Package 'modiscloud'

February 20, 2015

Type Package

Title R tools for processing Level 2 Cloud Mask products from MODIS

Version 0.14

Date 2013-02-08

Author Nicholas J. Matzke, Dept. of Integrative Biology, U.C. Berkeley

Maintainer Nicholas J. Matzke <matzke@berkeley.edu>

Depends date, sp, sfsmisc, raster, rgdal

Description Package for processing downloaded MODIS Cloud Product HDF files and derived files. Specifically, MOD35_L2 cloud product files, and the associated MOD03 geolocation files (for MODIS-TERRA); and MYD35_L2 cloud product files, and the associated MYD03 geolocation files (for MODIS-AQUA). The package will be most effective if the user installs MRTSwath (MODIS Reprojection Tool for swath products; https://lpdaac.usgs.gov/tools/modis_reprojection_tool_swath), and adds the directory with the MRTSwath executable to the default R PATH by editing ~/.Rprofile.

URL http://phylo.wikidot.com/modiscloud

License GPL (>= 2)

LazyLoad yes

ByteCompile true

Repository CRAN

Date/Publication 2013-02-09 07:43:26

NeedsCompilation no

R topics documented:

`modiscloud-package` *Process MODIS cloud mask product files to TIF*

Description

Process MODIS cloud mask product files to TIF, and then extract data

Details

Package:	modiscloud
Type:	Package
Version:	0.14
Date:	2013-02-08
License:	GPL (>= 2)
LazyLoad:	yes

This package helps the user process downloaded MODIS cloud product HDF files to TIF, and then extract data. Specifically, MOD35_L2 cloud product files, and the associated MOD03 geolocation files (for MODIS-TERRA); and MYD35_L2 cloud product files, and the associated MYD03 geolocation files (for MODIS-AQUA).

The package will be most effective if the user installs MRTSwath (MODIS Reprojection Tool for swath products; https://lpdaac.usgs.gov/tools/modis_reprojection_tool_swath), and adds the directory with the MRTSwath executable to the default R PATH by editing `~/.Rprofile`.

Each MOD35_L2/MYD35_L2 file requires a corresponding MOD03/MYD03 geolocation file to be successfully processed with the MRTSwath tool.

MRTSwath is the MRT (MODIS Reprojection Tool) for the MODIS level 1 and level 2 products (cloud mask is level 2, I think).

A few example MODIS Cloud Product files, and derived TIFs, are found in the data-only package `modiscdata`. These were too big to put in the main package, according to CRAN repository policies (http://cran.r-project.org/web/packages/policies.html).

Note: This code was developed for the following publication. Please cite if used: Goldsmith, Gregory; Matzke, Nicholas J.; Dawson, Todd (2013). "The incidence and implications of clouds for cloud forest plant water relations." Ecology Letters, 16(3), 207-314. DOI: http://dx.doi.org/10.1111/ele.12039

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

https://lpdaac.usgs.gov/get_data/

NASA (2001). "MODLAND Product Filename Convention." <URL: http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename>.

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

check_for_matching_geolocation_files

Examples

```
# Test function for checking roxygen2, roxygenize package documentation building
is.pseudoprime(13, 4)

# Some MODIS files are stored in this package's "extdata/" directory
# Here are some example MODIS files in modiscloud/extdata/
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
```

```
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
moddir = system.file("extdata/2002raw/", package="modiscdata")

# This directory actually has MYD files (from the MODIS-AQUA platform)
# (*will* work with the default files stored in modiscloud/extdata/2002raw/)
list.files(path=moddir, pattern="MYD")

# Check for matches (for MODIS-AQUA platform)
# (*will* work with the default files stored in modiscloud/extdata/2002raw/)
fns_df = check_for_matching_geolocation_files(moddir=moddir, modtxt="MYD35_L2", geoloctxt="MYD03", return_geolo


## End(Not run)


#######################################################
# Run MRTSwath tool "swath2grid"
#######################################################

# Source MODIS files (both data and geolocation)
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
# install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
moddir = system.file("extdata/2002raw/", package="modiscdata")

# Get the matching data/geolocation file pairs
fns_df = check_for_matching_geolocation_files(moddir, modtxt="MYD35_L2", geoloctxt="MYD03")
fns_df

# Resulting TIF files go in this directory
tifsdir = getwd()


# Box to subset
ul_lat = 13
ul_lon = -87
lr_lat = 8
lr_lon = -82

for (i in 1:nrow(fns_df))
{
```

```
prmfn = write_MRTSwath_param_file(prmfn="tmpMRTparams.prm", tifsdir=tifsdir, modfn=fns_df$mod35_L2_fns[i], geol
print(scan(file=prmfn, what="character", sep="\n"))

run_swath2grid(mrtpath="swath2grid", prmfn="tmpMRTparams.prm", tifsdir=tifsdir, modfn=fns_df$mod35_L2_fns[i], g

}

tiffns = list.files(tifsdir, pattern=".tif", full.names=TRUE)
tiffns


# For some unit testing etc., swath2grid may not be available.  If so, use the default TIFs:
if (length(tiffns) == 0)
{
library(modiscdata)
tifsdir = system.file("extdata/2002tif/", package="modiscdata")
tiffns = list.files(tifsdir, pattern=".tif", full.names=TRUE)
}

#########################################################
# Load a TIF
#########################################################
library(rgdal) # for readGDAL

# numpixels in subset
xdim = 538
ydim = 538


# Read the grid and the grid metadata
coarsen_amount = 1
xdim_new = xdim / floor(coarsen_amount)
ydim_new = ydim / floor(coarsen_amount)

fn = tiffns[1]
grd = readGDAL(fn, output.dim=c(ydim_new, xdim_new))

grdproj = CRS(proj4string(grd))
grdproj
grdbbox = attr(grd, "bbox")
grdbbox




############################
# Extract values from a particular pixel
############################
# Greg's field site
greglat = 10.2971
greglon = -84.79282
```

```
grdr = raster(grd)

# Input the points x (longitude), then y (latitude)
point_to_sample = c(greglon, greglat)
xycoords = adf(matrix(data=point_to_sample, nrow=1, ncol=2))
names(xycoords) = c("x", "y")

xy = SpatialPoints(coords=xycoords, proj4string=grdproj)
#xy = spsample(x=grd, n=10, type="random")
pixelval = extract(grdr, xy)

# Have to convert to 8-bit binary string, and reverse to get the count correct
# (also reverse the 2-bit strings in the MODIS Cloud Mask table)
pixelval = rev(t(digitsBase(pixelval, base= 2, 8)))
print(pixelval)


## End(Not run)


#########################################################
# Load a TIF
#########################################################
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
# install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
tifsdir = system.file("extdata/2002tif/", package="modiscdata")
tiffns = list.files(tifsdir, pattern=".tif", full.names=TRUE)
tiffns

library(rgdal) # for readGDAL

# numpixels in subset
xdim = 538
ydim = 538


# Read the grid and the grid metadata
coarsen_amount = 1
xdim_new = xdim / floor(coarsen_amount)
ydim_new = ydim / floor(coarsen_amount)

fn = tiffns[1]
grd = readGDAL(fn, output.dim=c(ydim_new, xdim_new))

grdproj = CRS(proj4string(grd))
```

```
grdproj
grdbbox = attr(grd, "bbox")
grdbbox

#########################################################
# Extract a particular bit for all the pixels in the grid
#########################################################
bitnum = 2
grdr_vals_bits = get_bitgrid(grd, bitnum)
length(grdr_vals_bits)
grdr_vals_bits[1:50]

#########################################################
# Extract a particular pair of bits for all the pixels in the grid
#########################################################
bitnum = 2
grdr_vals_bitstrings = get_bitgrid_2bits(grd, bitnum)
length(grdr_vals_bitstrings)
grdr_vals_bitstrings[1:50]


## End(Not run)

#########################################################
# Load some bit TIFs (TIFs with just the cloud indicators extracted)
# and add up the number of cloudy days, out of the total
# number of observation attempts
#########################################################
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
# install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
tifsdir = system.file("extdata/2002bit/", package="modiscdata")
tiffns = list.files(tifsdir, pattern=".tif", full.names=TRUE)
tiffns

library(rgdal) # for readGDAL

# numpixels in subset
xdim = 538
ydim = 538

# Read the grid and the grid metadata
coarsen_amount = 1
xdim_new = xdim / floor(coarsen_amount)
ydim_new = ydim / floor(coarsen_amount)
```

```
sum_nums = NULL
for (j in 1:length(tiffns))
{
fn = tiffns[j]

grd = readGDAL(fn, output.dim=c(ydim_new, xdim_new))

grdr = raster(grd)
pointscount_on_SGDF_points = coordinates(grd)
grdr_vals = extract(grdr, pointscount_on_SGDF_points)

# Convert to 1/0 cloudy/not
data_grdr = grdr_vals
data_grdr[grdr_vals > 0] = 1

grdr_cloudy = grdr_vals
grdr_cloudy[grdr_vals < 4] = 0
grdr_cloudy[grdr_vals == 4] = 1

# Note: Don't run the double-commented lines unless you want to collapse different bit values.
# grdr_clear = grdr_vals
# grdr_clear[grdr_vals == 4] = 0
# grdr_clear[grdr_vals == 3] = 1
# grdr_clear[grdr_vals == 2] = 1
# grdr_clear[grdr_vals == 1] = 1
# grdr_clear[grdr_vals == 0] = 0
#

if (j == 1)
{
sum_cloudy = grdr_cloudy
#sum_not_cloudy = grdr_clear
sum_data = data_grdr
} else {
sum_cloudy = sum_cloudy + grdr_cloudy
sum_data = sum_data + data_grdr
}

}


# Calculate percentage cloudy
sum_nums = sum_cloudy / sum_data

grd_final = numslist_to_grd(numslist=sum_nums, grd=grd, ydim_new=ydim_new, xdim_new=xdim_new)

# Display the image (this is just the sum of a few images)
image(grd_final)


## End(Not run)
```

adf *Convert to data.frame, without factors*

Description

Shortcut for: `as.data.frame(x, row.names=NULL, stringsAsFactors=FALSE)`

Usage

```
adf(x)
```

Arguments

x	matrix or other object transformable to data.frame

Details

This function, and adf2, are useful for dealing with errors due to automatic conversion of some columns to factors. Another solution may be to prepend `options(stringsAsFactors = FALSE)` at the start of one's script, to turn off all default stringsAsFactors silliness.

Value

data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

adf2

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
adf(x)
```

adf2 *Convert to data.frame, without factors*

Description

Shortcut for: `tmp_rownames = 1:nrow(x); as.data.frame(x, row.names=tmp_rownames, stringsAsFactors=FAL`

Usage

```
adf2(x)
```

Arguments

x	matrix or other object transformable to data.frame

Details

This function, and adf2, are useful for dealing with errors due to automatic conversion of some columns to factors. Another solution may be to prepend `options(stringsAsFactors = FALSE)` at the start of one's script, to turn off all default stringsAsFactors silliness.

In adf2, rownames are forced to be numbers; this can prevent errors due to e.g. repeated rownames after an `rbind` operation.

Value

data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

adf

Examples

```
x = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
adf2(x)
```

byteint2bit *Convert a byte integer (0-255) to a list of 8 bits*

Description

MODIS HDFs converted to tifs result in each pixel having a integer value from 0 to 255. byteint2bit converts this to a string of 8 bits (0/1 values).

Usage

```
byteint2bit(intval, reverse = TRUE)
```

Arguments

intval,	an integer between 0 and 255
reverse,	a logical (TRUE/FALSE) indicating whether or not the string of bits should be reversed. Default: TRUE

Details

In the case of MODIS, the bits are read from RIGHT to LEFT, which is unnatural, so by default the function uses rev() to reverse the order of reading.

Note: the reversal means that, when interpreting the two 2-bit strings in the MODIS image, the interpretation of 01 and 10 is reversed from in the MODIS documentation.

Value

byte_in_binary, the 8 bits (0/1 values) in the correct order

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

extract_bit

byteint2bit_list

Examples

```
byteint2bit(intval=0, reverse=FALSE)
byteint2bit(intval=0, reverse=TRUE)
byteint2bit(intval=1, reverse=FALSE)
byteint2bit(intval=1, reverse=TRUE)
byteint2bit(intval=254, reverse=FALSE)
byteint2bit(intval=254, reverse=TRUE)
byteint2bit(intval=255, reverse=FALSE)
byteint2bit(intval=255, reverse=TRUE)
```

---

byteint2bit_list *Convert a list of byte integer (0-255) to a table of 8 bits per row*

Description

MODIS HDFs converted to tifs result in each pixel having a integer value from 0 to 255. byteint2bit converts this to a string of 8 bits (0/1 values).

Usage

```
byteint2bit_list(grdr_vals, reverse = TRUE)
```

Arguments

grdr_vals,	a list of integers between 0 and 255
reverse,	a logical (TRUE/FALSE) indicating whether or not the string of bits should be reversed. Default: TRUE

Details

In the case of MODIS, the bits are read from RIGHT to LEFT, which is unnatural, so by default the function uses rev() to reverse the order of reading.

This function is just the mapply (multiply apply) version of byteint2bit

Note: the reversal means that, when interpreting the two 2-bit strings in the MODIS image, the interpretation of 01 and 10 is reversed from in the MODIS documentation.

Value

byte_in_binary_table, a table, each cell has the 8 bits (0/1 values) in the correct order

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

extract_bit

byteint2bit

Examples

```
grdr_vals = c(0,1,2,253,254,255)
byteint2bit_list(grdr_vals, reverse=TRUE)
```

---

check_for_matching_geolocation_files

Checks that every MODIS cloud project HDF has a matching MOD03 file

---

Description

Each MOD35_L2 cloud mask product file requires a corresponding MOD03 geolocation file to be successfully processed with the MRTSwath tool.

Usage

```
check_for_matching_geolocation_files(moddir = getwd(),
  modtxt = "MOD35_L2", geoloctxt = "MOD03",
  return_geoloc = FALSE, return_product = FALSE)
```

Arguments

moddir	the string describing the directory containing the MOD35_L2 and MOD03 files; both must be in the same directory. Default: getwd(), which gives the present working directory.
modtxt	the text string indicating which HDF files are the MODIS cloud product (or hypothetically, other product). Default: MOD35_L2 (MODIS cloud mask product)
geoloctxt	the text string indicating which HDF files are the MODIS geolocation files (or hypothetically, another set of files). Default: MOD03
return_geoloc	if TRUE, return the list of unmatched geolocation files (e.g. MOD03 or MYD03)
return_product	if TRUE, return the list of unmatched product files (e.g. MOD35_L2 or MYD35_L2 or MOD06_L2 or MYD06_L2)

Details

MRTSwath is the MRT (MODIS Reprojection Tool) for the MODIS level 1 and level 2 products (cloud mask is level 2, I think).

E.g. this cloud mask file:

MOD35_L2.A2012001.0420.005.2012001131638.hdf

...goes with this corresponding geolocation file:

MOD03.A2012001.0420.005.2012001104706.hdf

...which is a large file (~30 MB) containing detailed information on the position, tilt, etc. of the MODIS satellite.

For whatever reason, even a search done at the same time at http://reverb.echo.nasa.gov/reverb/#utf8= will not necessarily return the same number of MOD35_L2 and MOD03 files. MRTSwath tool needs one of each, however.

Value

data.frame of matching files; or a list of non-matching files, if `return_geoloc` or `return_product` are TRUE.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

`extract_time_from_MODISfn`

`modfns_to_ftp_download_cmd`

Examples

```
# Check your working directory
moddir = getwd()

# Here are some example MODIS files in modiscloud/extdata/
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
moddir = system.file("extdata/2002raw/", package="modiscdata")

# You need to have some e.g. MOD files in it (from the MODIS-TERRA platform)
# (*won't* work with the default files stored in modiscloud/extdata/2002raw/)
list.files(path=moddir, pattern="MOD")
```

```
# This directory actually has MYD files (from the MODIS-AQUA platform)
# (*will* work with the default files stored in modiscloud/extdata/2002raw/)
list.files(path=moddir, pattern="MYD")

# Check for matches (for MODIS-TERRA platform)
# (*won't* work with the default files stored in modiscloud/extdata/2002raw/)
check_for_matching_geolocation_files(moddir=moddir, modtxt="MOD35_L2", geoloctxt="MOD03", return_geoloc=FALSE,

# Check for matches (for MODIS-AQUA platform)
# (*will* work with the default files stored in modiscloud/extdata/2002raw/)
check_for_matching_geolocation_files(moddir=moddir, modtxt="MYD35_L2", geoloctxt="MYD03", return_geoloc=FALSE,

## End(Not run)
```

`dates_from_fileslist` *Convert each MODIS filename to year, month, day, hourmin*

Description

For each MODIS filename in a list, return a table of date/time information.

Usage

```
dates_from_fileslist(fns)
```

Arguments

`fns`	filenames

Value

`dates_table` a table of dates

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

`yearday_to_date`

`extract_time_from_MODISfn`

Examples

```
fns = c("MOD03.A2008001.0400.005.2010216170200.hdf", "MOD03.A2007002.0300.005.2010216170200.hdf")
dates_from_fileslist(fns)
```

`extract_bit` *Get the value of a particular bit in a byte*

Description

In many MODIS products, the information for each pixel is stored in a byte, which represents numbers between 0 and 255. A byte is made up of 8 bits, and is a base 2 representation of the number. In the MODIS cloud product, each bit encodes information; see documentation of the MODIS cloud product in question.

Usage

```
extract_bit(intval, bitnum)
```

Arguments

`intval`	An integer between 0 and 255
`bitnum`	The bit number to return, between 1 and 8

Details

This function takes a byte and extracts the bits. The bits are in whatever order they are in the byte. If reading in the reverse direction is needed, the user should use `rev`.

Value

`bitval` The value of the bit (0 or 1).

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

`byteint2bit`

`get_bitgrid_2bits`

Examples

```
intval = 123
extract_bit(intval, bitnum=1)
extract_bit(intval, bitnum=2)
extract_bit(intval, bitnum=8)
```

`extract_fn_from_path` *Get the filename from a path*

Description

The filename is split on slashes, and the last item is taken; this should be just the filename.

Usage

```
extract_fn_from_path(fn_with_path)
```

Arguments

`fn_with_path`	The filename, with partial or full path

Value

`fn` The extracted filename

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

`strsplit`

Examples

```
fn_with_path = "/Library/Frameworks/R.framework/Versions/2.15/Resources/library/modiscloud/extdata/2002raw/MYD3
extract_fn_from_path(fn_with_path)
```

`extract_time_from_MODISfn` *Extract the year, day, and hour from a MODIS filename*

Description

The filenames of MODIS images contain the following date information: MOD35_L2.Ayyyyddd.hhhh.etc.

Usage

```
extract_time_from_MODISfn(fn)
```

Arguments

`fn`	The MODIS filename of interest

Details

MODLAND Level 2 products
ESDT.AYYYYDDD.HHMM.CCC.YYYYDDDHHMMSS.hdf

ESDT = Earth Science Data Type name (e.g., MOD14)
YYYYDDD = MODIS acquisition year and Julian day
HHMM = MODIS acquisition UTC time
CCC = Collection number
YYYYDDDHHMMSS = Processing Year, Julian day and UTC Time
hdf = Suffix denoting HDF file

DDD is the day of the year, from 001 to 365 (or 366 for leap years)

Value

`newdate` A list with three items: $month, $day, $year

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

NASA (2001). "MODLAND Product Filename Convention." <URL: http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename>.

See Also

`dates_from_fileslist`

http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename

Examples

```
yearday_to_date(year=2012, day=364)
# $month
# [1] 12
#
# $day
# [1] 29
#
# $year
# [1] 2008

fn = "MOD03.A2008001.0400.005.2010216170200.hdf"
extract_time_from_MODISfn(fn)
```

`fermat.test` *Test an integer for primality with Fermat's little theorem*

Description

This is an example function for making R packages and R documentation with `roxygen2` and `roxygenize`.

Usage

```
fermat.test(n)
```

Arguments

`n`	the integer to test for primality

Details

Fermat's little theorem states that if n is a prime number and a is any positive integer less than n, then a raised to the nth power is congruent to $a \ modulo \ n$.

Value

Whether the integer passes the Fermat test for a randomized $0 < a < n$ no workyATcallGraph-Primitives

Note

`fermat.test` doesn't work for integers above approximately fifteen because modulus loses precision.

Author(s)

Peter Danenberg <pcd@roxygen.org>

References

http://en.wikipedia.org/wiki/Fermat's_little_theorem

---

foo *Test function*

Description

This is an example function for making R packages and R documentation with roxygen2 and roxygenize.

Usage

```
foo(d)
```

Arguments

d	hi

Details

This function just calculates the mean.

Value

meand

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

refline1 refline2

See Also

fermat.test

Examples

```
foo(c(1,2,3,4)) # 2.5
```

foo2 *Test function*

Description

This is an example function for making R packages and R documentation with roxygen2 and roxygenize.

Usage

```
foo2(d)
```

Arguments

d	description_of_input_param

Details

This function just calculates the mean.

Summary_paragraph

Details_paragraph

Value

description_of_what_is_returned

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

refline1 refline2

See Also

fermat.test

Examples

```
foo(c(1,2,3,4)) # 2.5
```

`get_bitgrid` *Extract the value of a particular bit at each pixel*

Description

Given an input grid (as a SpatialGridDataFrame object), extract a single desired bit value from all of the pixels

Usage

```
get_bitgrid(grd, bitnum)
```

Arguments

`grd`	A SpatialGridDataFrame, derived from e.g. readGDAL
`bitnum`	The bit you would like to extract

Details

Note: this means that, when interpreting the two 2-bit strings in the MODIS image, you would have to extract each bit separately.

Value

`grdr_vals_bits`, a list of 0/1 values for the bit in question (numeric)

Author(s)

Nicholas J. Matzke `<matzke@berkeley.edu>`

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

`extract_bit`

`get_bitgrid_2bits`

Examples

```
#########################################################
# Load a TIF
#########################################################
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
tifsdir = system.file("extdata/2002tif/", package="modiscdata")
tiffns = list.files(tifsdir, pattern=".tif", full.names=TRUE)
tiffns

library(rgdal) # for readGDAL

# numpixels in subset
xdim = 538
ydim = 538


# Read the grid and the grid metadata
coarsen_amount = 1
xdim_new = xdim / floor(coarsen_amount)
ydim_new = ydim / floor(coarsen_amount)

fn = tiffns[1]
grd = readGDAL(fn, output.dim=c(ydim_new, xdim_new))

grdproj = CRS(proj4string(grd))
grdproj
grdbbox = attr(grd, "bbox")
grdbbox

#########################################################
# Extract a particular bit for all the pixels in the grid
#########################################################
bitnum = 2
grdr_vals_bits = get_bitgrid(grd, bitnum)
length(grdr_vals_bits)
grdr_vals_bits[1:50]

## End(Not run)
```

`get_bitgrid_2bits` *extract the value of 2 particular bits at each pixel*

Description

Some MODIS cloud products have 2-bit codes, e.g. the MOD35_L2 product has a 2-bit code specifying the 4 possible outputs of the cloud-classification algorithm: confident cloudy, probable cloudy, probable clear, confident clear.

Usage

```
get_bitgrid_2bits(grd, bitnums)
```

Arguments

grd	A SpatialGridDataFrame, derived from e.g. readGDAL
bitnums	The bit you would like to extract

Details

Note: this means that, when interpreting the two 2-bit strings in the MODIS image, you have to reverse the bit order. This is done by default here.

Value

grdr_vals_bitstrings, a list of strings for the bits in question (string, e.g. "11" or "01" or "00")

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

extract_bit

get_bitgrid

Examples

```
#######################################################
# Load a TIF
#######################################################
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
```

```
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
tifsdir = system.file("extdata/2002tif/", package="modiscdata")
tiffns = list.files(tifsdir, pattern=".tif", full.names=TRUE)
tiffns

library(rgdal) # for readGDAL

# numpixels in subset
xdim = 538
ydim = 538


# Read the grid and the grid metadata
coarsen_amount = 1
xdim_new = xdim / floor(coarsen_amount)
ydim_new = ydim / floor(coarsen_amount)

fn = tiffns[1]
grd = readGDAL(fn, output.dim=c(ydim_new, xdim_new))

grdproj = CRS(proj4string(grd))
grdproj
grdbbox = attr(grd, "bbox")
grdbbox

########################################################
# Extract a particular pair of bits for all the pixels in the grid
########################################################
bitnum = 2
grdr_vals_bitstrings = get_bitgrid_2bits(grd, bitnum)
length(grdr_vals_bitstrings)
grdr_vals_bitstrings[1:50]

## End(Not run)
```

`get_dates_from_POSIXct` *Get the time information from POSIXct times*

Description

This function, for each input date, gets the year, month, day, and hourmin from a POSIXct date (which is the # of seconds from "1970-01-01 00:00.00 UTC"). Returns a formatted date in data.frame format

Usage

```
get_dates_from_POSIXct(POSIX_ct_dates)
```

Arguments

POSIX_ct_dates	list of POSIXct dates

Details

The function contains a check for singe dates. This is a version of get_dates_from_POSIXct for use with a list of input POSIXct dates, rather than a single date.

Value

dtf data.frame containing year, month, day, julian day, and hourmin / sec

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

get_date_from_POSIXct

make_POSIXct_date

Examples

```
POSIX_ct_dates = c(10000000, 20000000)
get_dates_from_POSIXct(POSIX_ct_dates)
```

`get_date_from_POSIXct` *Get the time information from a POSIXct time*

Description

This function gets the year, month, day, and hourmin, from a POSIXct date (which is the # of seconds from "1970-01-01 00:00.00 UTC"). Returns a formatted date in data.frame format

Usage

```
get_date_from_POSIXct(POSIX_ct_date)
```

Arguments

`POSIX_ct_date`	(# of seconds from "1970-01-01 00:00.00 UTC")

Details

The function contains a check for multiple dates.

Value

tdf a time-data.frame containing year, month, day, julian day, and hourmin / sec

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

`get_dates_from_POSIXct`

`make_POSIXct_date`

`as.POSIXct`

`as.POSIXlt`

Examples

```
# 10 million seconds from 1/1/1970
POSIX_ct_date = 10000000
get_date_from_POSIXct(POSIX_ct_date)
```

`is.pseudoprime` *Check an integer for pseudo-primality to an arbitrary precision*

Description

A number is pseudo-prime if it is probably prime, the basis of which is the probabilistic Fermat test; if it passes two such tests, the chances are better than 3 out of 4 that n is prime.

Usage

```
is.pseudoprime(n, times)
```

Arguments

`n`	the integer to test for pseudoprimality.
`times`	the number of Fermat tests to perform

Details

This is an example function for making R packages and R documentation with `roxygen2` and `roxygenize`.

Value

Whether the number is pseudoprime

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Abelson, Hal; Jerry Sussman, and Julie Sussman. Structure and Interpretation of Computer Programs. Cambridge: MIT Press, 1984.

See Also

`fermat.test`

Examples

```
is.pseudoprime(13, 4)# TRUE most of the time
```

`is.pseudoprime2` *Check an integer for pseudo-primality to an arbitrary precision, second version*

Description

A number is pseudo-prime if it is probably prime, the basis of which is the probabilistic Fermat test; if it passes two such tests, the chances are better than 3 out of 4 that n is prime.

Usage

```
is.pseudoprime2(n, times)
```

Arguments

`n`	the integer to test for pseudoprimality.
`times`	the number of Fermat tests to perform

Details

This is an example function for making R packages and R documentation with `roxygen2` and `roxygenize`.

Value

Whether the number is pseudoprime

Author(s)

Peter Danenberg <pcd@roxygen.org>

References

Abelson, Hal; Jerry Sussman, and Julie Sussman. Structure and Interpretation of Computer Programs. Cambridge: MIT Press, 1984.

See Also

`fermat.test`

Examples

```
is.pseudoprime2(13, 4)# TRUE most of the time
```

make_cloudcount_product *Take a cloudcount raster and a number-of-observations raster and make a fraction cloud product*

Description

Takes a cloudcount raster, and a number-of-observations raster, and makes a fraction cloud product.

Usage

```
make_cloudcount_product(master_bitgridvalsCC,
  master_bitgridvals0, num_its, grd, xdim_new, ydim_new,
  countzeros = FALSE)
```

Arguments

master_bitgridvalsCC	Cloud count raster
master_bitgridvals0	Raster indicating the count of valid observations
num_its	The total number of images or files, i.e. the number of attempts to see clouds
grd	An input grid with the appropriate projection and extent to match the rasters
xdim_new	The x-dimension (in # of pixels; i.e. # of columns) of the input/output grids
ydim_new	The y-dimension (in # of pixels; i.e. # of rows) of the input/output grids
countzeros	Should zeros be counted? TRUE or FALSE

Value

grd_final, a grid with fraction cloudy

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

sum_bitgrid

Examples

```
# For examples, see the function numslist_to_grd()
```

`make_POSIXct_date` *Take year, month, day, hourmin, convert to POSIXct*

Description

Make a standard-formatted date.

Usage

```
make_POSIXct_date(year, month, day, hourmin,
  timezone_txt = "UTC")
```

Arguments

`year`	calendar year
`month`	number of the month (1-12)
`day`	number of the day (1-31)
`hourmin`	a string (derived from MODIS filename) with HHSS (hours, seconds)
`timezone_txt`	the timezone; default 'UTC', your results may vary for the other timezones R knows about

Value

`POSIX_ct_date`, a POSIXct-formatted date

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

`get_date_from_POSIXct`

`get_dates_from_POSIXct`

`as.POSIXct`

Examples

```
# Example use of make_POSIXct_date
year=2008
month=3
day=3
hourmin="0404"
make_POSIXct_date(year, month, day, hourmin, timezone_txt="UTC")
```

`make_weeks_list` *Make a list of numbered weeks*

Description

Make a list of weeks for a given number of days.

Usage

```
make_weeks_list(numweeks = 53)
```

Arguments

`numweeks`	Number of weeks you would like to get the daily week categories for. Default is 53 weeks, i.e. 53*7 = 371 days.

Details

E.g., for 15 days,

`make_weeks_list(numweeks=17)`

...would return:

week1
week1
week1
week1
week1
week1
week1
week2
week2
week2
week2
week2
week2
week2
week2
week3
week3
week3

This is useful for categorizing e.g. daily satellite data by week.

Value

weeks_list, a list of the week category that each day falls into

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

NASA (2001). "MODLAND Product Filename Convention." <URL: http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename>.

Examples

```
# Make the weeks_list
weeks_list = make_weeks_list(numweeks=53)
weeks_list

# Which week category you are in
day=1
weeks_list[day]
day=8
weeks_list[day]
day=100
weeks_list[day]
day=366
weeks_list[day]
```

---

modfns_to_ftp_download_cmd

Make download commands for MODIS files

Description

Take some MODIS product filenames and produce the appropriate FTP "get" command.

Usage

```
modfns_to_ftp_download_cmd(mod03_fns_dropped,
  ftp_prefix = "get allData/5/", output_to = "screen")
```

Arguments

mod03_fns_dropped	A list of MODIS product files (e.g. the files returned as mismatches by check_for_matching_geolocati
ftp_prefix	The prefix that the user would like to pre-pend to the FTP directory

output_to — If "screen", the FTP commands are printed to screen (and returned as a list). If "data.frame", the FTP commands are only returned as a list. Any other string is interpreted as a filename, and the FTP commands will be saved to that as a textfile (and returned as a list).

Details

E.g., an input of:

```
MYD03.A2007019.1935.005.2009277181756.hdf
```

...is converted to:

```
get   allData/5/MYD03/2007/019/MYD03.A2007019.1935.005.2009277181756.hdf   MYD03.A2007019.1935.005.20
```

The user must specify `ftp_prefix`, e.g. for the products I have worked with, it is as follows:

```
MOD03: get allData/5/
MOD35_L2: get   allData/5/
MYD03: get allData/5/
MYD35_L2: get allData/5/
MYD06_L2: get   allData/51/
MYD06_L2: get allData/51/
```

Value

`ftp_cmds`, A list of the FTP commands.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

check_for_matching_geolocation_files

---

numslist_to_grd *Convert a list of numbers to a grid*

Description

Take a cloudcount raster and a number-of-observations raster and make a fraction cloud raster.

Usage

```
numslist_to_grd(numslist, grd, ydim_new, xdim_new)
```

Arguments

numslist	The list of numbers, each corresponding to a pixel
grd	A grid object, from which the projection and extend of the new grid will be taken
ydim_new	Number of rows (number of pixels in height)
xdim_new	Number of columns (number of pixels in width)

Value

grd_final, as SpatialPixelsDataFrame object

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

make_cloudcount_product

Examples

```
#########################################################
# Load some bit TIFs (TIFs with just the cloud indicators extracted)
# and add up the number of cloudy days, out of the total
# number of observation attempts
#########################################################
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
tifsdir = system.file("extdata/2002_bit/", package="modiscdata")
tiffns = list.files(tifsdir, pattern=".tif", full.names=TRUE)
tiffns

library(rgdal) # for readGDAL

# numpixels in subset
xdim = 538
```

```
ydim = 538

# Read the grid and the grid metadata
coarsen_amount = 1
xdim_new = xdim / floor(coarsen_amount)
ydim_new = ydim / floor(coarsen_amount)


sum_nums = NULL
for (j in 1:length(tiffns))
{
fn = tiffns[j]

grd = readGDAL(fn, output.dim=c(ydim_new, xdim_new))

grdr = raster(grd)
pointscount_on_SGDF_points = coordinates(grd)
grdr_vals = extract(grdr, pointscount_on_SGDF_points)

# Convert to 1/0 cloudy/not
data_grdr = grdr_vals
data_grdr[grdr_vals > 0] = 1

grdr_cloudy = grdr_vals
grdr_cloudy[grdr_vals < 4] = 0
grdr_cloudy[grdr_vals == 4] = 1

# Note: Don't run the double-commented lines unless you want to collapse different bit values.
# grdr_clear = grdr_vals
# grdr_clear[grdr_vals == 4] = 0
# grdr_clear[grdr_vals == 3] = 1
# grdr_clear[grdr_vals == 2] = 1
# grdr_clear[grdr_vals == 1] = 1
# grdr_clear[grdr_vals == 0] = 0
#

if (j == 1)
{
sum_cloudy = grdr_cloudy
#sum_not_cloudy = grdr_clear
sum_data = data_grdr
} else {
sum_cloudy = sum_cloudy + grdr_cloudy
sum_data = sum_data + data_grdr
}

}


## Calculate percentage cloudy
sum_nums = sum_cloudy / sum_data

grd_final = numslist_to_grd(numslist=sum_nums, grd=grd, ydim_new=ydim_new, xdim_new=xdim_new)
```

```
# Display the image (this is just the sum of a few images)
image(grd_final)


## End(Not run)
```

`run_swath2grid` *Run MRTSwath swath2grid tool*

Description

MRTSwath is the "MODIS Reprojection Tool for swath products". See: https://lpdaac.usgs.gov/tools/modis_reprojection_tool_swath).

Usage

```
run_swath2grid(mrtpath = "swath2grid",
  prmfn = "tmpMRTparams.prm", tifsdir, modfn, geoloc_fn,
  ul_lon, ul_lat, lr_lon, lr_lat)
```

Arguments

`mrtpath`	This is the path to the MRTSwath executable `swath2grid`. If your `~/.Rprofile` file has the location of `swath2grid` in the PATH, then you can just use `mrtpath="swath2grid"`. Otherwise, the user must provide the full path to swath2grid.
`prmfn`	The name of the parameter/control file which will be the input to MRTSwath's `swath2grid` function.
`tifsdir`	The directory to save the output TIF files in
`modfn`	The filename of the MODIS data
`geoloc_fn`	The filename of the corresponding geolocation file (annoyingly, this is a much larger file than the data file!)
`ul_lon`	Upper left (ul) longitude (x-coordinate) for subsetting
`ul_lat`	Upper left (ul) latitude (y-coordinate) for subsetting
`lr_lon`	Lower right (lr) longitude (x-coordinate) for subsetting
`lr_lat`	Lower right (lr) latitude (y-coordinate) for subsetting

Details

If you want this function to use MRTSwath tool successfully, you should add the directory with the MRTSwath executable to the default R PATH by editing `~/.Rprofile`.

Value

`cmdstr` The string giving the system command that ran `swath2grid`

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

NASA (2001). "MODLAND Product Filename Convention." <URL: http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename>.

See Also

write_MRTSwath_param_file

http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename

Examples

```
#########################################################
# Run MRTSwath tool "swath2grid"
#########################################################

# Source MODIS files (both data and geolocation)
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
moddir = system.file("extdata/2002raw/", package="modiscdata")

# Get the matching data/geolocation file pairs
fns_df = check_for_matching_geolocation_files(moddir, modtxt="MYD35_L2", geoloctxt="MYD03")
fns_df

# Resulting TIF files go in this directory
tifsdir = getwd()


# Box to subset
ul_lat = 13
ul_lon = -87
lr_lat = 8
lr_lon = -82

for (i in 1:nrow(fns_df))
{

prmfn = write_MRTSwath_param_file(prmfn="tmpMRTparams.prm", tifsdir=tifsdir, modfn=fns_df$mod35_L2_fns[i], geol
print(scan(file=prmfn, what="character", sep="\n"))
```

```
run_swath2grid(mrtpath="swath2grid", prmfn="tmpMRTparams.prm", tifsdir=tifsdir, modfn=fns_df$mod35_L2_fns[i], g

}

list.files(tifsdir, pattern=".tif", full.names=TRUE)

## End(Not run)
```

`slashslash` *Remove double slash (slash a slash)*

Description

Shortcut for: `gsub(pattern="//", replacement="/", x=tmpstr)`

Usage

```
slashslash(tmpstr)
```

Arguments

`tmpstr`	a path that you want to remove double slashes from

Details

This function is useful for removing double slashes that can appear in full pathnames due to inconsistencies in trailing slashes in working directories etc.

Value

outstr a string of the fixed path

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

gsub

Examples

```
tmpstr = "/Library/Frameworks/R.framework/Versions/2.15/Resources/library/modiscloud/extdata/2002raw//MYD03.A20

outstr = slashslash(tmpstr)
outstr
```

sum_bitgrid *Take a series of byte tifs, extract the values of a bit, and add them up*

Description

This function is useful if you have a series of cloud product byte tifs, and need to extract a bit from each, and then add up (e.g. to get the total number of cloudy observations).

Usage

```
sum_bitgrid(fns, numits = length(fns), bitnums, ydim_new,
  xdim_new, test_for_1 = 1)
```

Arguments

fns	A list of filenames which can be auto-read by readGDAL (e.g. geoTIFFs)
numits	Number of iterations to run for (default = length(fns))
bitnums	The bit(s) you would like to extract. If length(bitnums) = 1, do default behavior. If the length is 2, then run the alternative function.
ydim_new	The dimensions of the extracted image. This is REQUIRED, since each day of a MODIS image may have different pixel dimensions over the same location. readGDAL will sample each image (nearest neighbor pixel) to the correct dimensions.
xdim_new	The dimensions of the extracted image. This is REQUIRED, since each day of a MODIS image may have different pixel dimensions over the same location. readGDAL will sample each image (nearest neighbor pixel) to the correct dimensions.
test_for_1	What your extracted bit(s) should match. If 1 (default), just count up 1s. If a string, e.g. "!=11", use that in an if-then statement to get the 1s to count

Value

grdr_vals_bits, a matrix of 0/1 values for the bit in question

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

extract_bit

numslist_to_grd

Examples

```
# See numslist_to_grd example for an idea of how this works.
```

---

unlist_df *Unlist the columns in a data.frame*

Description

Sometimes, matrices or data.frames will malfunction due to their having lists as columns and other weirdness. This is a shortcut for `data.frame(lapply(df, function(x) unlist(x)))`.

Usage

```
unlist_df(df)
```

Arguments

df	matrix or other object transformable to data.frame

Value

data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

unlist_df2

Examples

```
df = adf(matrix(c(1,2,3,4,5,6), nrow=3, ncol=2))
df2 = unlist_df(df)
df2
```

unlist_df2 *Unlist the columns in a data.frame, with more checks*

Description

Sometimes, matrices or data.frames will malfunction due to their having lists as columns and other weirdness. This runs unlist_df and additional checks.

Usage

```
unlist_df2(df)
```

Arguments

df	matrix or other object transformable to data.frame

Value

data.frame

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

unlist_df

Examples

```
df = adf(matrix(c(1,2,3,4,5,6), nrow=3, ncol=2))
df2 = unlist_df2(df)
df2
```

write_MRTSwath_param_file *Write a parameter control file for MRTSwath*

Description

MRTSwath is the "MODIS Reprojection Tool for swath products". See: https://lpdaac.usgs.gov/tools/modis_reprojection_tool_swath).

Usage

```
write_MRTSwath_param_file(prmfn = "tmpMRTparams.prm",
  tifsdir, modfn, geoloc_fn, ul_lon, ul_lat, lr_lon,
  lr_lat)
```

Arguments

prmfn	The name of the parameter/control file which will be the input to MRTSwath's swath2grid function.
tifsdir	The directory to save the output TIF files in
modfn	The filename of the MODIS data
geoloc_fn	The filename of the corresponding geolocation file (annoyingly, this is a much larger file than the data file!)
ul_lon	Upper left (ul) longitude (x-coordinate) for subsetting
ul_lat	Upper left (ul) latitude (y-coordinate) for subsetting
lr_lon	Lower right (lr) longitude (x-coordinate) for subsetting
lr_lat	Lower right (lr) latitude (y-coordinate) for subsetting

Details

If you want this function to use MRTSwath tool successfully, you should add the directory with the MRTSwath executable to the default R PATH by editing ~/.Rprofile.

This function hard-codes these options into the parameter file:
* all the bands are extracted
* the output file is a GeoTIFF
* the output projection is Geographic (plain unprojected Latitude/Longitude)
* the resampling is Nearest Neighbor (NN), which of course is the only one which makes sense when the pixels encode bytes that encode bits that encode discrete classification results, 0/1 error flags, etc.

MRTswath can do many other projections and output formats; users can modify this function to run those options.

Value

prmfn The name of the temporary parameter file

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

NASA (2001). "MODLAND Product Filename Convention." <URL: http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename>.

See Also

run_swath2grid

http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename

Examples

```
# Source MODIS files (both data and geolocation)
# Code excluded from CRAN check because it depends on modiscdata
## Not run:
library(devtools)
# The modiscdata (MODIS c=cloud data=data) package is too big for CRAN (60 MB); so it is available on github:
# https://github.com/nmatzke/modiscdata
# If we can't get install_github() to work, try install_url():
# install_github(repo="modiscdata", username="nnmatzke")
install_url(url="https://github.com/nmatzke/modiscdata/archive/master.zip")
library(modiscdata)
moddir = system.file("extdata/2002raw/", package="modiscdata")

# Get the matching data/geolocation file pairs
fns_df = check_for_matching_geolocation_files(moddir, modtxt="MYD35_L2", geoloctxt="MYD03")
fns_df

# Resulting TIF files go in this directory
tifsdir = getwd()


# Box to subset
ul_lat = 13
ul_lon = -87
lr_lat = 8
lr_lon = -82

for (i in 1:nrow(fns_df))
{

prmfn = write_MRTSwath_param_file(prmfn="tmpMRTparams.prm", tifsdir=tifsdir, modfn=fns_df$mod35_L2_fns[i], geol
print(scan(file=prmfn, what="character", sep="\n"))

}

## End(Not run)
```

---

yearday_to_date *Convert a year + a day number to a date*

Description

The filenames of MODIS images contain the following date information: MOD35_L2.Ayyyyddd.hhhh.etc.

Usage

```
yearday_to_date(year = 2012, day = 1)
```

Arguments

year	The year, read as numeric
day	The day of the year, read as numeric, from 1 to 366

Details

MODLAND Level 2 products ESDT.AYYYYDDD.HHMM.CCC.YYYYDDDHHMMSS.hdf

ESDT = Earth Science Data Type name (e.g., MOD14) YYYYDDD = MODIS acquisition year and Julian day HHMM = MODIS acquisition UTC time CCC = Collection number YYYYDDDHHMMSS = Processing Year, Julian day and UTC Time hdf = Suffix denoting HDF file

DDD is the day of the year, from 001 to 365 (or 366 for leap years)

This is mildly annoying to interpret as e.g. months for graphing cloudy days per month, so `yearday_to_date` converts a (numeric) year and day to the calendar date.

Value

`newdate` A list with three items: `$month`, `$day`, `$year`

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

NASA (2001). "MODLAND Product Filename Convention." <URL: http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename>.

See Also

`make_POSIXct_date`

`yearday_to_date`

`http://landweb.nascom.nasa.gov/cgi-bin/QA_WWW/newPage.cgi?fileName=hdf_filename`

Examples

```
yearday_to_date(year=2012, day=364)
# $month
# [1] 12
#
# $day
# [1] 29
#
# $year
# [1] 2008
```

`yearmonthday_to_julianday` *Get the julian day for a year/month/day date*

Description

This function uses the date package's `mdy.date` function to get the Julian day (count of the day in the year, from 1-365, or 1-366 for leap years) from the input year, month, and day.

Usage

```
yearmonthday_to_julianday(year, month, day)
```

Arguments

`year`	as numeric, 4 digits
`month`	as numeric, 1-2 digits
`day`	as numeric, 1-2 digits

Value

julian_day, the julian day between 1-365 or 1-366 (for leap years)

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

References

Ackerman S, Frey R, Strabala K, Liu Y, Gumley L, Baum B and Menzel P (2010). "Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35)." MODIS Cloud Mask Team, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison. <URL: http://modis-atmos.gsfc.nasa.gov/_docs/MOD35_ATBD_Collection6.pdf>.

GoldsmithMatzkeDawson2013

See Also

`yearday_to_date`

Examples

```
year=2011; month=06; day=15
yearmonthday_to_julianday(year, month, day)
```

Index