16_days_composite_day_of_the_year.tif")

orgTime(files)
orgTime(files,nDays=2,begin="2010350",end="2011015")

## Not run:
# Using other files, e.g. from GIMMS (Jul 1981 to Dec 1982)
library(gimms)

files.v1 <- system.file("extdata/inventory_ecv1.rds", package = "gimms")
files.v1 <- readRDS(files.v1)[1:3]
```

```
dates.v1 <- monthlyIndices(files.v1, timestamp = TRUE)

orgTime(dates.v1)

## End(Not run)
```

---

preStack *Organise (MODIS) Files in Preparation for Stacking*

Description

This function lets you sort a vector of filenames according to date. It is thought to be used on results from runGdal or runMrt.

Usage

```
preStack(pattern = "*", path = "./", files = NULL, timeInfo = NULL)
```

Arguments

pattern	Regular expression passed to list.files
path	character. Location of MODIS files to stack.
files	character vector of filenames. If provided, arguments pattern and path are ignored.
timeInfo	Ouput from orgTime.

Value

A character vector of filenames within the query. If timeInfo is provided, filenames are sorted and subsetted by date.

Author(s)

Matteo Mattiuzzi

Examples

```
# see Examples in ?smooth.spline.raster
```

reformatDOY *Reformat MODIS "composite_day_of_the_year" SDS*

Description

In order to create custom temporal aggregation levels (e.g., half-monthly, monthly) from native 16-day MODIS composites, a convenient representation of the pixel-wise acquisition date is urgently required. Since the MODIS "composite_day_of_the_year" SDS merely includes the day of the year (DOY), but not the year itself, this function creates complete date information from both the respective MODIS layer name and the pixel-wise DOY information.

Usage

```
reformatDOY(x, cores = 1L, ...)
```

Arguments

x	`character` or `Raster*`. MODIS "composite_day_of_the_year" layer(s).
cores	`integer`. Number of cores for parallel processing.
...	Additional arguments passed to `extractDate`.

Value

A `Raster*` object.

Author(s)

Florian Detsch

See Also

`repDoy`.

Examples

```
## Not run:
runGdal("MOD13Q1", collection = getCollection("MOD13Q1", forceCheck = TRUE),
        begin = "2000353", end = "2000366", extent = "Luxembourg",
        job = "reformatDOY", SDSstring = "000000000010")

fls <- list.files(paste0(getOption("MODIS_outDirPath"), "/reformatDOY"),
                  pattern = "day_of_the_year.tif$", full.names = TRUE)

## raw doy
raw <- raster(fls)
unique(raw[])

## reformatted dates
```

```
rfm <- reformatDOY(cmp)
unique(rfm[])

## End(Not run)
```

---

repDoy *Repair MODIS "composite_day_of_the_year" SDS*

Description

Currently works only for MODIS 16 days composites! In MODIS composites, the Julian dates inside the 'composite_day_of_the_year' SDS are referring always to the year they are effectively in. The problem is that the layer/SDS name from the last files from Terra and Aqua within a year can include dates from the following year and so starting again with 1. The problem comes if you want to sort values of a time series by date (e.g. for precise time series functions). This function generates a sequential vector beginning always with the earielst SDS/layer date and ending with the total sum of days of the time serie length.

Usage

```
repDoy(pixX, layerDate = NULL, bias = 0)
```

Arguments

pixX	matrix of values, usually derived from as.matrix.
layerDate	If NULL (default), try to autodetect layer dates. If you want to be sure, use the result from extractDate(..., asDate = TRUE) or orgTime.
bias	integer. Bias applied to all values in pixX.

Value

A matrix with sequential Julian dates.

Author(s)

Matteo Mattiuzzi

Examples

```
## Not run:
runGdal(product="M.D13A2", begin="2010350", end="2011016", extent="Luxembourg",
        job="deleteme", SDSstring="100000000010")

ndviFiles <- preStack(path=paste(MODIS:::.getDef()$outDirPath,"deleteme",sep="/")
                      ,pattern="*_NDVI.tif$")

ndvi <- stack(ndviFiles)
```

```
doyFiles <- preStack(path=paste(MODIS:::.getDef()$outDirPath,"deleteme",sep="/")
                     ,pattern="*_composite_day_of_the_year.tif$")

doy <- stack(doyFiles)
layerDates <- extractDate(names(doy))

pixX <- 169

y <- ndvi[pixX]
x1 <- doy[pixX]
x2 <- repDoy(doy[pixX],layerDates)

x1
x2
# the plotting example is not really good.
# To create a figurative example it would be necessary to dolwnload to much data!
plot("",xlim=c(1,max(x1,x2)),ylim=c(0,2000),xlab="time",ylab="NDVI*10000")
lines(y=y,x=x1,col="red",lwd=3)
lines(y=y,x=x2,col="green",lwd=2)

# repDoy function is thought to be enbeded in something like that:
tr <- blockSize(ndvi)

doyOk <- brick(doy)
doyOk <- writeStart(doyOk, filename='test.tif',  overwrite=TRUE)

for (i in 1:tr$n)
{
  pixX  <- getValues(doy,tr$row[i],tr$nrows[i])
  ok    <- repDoy(pixX,layerDates)
  doyOk <- writeValues(x=doyOk,v=ok,start=tr$row[i])
}
doyOk <- writeStop(doyOk)

# unlink(filename(doyOk))

## End(Not run)
```

---

runGdal *Process MODIS HDF with GDAL*

Description

Downloads MODIS grid data from archive (FTP or local) and processes the files.

Usage

```
runGdal(product, collection = NULL, begin = NULL, end = NULL,
```

```
extent = NULL, tileH = NULL, tileV = NULL, SDSstring = NULL,
job = NULL, checkIntegrity = TRUE, forceDownload = TRUE,
overwrite = FALSE, ...)
```

Arguments

product	character, see getProduct.
collection	character or integer, see getCollection.
begin	character. Begin date of MODIS time series, see transDate for formatting.
end	Same for end date.
extent	Extent information, defaults to 'global'. See getTile.
tileH	numeric or character. Horizontal tile number, see getTile.
tileV	numeric or character. Vertical tile number(s), see tileH.
SDSstring	character, see getSds.
job	character. Name of the current job for the creation of the output folder. If not specified, it is created in 'PRODUCT.COLLECTION_DATETIME'.
checkIntegrity	logical, see getHdf.
forceDownload	logical, see getHdf.
overwrite	logical, defaults to FALSE. Determines whether or not to overwrite existing SDS output files.
...	Additional arguments passed to MODIS:::combineOptions() (eg 'wait'), see also MODISoptions.

Details

outProj	CRS/ prj4 or EPSG code of output, any format supported by gdal see examples. Default is 'asIn' (no warping). See ?MODISoptions.
pixelSize	Numeric single value. Output pixel size in target reference system unit. Default is 'asIn'. See ?MODISoptions.
resamplingType	Character. Default is 'near', can be one of: 'bilinear', 'cubic', 'cubicspline', 'lanczos'. See ?MODISoptions.
blockSize	integer. Default NULL that means the stripe size is set by GDAL. Basically it is the "-co BLOCKYSIZE=" parameter. See: http://www.gdal.org/frmt_gtiff.html
compression	logical. Default is TRUE, compress data with the lossless LZW compression with "predictor=2". See: http://www.gdal.org/frmt_gtiff.html
dataFormat	Data output format, see getOption("MODIS_gdalOutDriver") column 'name'.
localArcPath	Character. See ?MODISoptions. Local path to look for and/or to download MODIS files.
outDirPath	Character. See ?MODISoptions. Root directory where to write job folder.

runGdal uses a lot of **MODIS** package functions, see in section Arguments and Methods the respective '?function' for details and inputs.
If extent is a Raster* object, the output has exactly the same extent, pixel size, and projection.

If `extent` is a **sp** object (i.e., polygon shapefile), the output has exactly the same extent and projection.
If `tileH` and `tileV` are used (instead of `extent`) to define the area of interest, and `outProj` and `pixelSize` are `'asIn'`, the result is only converted from multilayer-HDF to `dataFormat`, default "GeoTiff" (`MODISoptions`).

Value

A `list` of the same length as 'product'. Each product slot holds a sub-`list` of processed dates which, for each time step, include the corresponding output files as `character` objects.

Note

You need to have a GDAL installed on your system!
http://www.gdal.org/gdal_utilities.html

On Unix-alkes, install 'gdal-bin' (i.e. Ubuntu: 'sudo apt-get install gdal-bin')
On Windows, you need to install GDAL through OSGeo4W (http://trac.osgeo.org/osgeo4w/) or FWTools (http://fwtools.maptools.org/) since the standard GDAL does not support HDF4 format.

Author(s)

Matteo Mattiuzzi, Florian Detsch

See Also

`getHdf`, `runMrt`.

Examples

```
## Not run:
# LST in Austria
runGdal( product="MOD11A1", extent="austria", begin="2010001", end="2010005", SDSstring="101")

# LST with interactiv area selection
runGdal( product="MOD11A1", begin="2010001", end="2010005", SDSstring="101")

### outProj examples
# LST of Austria warped to UTM 34N (the three different possibilites to specify "outProj")
# to find am EPSG or prj4 you may use: prj <- make_EPSG() See
runGdal( job="LSTaustria", product="MOD11A1", extent="Austria", begin="2010001", end="2010005",
         SDSstring="101", outProj="EPSG:32634")

runGdal( job="LSTaustria", product="MOD11A1", extent="Austria", begin="2010001", end="2010005",
         SDSstring="101", outProj="32634")

runGdal( job="LSTaustria", product="MOD11A1", extent="Austria", begin="2010001", end="2010005",
       SDSstring="101", outProj="+proj=utm +zone=34 +ellps=WGS84 +datum=WGS84 +units=m +no_defs")
```

```
### resamplingType examples
runGdal( job="LSTaustria", product="MOD11A1", extent="Austria", begin="2010001", end="2010005",
         SDSstring="1", resamplingType="lanczos", outProj="32634", pixelSize=100)

### processing entire tiles and keeping Sinusoidal projection
# This corresponds to a format conversion (eos-hdf04 to Geotiff) and
# layer extraction (multi-layer to single layer)
runGdal( job="LSTaustria", product="MOD11A1", tileH=18:19,tileV=4, begin="2010001", end="2010005",
         SDSstring="1", outProj="asIn")


## End(Not run)
```

---

runMrt *Run MODIS Reprojection Tool with Specified Parameters*

Description

Specifying input parameters, this function gets MODIS grid data from the archive (HTTP/FTP or local) and processes them with the 'MRT-grid' tool. See also the 'MRT' manual, available online via https://lpdaac.usgs.gov/sites/default/files/public/mrt41_usermanual_032811.pdf, for further information.

Usage

```
runMrt(...)
```

Arguments

...	See Details.

Details

product	See getProduct.
begin	See transDate.
end	See transDate.
extent	See getTile.
SDSstring	See getSds. Default is to extract all SDS.
job	character. Name of the current job for the creation of the output folder. If not specified, it is created in '
localArcPath	character. Defaults to options("MODIS_localArcPath"). Local path to look for and/or download MO
outDirPath	character. Defaults to options("MODIS_outDirPath"). Root directory where to write job folder.
dataType	character, defaults to 'GeoTiff' (see MODISoptions. 'MRT' supports: "raw binary" (hdr+dat), "HDF
outProj	character, see 'MRT' manual.
zone	Optional UTM zone number when outProj = "UTM". If not set, it is autodetected. See 'MRT' manual.
projPara	character in the form "6371007.18 0.00 0.00 ...". For outProj %in% c("GEO","SIN"), it is autodetect
datum	character, defaults to 'NODATUM'. See 'MRT' manual.
mosaic	logical, defaults to TRUE. Mosaic files or not? One case for setting mosaic=FALSE is a too large extent

anonym	logical, defaults to TRUE. If FALSE, the job name is added at the end of the root filename.
quiet	logical, defaults to FALSE. It is up to you to switch to 'boring' alias FALSE. Not fully implemented!
dlmethod	default options("MODIS_dlmethod"). Argument passed to download.file (see MODISoptions).
stubbornness	Default is options("MODIS_stubborness"). See ?MODISoptions

Author(s)

Matteo Mattiuzzi and Forrest Stevens

Source

You can obtain MRT-grid after registration from: https://lpdaac.usgs.gov/tools/modis_reprojection_tool.

See Also

getHdf.

Examples

```
## Not run:
runMrt( product="MOD11A1", extent="austria", begin="2010001", end="2010002", SDSstring="101",
        job="ExampleGEOdelme", outProj="GEO")
runMrt( product="MOD11A1", extent="austria", begin="2010001", end="2010002", SDSstring="101",
        job="ExampleSINdelme", outProj="SIN")
runMrt( product="MOD11A1", extent="austria", begin="2010001", end="2010002", SDSstring="101",
        job="ExampleUTMdelme", outProj="UTM")

## End(Not run)
```

---

smooth.spline.raster *Filter Time Series Imagery with a Cubic Spline*

Description

This function uses the smooth.spline function to filter a vegetation index time serie of satellite data.

Usage

```
smooth.spline.raster(x, w = NULL, t = NULL, groupYears = TRUE,
  timeInfo = orgTime(x), df = 6, outDirPath = "./", ...)
```

Arguments

x	RasterBrick (or RasterStack) or character vector of filenames, sorted 'Vegetation index'.
w	RasterBrick (or RasterStack) with weighting information, e.g. derived from makeWeights.
t	In case of MODIS composite, the corresponding 'composite_day_of_the_year' RasterBrick (or RasterStack).
groupYears	logical. If TRUE, output files are grouped by years.
timeInfo	Result from orgTime.
df	numeric, yearly degree of freedom value passed to smooth.spline. If set as character (i.e., df = "6"), it is not adapted to the time serie length but used as a fixed value (see Details).
outDirPath	Output path, defaults to the current working directory.
...	Arguments passed to writeRaster. Note that filename is created automatically.

Details

numeric values of df (e.g., df = 6) are treated as yearly degrees of freedom. Here, the length of the input time series is not relevant since df is adapted to it with: df*('length of _input_ timeserie in days'/365). The input length can differ from the output because of the pillow argument in orgTime.

character values of df (e.g., df = "6"), on the other hand, are not adopted to the length of the input time series.

Currently tested on MODIS and Landsat data. Using M*D13 data, it is also possible to use the 'composite_day_of_the_year' layer and the 'VI_Quality' layer. This function is currently under heavy development and a lot of changes are expected to come soon.

Value

The filtered data and a text file with the dates of the output layers.

Author(s)

Matteo Mattiuzzi

See Also

whittaker.raster, raster.

Examples

```
## Not run:
# The full capacity of the following functions is currently avaliable only with M*D13 data.
# !! The function is very new, double check the result!!

# You need to extract the: 'vegetation index', 'VI_Quality layer',
# and 'composite day of the year' layer.
```

```
# runGdal(product="MOD13A2",begin="2004340",extent="sicily",end="2006070",
# job="fullCapa",SDSstring="101000000010")
# Afterward extract it to:
options("MODIS_outDirPath")

# the only obligatory dataset is "x" (vegetatino index), get the 'vi' data on the source directory:
path <- paste0(options("MODIS_outDirPath"),"/fullCapa")
vi <- preStack(path=path, pattern="*_NDVI.tif$")

# "orgTime" detects timing information of the input data and generates based on the arguments
# the output date information. For spline functions (in general) the beginning and
# the end of the time series is always problematic.
# So there is the argument "pillow" (default 75 days) that adds
# (if available) some more layers on the two endings.

timeInfo <- orgTime(vi,nDays=16,begin="2005001",end="2005365",pillow=40)

# now re-run "preStack" with two diferences, 'files' (output of the first 'preStack' call)
# and the 'timeInfo'.
# Here only the data needed for the filtering is extractet:
vi <- preStack(files=vi,timeInfo=timeInfo)

smooth.spline.raster(x=vi,timeInfo=timeInfo)

# Filter with weighting and time information:
# if the files are M*D13 you can use also Quality layers and the composite day of the year:
w <- stack(preStack(path=path, pattern="*_VI_Quality.tif$", timeInfo=timeInfo))
w <- makeWeights(w,bitShift=2,bitMask=15,threshold=6)
# you can also pass only the names
t <- preStack(path=path, pattern="*_composite_day_of_the_year.tif$", timeInfo=timeInfo)

smooth.spline.raster(x=vi,w=w,t=t,timeInfo=timeInfo)

## End(Not run)
```

`temporalComposite` *Calculate MODIS Composite Images*

Description

Based on a user-defined function, e.g. `max` for maximum value composites (MVC), aggregate native 16-day MODIS datasets to custom temporal composites.

Usage

```
temporalComposite(x, y, timeInfo = extractDate(x, asDate =
  TRUE)$inputLayerDates, interval = c("month", "year", "fortnight"),
  fun = max, na.rm = TRUE, cores = 1L, filename = "", ...)
```

Arguments

x	Raster* or character. MODIS vegetation index.
y	Raster* or character. MODIS "composite_day_of_the_year" SDS associated with 'x'.
timeInfo	Date vector corresponding to all input layers. If not further specified, this is tried to be created through invoking extractDate upon 'x', assuming standard MODIS file names.
interval	character. Time period for aggregation, see aggInterval.
fun, na.rm	function. See overlay.
cores	integer. Number of cores for parallel processing.
filename	character. Optional output filename.
...	Additional arguments passed to writeRaster.

Value

A Raster* object.

Author(s)

Florian Detsch

See Also

aggInterval, calc, writeRaster.

Examples

```
## Not run:
library(mapview)
frc <- as(subset(franconia, district == "Mittelfranken"), "Spatial")
tfs <- runGdal("MOD13A1", begin = "2015001", end = "2016366", extent = frc,
            job = "temporalComposite", SDSstring = "100000000010")

ndvi <- sapply(tfs[[1]], "[[", 1)
cdoy <- sapply(tfs[[1]], "[[", 2)

mmvc <- temporalComposite(ndvi, cdoy)
plot(mmvc[[1:4]])

## End(Not run)
```

transDate *MODIS Date Conversion and Testing*

Description

This function converts a sequence of input dates to 'YYYY-MM-DD' and 'YYYYDDD'.

Usage

```
transDate(begin = NULL, end = NULL)
```

Arguments

begin, end	character or Date. Begin (end) date of MODIS time series, see Note. If not provided, this defaults to "1972-01-01" (Sys.Date()).

Value

A list of begin and end dates formatted according to 'YYYY-MM-DD' (first two slots; class Date) and 'YYYYDDD' (second two slots; class character).

Note

If input dates are supplied as character, this function either expects 7-digit strings in the MODIS intrinsic form '%Y%j' or, alternatively, 10-digit strings in the form '%Y-%m-%d' where the two field separators need to be uniform (see Examples).

Author(s)

Matteo Mattiuzzi, Florian Detsch

See Also

strptime.

Examples

```
transDate()
transDate(begin = "2009.01.01") # ends with current date
transDate(end = "2009.01.01") # starts with Landsat 1
transDate(begin = c("2009-01-01", "2010-01-01"), end = "2011.03.16")
```

`whittaker.raster` *Filter Vegetation Index with Modified Whittaker Approach*

Description

Use a modified Whittaker filter function (see References) from package **ptw** to filter a vegetation index (VI) time serie of satellite data.

Usage

```
whittaker.raster(vi, w = NULL, t = NULL, timeInfo = orgTime(vi),
  lambda = 5000, nIter = 3, outputAs = "single", collapse = FALSE,
  prefixSuffix = c("MCD", "ndvi"), outDirPath = ".",
  outlierThreshold = NULL, mergeDoyFun = "max", ...)
```

Arguments

`vi`	`Raster*` or `character` filenames, sorted VI. Use `preStack` functionality to ensure the right input.
`w`	`Raster*` or `character` filenames. In case of MODIS composite, the sorted 'VI_Quality' layers.
`t`	`Raster*` or `character` filenames. In case of MODIS composite, the sorted 'composite_day_of_the_year' layers. If missing, the date is determined using `timeInfo`.
`timeInfo`	Output from `orgTime`.
`lambda`	`character` or `integer`. Yearly lambda value passed to `whit2`. If set as `character` (i.e., `lambda = "600"`), it is not adapted to the time serie length, but used as a fixed value (see Details). High values = stiff/rigid spline.
`nIter`	`integer`. Number of iteration for the upper envelope fitting.
`outputAs`	`character`, organisation of output files. `"single"` (default) means each date one `RasterLayer`; `"yearly"` a `RasterBrick` for each year, and `"one"` one `RasterBrick` for the entire time series.
`collapse`	`logical`. Collapse input data of multiple years into one single year before filtering.
`prefixSuffix`	`character`, file naming. Names are composed dot-separated: `paste0(prefixSuffix[1], "YYYDDD", l`
`outDirPath`	`character`, output path. Defaults to the current working directory.
`outlierThreshold`	`numeric` in the same unit as `vi`, used for outlier removal (see Details).
`mergeDoyFun`	Especially when using `collapse = TRUE`, multiple measurements for one day can be present. Here you can choose how those values are merged to one single value: `"max"` uses the highest value, `"mean"` or `"weighted.mean"` use the `mean` if no weighting `"w"` is available, and `weighted.mean` if it is.
`...`	Arguments passed to `writeRaster` (except for `filename`).

Details

The argument `lambda` is passed to `MODIS:::miwhitatzb1`. You can set it as yearly `lambda`, which means that it doesn't matter how long the input time serie is because `lambda` is adapted to it with: `lambda*('length of input timeserie in days'/365)`. The input length can differ from the output because of the `pillow` argument in `orgTime`. But it can also be set as `character` (i.e., `lambda = "1000"`). In this case, the adaption to the time series length is not performed.

Value

A Whittaker-smoothened `RasterStack`.

Note

Currently tested on MODIS and Landsat data. Using M*D13, it is also possible to use the 'composite_day_of_the_year' and the 'VI_Quality' layers. Note that this function is currently under heavy development.

Author(s)

Matteo Mattiuzzi and Agustin Lobo

References

Modified Whittaker smoother, according to Atzberger & Eilers 2011 International Journal of Digital Earth 4(5):365-386.
Implementation in R: Agustin Lobo 2012

See Also

`smooth.spline.raster`, `raster`.

Examples

```
## Not run:
# The full capacity of the following functions is currently avaliable only with M*D13 data.
# !! The function is very new, double check the result !!

# You need to extract: 'vegetation index', 'VI_Quality layer', and 'composite day of the year'.
# runGdal(product="MOD13A2",begin="2004340",extent="sicily",end="2006070",
# job="fullCapa",SDSstring="101000000010")
# You can download this dataset from (2.6 MB):
path <- paste0(options("MODIS_outDirPath"),"fullCapa")

# the only obligatory dataset is the vegetatino index
# get the 'vi' data in the source directory:
vi <- preStack(path=path, pattern="*_NDVI.tif$")

# "orgTime" detects timing information of the input data and generates based on the arguments
# the output date information.
# For spline functions (in general) the beginning and the end of the time series
```

```
# is always problematic. So there is the argument "pillow" (default 75 days) that adds
# (if available) some more layers on the two endings.

timeInfo <- orgTime(vi,nDays=16,begin="2005001",end="2005365",pillow=40)

# now re-run "preStack" with two differences, 'files' (output of the first 'preStack' call)
# and the 'timeInfo'
# Here only the data needed for the filtering is extracted:
vi <- preStack(files=vi,timeInfo=timeInfo)

whittaker.raster(vi,timeInfo=timeInfo,lambda=5000)

# if the files are M*D13 you can use also Quality layers and the composite day of the year:
wt <- preStack(path=path, pattern="*_VI_Quality.tif$", timeInfo=timeInfo)
# can also be already stacked:
inT <- preStack(path=path, pattern="*_composite_day_of_the_year.tif$", timeInfo=timeInfo)

whittaker.raster(vi=vi, wt=wt, inT=inT, timeInfo=timeInfo, lambda=5000, overwrite=TRUE)

## End(Not run)
```

Index