Package 'Rnightlights'

November 23, 2017

Version 0.1.4

Date 2017-11-21

Title Satellite Nightlight Data Extraction

Description The Rnightlights package extracts raster and zonal statistics from satellite nightlight rasters downloaded from the United States National Oceanic and Atmospheric Administration (<http://www.noaa.gov>) free data repositories. Both the DMSP-OLS annual and SNPP-VIIRS monthly nightlight raster data are supported. Satellite nightlight raster tiles are downloaded and cropped to the country boundaries using shapefiles from the GADM database of Global Administrative Areas (<http://gadm.org>). Zonal statistics are then calculated at the lowest administrative boundary for the selected country and cached locally for future retrieval. Finally, a simple data explorer/browser is included that allows one to visualize the cached data e.g. graphing, mapping and clustering regional data.

License GPL-3

Imports cleangeo, compiler, data.table, doParallel, dplyr, foreach, gdalUtils, lubridate, methods, raster, readr, reshape2, rgdal, rgeos, R.utils, rvest, rworldmap, settings, sp, stats, stringr, utils, xml2

Suggests dendextend, ggdendro, ggplot2, leaflet, plotly, RColorBrewer, reshape, shiny, shinydashboard

SystemRequirements gdal wget curl aria2

RoxygenNote 6.0.1

BugReports https://github.com/chrisvwn/Rnightlights/issues

URL https://github.com/chrisvwn/Rnightlights

NeedsCompilation no

Author Christopher Njuguna [aut, cre, cph], Henrik Bengtsson [ctb] (data setup, https://cran.r-project.org/package=R.cache), Black Guru [ctb] (gdal zonal, http://www.guru-gis.net/efficient-zonal-statistics-using-r-and-gdal/), Jeff Chen [ctb] (masq function,

https://commercedataservice.github.io/tutorial_viirs_part1/),
Star Ying [ctb] (masq function,
https://commercedataservice.github.io/tutorial_viirs_part1/),
Chris Elvidge [ctb] (information + masq function)

Maintainer Christopher Njuguna <chris.njuguna@gmail.com>

Repository CRAN

Date/Publication 2017-11-23 13:14:56 UTC

R topics documented:

addREADME *Add README file to the root data path*

Description

Add README file to the root data path

Usage

```
addREADME(to = getNlDataPath())
```

Arguments

to	The folder to add the README file to

Value

None

Examples

```
## Not run: addREADME()
```

allValid *Check if a vector/list of values given is valid as per the given validation function*

Description

Check if a vector/list of values given is valid as per the given validation function. The function will also print a warning showing the values that are invalid. One can stop the warning being printed by wrapping the function in the `suppressWarnings` function.

Usage

```
allValid(testData, testFun, ...)
```

Arguments

testData	The list/vector of values to validate
testFun	The validation function to test each value of testData against
...	Other parameters to pass on to the testFun

Value

TRUE/FALSE

Examples

```
## Not run: allValid(c("KE", "UGA", "RWA", "TZ"), validCtryCode)

## Not run: allValid(c("2012", "2015"), validNlPeriod, "OLS")
```

---

createCtryNlDataDF *Initiates the country nightlight dataframe with the country data read from the polygon*

Description

Initiates the country nightlight dataframe with the country data read from the polygon. This includes admin levels, level names and area

Usage

```
createCtryNlDataDF(ctryCode)
```

Arguments

ctryCode	the ISO3 code of the country

Value

dataframe with the country admin level data

Examples

```
## Not run: initCtryNlData <- createCtryNlDataDF("KEN")
 #returns a data frame
```

`createNlDataDirs` *Create required data subdirectories in the root data path*

Description

Create required data subdirectories in the root data path

Usage

```
createNlDataDirs()
```

Value

None

Examples

```
## Not run: createNlDataDirs()
```

`createNlTilesSpPolysDF` *Creates a tile Spatial Polygons DataFrame from the "nlTiles" dataframe*

Description

Creates a Spatial Polygons DataFrame from the "nlTiles" dataframe of VIIRS tiles

Usage

```
createNlTilesSpPolysDF()
```

Value

TRUE/FALSE

Examples

```
## Not run: tilesSpPolysDFs <- createNlTilesSpPolysDF()
```

ctryCodeToName *Convert a country ISO3 code to the full name*

Description

Convert a country ISO3 code to the full name. Exposes the rworldmap function isoToName(ctryCode). #rworldmap::isoToName can resolve 2-letter ctryCodes but we only want 3-letter ISO3 codes. With no parameters returns a list of ctryCodes and their corresponding names as given by rworldMap::getMap@data

Usage

```
ctryCodeToName(ctryCode)
```

Arguments

ctryCode	The country Code to search for

Value

Character The full country name if the ctryCode is found. If ctryCode is not supplied then return a list of all country codes and their corresponding names

Examples

```
ctryCodeToName("KEN") #Kenya

ctryCodeToName("ARE") #United Arab Emirates

ctryCodeToName("USA") #United States of America

ctryCodeToName("JAM") #Jamaica
```

ctryNameToCode *Convert a country name to its ISO3 code*

Description

Convert a country name to its ISO3 code. Exposes the rworldmap function rwmGetISO3(ctryName). See the examples. With no parameters returns a list of ctryNames and their corresponding codes as given by rworldMap

Usage

```
ctryNameToCode(ctryName)
```

Arguments

ctryName	Chracter string common name of a country

Value

Character ISO3 ctryCode if found else NA

Examples

```
ctryNameToCode("kenya")
  #returns "KEN"

ctryNameToCode("ken")
  #returns "KEN"

ctryNameToCode("jamaica") #returns JAM
```

---

ctryShpLyrName2Num *Get the integer number of the layer. topmost country layer=0*

Description

Get the integer number of the layer. Is the last character in the name and is a digit. E.g. for the 3rd layer in Kenya shapefile polygon named "KEN_adm3" the layer number is "3"

Usage

```
ctryShpLyrName2Num(layerName)
```

Arguments

layerName	- the name of the polygon layer

Value

Integer layer number

Examples

```
## Not run: ctryShpLyrName2Num("KEN_adm1")
  #returns 1
```

dnldCtryPoly — *Download a country's polygon shapefile from* http://gadm.org

Description

Download a country's polygon shapefile from http://gadm.org

Usage

```
dnldCtryPoly(ctryCode)
```

Arguments

ctryCode	The ISO3 ctryCode of the country polygon to download

Value

TRUE/FALSE Success/Failure of the download

Examples

```
## Not run: downloadCtryPoly("KEN")
```

downloadNlTiles — *Download the listed tiles for a given nlType in a given nlPeriod*

Description

Download the listed tiles for a given nlType in a given nlPeriod

Usage

```
downloadNlTiles(nlType, nlPeriod, tileList)
```

Arguments

nlType	character The nightlight type
nlPeriod	character The nlPeriod to process in the appropriate format
tileList	integer vector or character vector of digits containing valid tile numbers as obtained by tileName2Idx for VIIRS. Ignore for nlType=="OLS"

Value

TRUE/FALSE if the download was successful

Examples

```
#download VIIRS tiles for "KEN" which are tiles 2 and 5 for the specified
    #time periods
## Not run: downloadNlTiles("VIIRS", "201401", c(2, 5))

#same as above but getting the tileList automatically
## Not run:
downloadNlTiles("VIIRS", "201401",
    tileName2Idx(getCtryTileList(ctryCodes="KEN",
    nlType="VIIRS")))

## End(Not run)

#returns TRUE if the download was successful
```

`downloadNlTilesOLS` *Download OLS nightlight tile*

Description

Download OLS nightlight tile

Usage

```
downloadNlTilesOLS(nlYear, downloadMethod = pkgOptions("downloadMethod"))
```

Arguments

`nlYear`	the year in "YYYY" format e.g. "2012"
`downloadMethod`	The method to use for download.

Value

TRUE/FALSE Whether the download was successful

Examples

```
## Not run:
if(downloadNlTilesOLS("201405"))
  print("download successful")

## End(Not run)
```

downloadNlTilesVIIRS *Download VIIRS nightlight tile*

Description

Download VIIRS nightlight tile

Usage

```
downloadNlTilesVIIRS(nlYearMonth, tileNum,
  downloadMethod = pkgOptions("downloadMethod"))
```

Arguments

nlYearMonth	the year in "YYYYMM" format e.g. "201401"
tileNum	the index of the tile as given by getNlTiles("VIIRS")
downloadMethod	The method to use for download.

Value

TRUE/FALSE Whether the download was successful

Examples

```
## Not run:
if(downloadNlTilesVIIRS("201401", "1"))
  print("download successful")

## End(Not run)
```

existsCtryNlData *Check if VIIRS nightlight stats exist locally*

Description

Check if VIIRS nightlight data for the country exists in the country nightlight data file. First checks if the country nightlight data file exists.

Usage

```
existsCtryNlData(ctryCode, nlPeriod, stat, nlType)
```

Arguments

ctryCode	character the ISO3 code of the country
nlPeriod	character the nlPeriod
stat	character the stat to check for
nlType	character the nlType

Value

TRUE/FALSE

Examples

```
## Not run: existsCtryNlData("KEN", "201401", "sum", "VIIRS")
```

---

existsCtryNlDataFile *Check if a country's data file exists*

Description

Check if a country's data file exists

Usage

```
existsCtryNlDataFile(ctryCode)
```

Arguments

ctryCode	the ISO3 country code

Value

TRUE/FALSE

Examples

```
## Not run:
ctryCode <- "KEN"
if(existsCtryNlDataFile(ctryCode))
 message("Data file for ", ctryCode, " found")

## End(Not run)
```

existsPolyFnamePath *Check if the decompressed country polygon has been downloaded and stored in the polygon folder*

Description

Check if the decompressed country polygon has been downloaded and stored in the polygon folder

Usage

```
existsPolyFnamePath(ctryCode)
```

Arguments

ctryCode	The ctryCode to process

Value

TRUE/FALSE

Examples

```
## Not run: existsPolyFnamePath("KEN")
 #returns TRUE/FALSE
```

existsPolyFnameZip *Check if the compressed country polygon has been downloaded and stored in the polygon folder*

Description

Check if the compressed country polygon has been downloaded and stored in the polygon folder

Usage

```
existsPolyFnameZip(ctryCode)
```

Arguments

ctryCode	The ctryCode of interest

Value

TRUE/FALSE

Examples

```
## Not run: existsPolyFnameZip("KEN")
 #returns TRUE/FALSE
```

`exploreData` *Run a web application to explore the processed nightlight data cached locally*

Description

Run a web application to perform some exploratory data analysis. The application written in shiny either loads demo data or the data processed and cached locally.

Usage

```
exploreData()
```

Value

None

Examples

```
exploreData()
```

`fnAggRadGdal` *Calculate zonal statistics using GDAL. Faster than fnAggRadRast for large polygons.*

Description

Calculate zonal statistics. Alternative to fnAggRadRast using GDAL. Faster for large polygons. Modified from http://www.guru-gis.net/efficient-zonal-statistics-using-r-and-gdal/

Usage

```
fnAggRadGdal(ctryCode, ctryPoly, nlPeriod, fnStats = pkgOptions("stats"),
  nlType)
```

Arguments

ctryCode	character string the ISO3 country code to be processed
ctryPoly	Polygon the loaded country polygon layer
nlPeriod	character string the nlPeriod to be processed
fnStats	character vector The stats to calculate
nlType	the nlType of interest

Value

data.frame of polygon attributes and the calculated stats

Examples

```
## Not run: validNlMonthNum("01","VIIRS")
 #returns TRUE
```

---

fnAggRadRast *Calculate statistics on a nightlight raster that fall within a polygon*

Description

Calculate stats on the radiance of the pixels in a nightlight raster that fall within a polygon and its subpolygons using the raster package. Given a country polygon with subpolygons representing lower admin levels, it will crop and mask the raster to each subpolygon and calculate the total radiance for the polygon and return a vector of total radiances that matches the subpolygons

Usage

```
fnAggRadRast(ctryPoly, ctryRastCropped, stats, nlType)
```

Arguments

ctryPoly	The polygon of the admin level/region of interest. In general is a country polygon with sub-regions usually the lowest known admin level as given by the GADM polygons.
ctryRastCropped	The raster containing nightlight radiances to sum. Usually will have already be cropped to the country outline
stats	The statistics to calculate
nlType	Character vector The nlType to process

Value

Integer Sum of radiances of all pixels in a raster that fall within a polygon region

Examples

```
#read the Kenya polygon downloaded from GADM and load the lowest admin level (ward)
## Not run:
ctryPoly <- readOGR(getPolyFnamePath("KEN"), getCtryShpLowestLyrName("KEN"))

# the VIIRS nightlight raster cropped earlier to the country outline
ctryRastCropped <- getCtryRasterOutputFname("KEN","VIIRS","201401")

#calculate the sum of radiances for the wards in Kenya
sumAvgRadRast <- fnAggRadRast(ctryPoly, ctryRastCropped)

## End(Not run)
```

`getAllNlCtryCodes` *Check if a month number is valid for a given nightlight type*

Description

Check if a month number is valid for a given nightlight type. Note month num is only valid for "VIIRS" nightlight type

Usage

```
getAllNlCtryCodes(omit = "none")
```

Arguments

`omit`	The ctryCodes to exclude from processing. Else keywords missing=ctryCodes that are in rworldmap but not on gadm long=ctryCodes that take very long to process

Value

character vector of country codes

`getAllNlPeriods` *Generate a list of all possible nlPeriods for a given nlType*

Description

Generate a list of all possible nlPeriods for a given nlType

Usage

```
getAllNlPeriods(nlType)
```

Arguments

nlType	type of nightlight either "VIIRS" or "OLS"

Value

character vector list of nlPeriods

Examples

```
getAllNlPeriods("OLS")
 #returns a vector of all years from 1994 to 2013

getAllNlPeriods("VIIRS")
 #returns a vector of all yearMonths from 201401 to present
```

---

getCtryNlData *Returns nightlight statistics for the given ctryCode and nlType in the given nPeriods*

Description

Returns nightlight data for the given ctryCode and stats in the given nlPeriods and of the specified nlType. Note that getCtryNldata only processes one ctryCode at a time. `ignoreMissing` plays a significant role here. It can take 3 values:

- NULL (default) only return data if found for all nlPeriods and all stats provided otherwise return NULL.
- TRUE return any partial data that is found for the provided nlPeriods and stats. Ignore any missing data.
- FALSE return all data that is found and call `processNlData` to download and process any missing nlPeriods and stats.

Farther, if `nlPeriods` is missing, it is assigned values based on the value of ignoreMissing. If ignoreMissing is FALSE, nlPeriods is assigned all existing nlPeriods to date. This is the equivalent of retrieving all nightlight data for the given country and stats. If ignoreMissing is TRUE or NULL then the existing data is returned.

Usage

```
getCtryNlData(ctryCode, nlPeriods, nlType, stats = pkgOptions("stats"),
  ignoreMissing = NULL, source = "local")
```

Arguments

ctryCode	the ISO3 code of the country. Only 1 country can be processed at a time
nlPeriods	a vector of nlPeriods. Must be appropriate nlPeriods for the nlType.
nlType	the nightlight type i.e. "OLS" or "VIIRS" (default)
stats	a vector of stats. if not supplied defaults to all stats as listed in pkgOptions("stats")
ignoreMissing	controls how the function behaves if any data is not found in the data file • NULL (default) only return data if found for ALL nlPeriods and ALL stats provided otherwise return NULL • TRUE return any partial data that is found for the provided nlPeriods and stats. Ignore any missing data • FALSE return all data that is found and call processNlData to download and process any missing nlPeriods and stats
source	"local" or "remote" Whether to download and process the data locally or to download the pre-processed data from a remote source/repo

Value

dataframe of data for one country in one nlType in one or multiple nlPeriods

Examples

```
#missing stats implies all stats as given by pkgOptions("stats")

## Not run: getCtryNlData("KEN", nlType="VIIRS", ignoreMissing=NULL)
    #returns all existing data i.e. all nlPeriods and all stats for KEN

## Not run: getCtryNlData("KEN", ignoreMissing=TRUE)
    #same as ignoreMissing=NULL. Returns all existing data i.e. all nlPeriods
    #and all stats for KEN

## Not run: getCtryNlData(ctryCode="KEN", nlType="VIIRS", ignoreMissing=FALSE)
    #for any missing data between 201401 to present download and process the
    #data then return all data

## Not run: getCtryNlData("KEN", nlPeriod=c("existingNlPeriod", "missingNlPeriod"),
    stats=c("sum", "unknownStat"), ignoreMissing=NULL)
## End(Not run)
    #Returns NULL
    #(ignoreMissing=NULL returns all data if exists or if any is missing returns NULL)

## Not run: getCtryNlData("KEN", nlPeriods=c("existingNlPeriod", "missingNlPeriod"),
    stats=c("existingStat", "missingStat"), ignoreMissing=TRUE)
## End(Not run)
   #Returns existingStat for existingNlPeriods
   #(ignoreMissing=TRUE returns only existing data)

## Not run: getCtryNlData("KEN", nlYearPeriods=c("existingNlPeriod", "missingNlPeriod"),
    stats=c("sum", "unknownStat"), ignoreMissing=FALSE)
```

```
## End(Not run)
    #Runs processNlData for missingStat in "missingNlPeriod" and returns
    #"existingStat" and "missingStat" for both "existingNlPeriod" and
    #"missingNlPeriod"
    #(ignoreMissing=FALSE must return all data: forces processing of any missing)
```

`getCtryNlDataColName` *Construct the name of a nightlight data column given the nightlight type and nlPeriod*

Description

Construct the name of a nightlight data column given the nightlight type and nlPeriod Used in creating and retrieving data columns from the nightlight data file

Usage

```
getCtryNlDataColName(nlPeriod, stat, nlType)
```

Arguments

`nlPeriod`	character vector The nlPeriod to process
`stat`	character vector The stat to be stored in the column
`nlType`	character vector The type of nightlight i.e. "OLS" or VIIRS. Default=VIIRS

Value

character string

Examples

```
## Not run:
ctryCode <- "KEN"
dt <- read.csv(getCtryNlDataFnamePath(ctryCode))
dt <- dt[,getCtryNlDataColName("201612", "sum", nlType="VIIRS")]
  #returns the column "NL_201612_SUM" if it exists in the KEN data file

## End(Not run)
```

`getCtryNlDataFname` *Check if a month number is valid for a given nightlight type*

Description

Get the name of the data file. This function can be altered to name the file as required and consistently retrieve the name. Used in the function getCtryNlDataFnamePath to concat the directory path and this filename. Currently all nlTypes are stored in one file. Can be altered to separate VIIRS and OLS data files for example.

Usage

```
getCtryNlDataFname(ctryCode)
```

Arguments

`ctryCode`	The ctryCode of interest

Value

Character filename of the country data file

Examples

```
ctryCode <- "KEN"
## Not run: getCtryNlDataFname(ctryCode)
 #returns name of the ctry data file
```

`getCtryNlDataFnamePath` *Get the full path to the file containing the country data*

Description

Get the full path to the file containing the country data

Usage

```
getCtryNlDataFnamePath(ctryCode)
```

Arguments

`ctryCode`	character string The ctryCode of interest

Value

Character string the full path to the data file of a country

Examples

```
#get the full path to the file containing data for KEN
getCtryNlDataFnamePath("KEN")

## Not run:
ctryDF <- read.csv(getCtryNlDataFnamePath("KEN"))
 #returns DF with nightlight data for the country

## End(Not run)

#@export only due to exploreData() shiny app
```

`getCtryPolyAdmLevelNames` *Get the list of admin level names in a polygon shapefile*

Description

Get the list of admin level names in a polygon shapefile

Usage

```
getCtryPolyAdmLevelNames(ctryCode)
```

Arguments

`ctryCode`	the ctryCode of the country of interest

Value

character vector of admin level names

Examples

```
## Not run: getCtryPolyAdmLevelsNames("KEN")
```

`getCtryPolyUrl` *Get the GADM url from which to download country polygons*

Description

Get the url from which to download country polygons. Polygons are downloaded from http://www.gadm.org. This provides the url to the zipped ESRI Shapefile which when decompressed contains a directory with the different country admin level boundary files. A sample url returned for Afghanistan: http://biogeo.ucdavis.edu/data/gadm2.8/shp/AFG_adm_shp.zip

Usage

```
getCtryPolyUrl(ctryCode)
```

Arguments

`ctryCode`	character string The ctryCode of interest

Value

Character string url of the zipped ESRI shapefile for the ctryCode

Examples

```
## Not run: getCtryPolyUrl(ctryCode)
 #returns url for the zipped country ESRI shapefile
```

`getCtryRasterOutputFname` *Get the full path to the file where the cropped VIIRS country raster is stored.*

Description

Get the full path to the file where the cropped VIIRS country raster is stored. This file is created when processing the country before extracting the data. It can be used to re-process a country much faster

Usage

```
getCtryRasterOutputFname(ctryCode, nlType, nlPeriod)
```

Arguments

ctryCode	the ctryCode of interest
nlType	the nlType of interest
nlPeriod	the nlPeriod of interest

Value

Character full path to the cropped VIIRS country raster for a country and a given year and month

Examples

```
## Not run: getCtryRasterOutputFname("KEN","VIIRS", "201412")

#export for exploreData() shiny app
```

---

getCtryShpLowestLyrName

Get the name of the lowest ctry admin level

Description

Get the name of the lowest ctry admin level

Usage

```
getCtryShpLowestLyrName(ctryCode)
```

Arguments

ctryCode	the ctryCode of interest

Value

character string The name of the lowest admin level

`getCtryShpLyrName` *Get the standard name of a polygon layer for a country*

Description

Get the standard name of a polygon layer for a country. Used to refer to a polygon layer by name. i.e. for CTRYCODE & lyrNum="0": lyrName="CTRYCODE_adm0", lyrNum="1": lyrName="KEN_adm1". Note this #' is different from the country official administration level name.

Usage

```
getCtryShpLyrName(ctryCode, lyrNum)
```

Arguments

`ctryCode`	the ISO3 code for the country
`lyrNum`	the order of the layer starting from 0 = country level, 1 = first admin level

Value

Character layer name

Examples

```
lyrName <- getCtryShpLyrName("KEN","0") #top layer name
  #returns "KEN_adm0"

#@export only due to exploreData() shiny app
```

`getCtryTileList` *Returns a list of VIIRS nightlight tiles that a country or countries intersects with*

Description

Given a list of countries, this function will provide alist of VIIRS nightlight tiles that intersect with them. This helps in processing multiple countries by determining which nightlight tiles are required for processing by allowing the download of all required tiles before processing.

Usage

```
getCtryTileList(ctryCodes, nlType, omitCountries = "none")
```

Arguments

ctryCodes	character vector of country codes to process
nlType	character string The nlType of interest
omitCountries	countries to exclude from processing. This is helpful when the number of countries to exclude is smaller than the number to process e.g. when one wants to process all countries and exclude countries that take long to process i.e. omitCountries = "long"

Value

TRUE/FALSE

Examples

```
## Not run:
getCtryTileList(ctryCodes=c("BUR", "KEN", "RWA", "UGA", "TZA"),
    nlType="VIIRS", omitCountries="none")

getCtryTileList(ctryCodes="all", nlType="OLS", omitCountries="long")

## End(Not run)
```

---

getNlDataPath *Gets the root path to the file directory"*

Description

Gets the root path to the file directory"

Usage

```
getNlDataPath()
```

Value

Returns the folder containing the root of the current data path as a @character string.

See Also

To set the directory where package data files are stored, see @see "setNlDataPath".

Examples

```
print(getNlDataPath())
```

getNlDir *Get the paths to the various data locations*

Description

Get the paths to the various locations of nightlights data generated by the Rnightlights package. These correspond to the various "dir..." options in the pkgOptions settings

Usage

```
getNlDir(dirName)
```

Arguments

dirName	character vector The name of the directory to retrieve

Examples

```
getNlDir("dirRasterOutput")

getNlDir("dirNlTiles")

getNlDir("dirPolygon")

getNlDir("dirZonals")
```

getNlTifLclNameOLS *Constructs the filename used to save/access the decompressed OLS .tif file*

Description

Constructs the filename used to save/access the decompressed OLS .tif file

Usage

```
getNlTifLclNameOLS(nlYear)
```

Arguments

nlYear	the year in which the tile was created

Value

a character vector filename of the .tif VIIRS tile

Examples

```
#using default dirNlTiles
## Not run: getNlTifLclNameOLS("2004")
 #returns "OLS_2004.tif"
```

---

getNlTiles *Create mapping of nightlight tiles*

Description

Creates a data.frame mapping nightlight tile names to their vertice coordinates. This is used to identify nightlight tiles as well as to build a spatial polygons dataframe used to plot the tiles. OLS only has one tile for the whole world and thus has a dummy entry. OLS is included to prevent code duplication by writing separate functions for OLS.

Usage

```
getNlTiles(nlType)
```

Arguments

nlType	the nlType of interest

Value

A data.frame of names of tiles and lon-lat coordinate of top-left corner of each

Examples

```
## Not run: getNlTiles("VIIRS")

## Not run: getNlTiles("OLS")
```

---

getNlTileTifLclNameOLS *Constructs the filename of the decompressed OLS .tif file*

Description

Constructs the filename of the decompressed OLS .tif file

Usage

```
getNlTileTifLclNameOLS(nlYear)
```

Arguments

nlYear	the nlYear in which the tile was created

Value

a character vector filename of the .tif OLS tile

Examples

```
#using default dirNlTiles
## Not run: getNlTileTifLclNameVIIRS("2004")
 #returns "OLS_2004_00N180W.tif"
```

---

getNlTileTifLclNamePath

Constructs the full path used to save/access the downloaded tile .tgz file

---

Description

Constructs the full path used to save/access the downloaded tile .tgz file

Usage

```
getNlTileTifLclNamePath(nlType, nlPeriod, tileNum)
```

Arguments

nlType	the nlType of interest
nlPeriod	the nlPeriod in which the tile was created
tileNum	the index of the tile as given in nlTileIndex

Value

a character string filename of the compressed .tgz VIIRS tile

Examples

```
## Not run: getNlTileZipLclNamePath("OLS", "2014")
 #returns "/dataPath/ols_2014_01.tgz"

## Not run: getNlTileZipLclNamePath("VIIRS", "201412", "1")
 #returns "/dataPath/viirs_2014_12_75N180W.tgz"
```

getNlTileTifLclNamePathOLS

Constructs the full path used to save/access the decompressed OLS .tif file

Description

Constructs the full path used to save/access the decompressed OLS .tif file

Usage

```
getNlTileTifLclNamePathOLS(nlYear, tileNum)
```

Arguments

nlYear	the year in which the tile was created
tileNum	ignored

Value

a character vector filename of the .tif OLS tile

Examples

```
#using default dirNlTiles
## Not run: getNlTileTifLclNamePathOLS("2014", "1")
 #returns "/dataPath/tiles/OLS_2014.tif"
```

getNlTileTifLclNamePathVIIRS

Constructs the full path used to save/access the decompressed VIIRS .tif file

Description

Constructs the full path used to save/access the decompressed VIIRS .tif file

Usage

```
getNlTileTifLclNamePathVIIRS(nlYearMonth, tileNum)
```

Arguments

nlYearMonth	the yearMonth in which the tile was created
tileNum	the index of the tile as given in nlTileIndex

Value

a character vector filename of the .tif VIIRS tile

Examples

```
#using default dirNlTiles
## Not run: getNlTileTifLclNamePathVIIRS("201401", "1")
 #returns "/dataPath/tiles/VIIRS_2014_01_75N180W.tif"
```

`getNlTileTifLclNameVIIRS` *Constructs the filename of the decompressed VIIRS .tif file*

Description

Constructs the filename of the decompressed VIIRS .tif file

Usage

```
getNlTileTifLclNameVIIRS(nlYearMonth, tileNum)
```

Arguments

`nlYearMonth`	the nlYearMonth in which the tile was created
`tileNum`	the index of the tile as given in nlTileIndex

Value

a character vector filename of the .tif VIIRS tile

Examples

```
#using default dirNlTiles
## Not run: getNlTileTifLclNameVIIRS("201401", "1")
 #returns "VIIRS_201401_75N180W.tif"
```

getNlTileZipLclNameOLS *The name with which to save the OLS tile locally*

Description

The name with which to save the OLS tile locally

Usage

```
getNlTileZipLclNameOLS(nlYear)
```

Arguments

nlYear	The year of the OLS tile

Value

character string filename

Examples

```
## Not run: getNlTileZipLclNameOLS("2012")
```

getNlTileZipLclNamePath *Constructs the full path used to save/access the compressed downloaded tile*

Description

Constructs the full path used to save/access the compressed downloaded tile Calls the relevant function for the given nlType

Usage

```
getNlTileZipLclNamePath(nlType, nlPeriod, tileNum)
```

Arguments

nlType	the nlType of interest
nlPeriod	the nlPeriod in which the tile was created
tileNum	the index of the tile as given in nlTileIndex

Value

a character string filename of the compressed .tgz VIIRS tile

Examples

```
## Not run: getNlTileZipLclNamePath("VIIRS", "201401", "1")
 #returns "/dataPath/VIIRS_2014_01_75N180W.tgz"

## Not run: getNlTileZipLclNamePath("OLS", "2004", "1")
 #returns "/dataPath/OLS_2004.tgz"
```

---

getNlTileZipLclNameVIIRS

Constructs the filename used to save/access the downloaded VIIRS tile .tgz file

---

Description

Constructs the filename used to save/access the downloaded VIIRS tile .tgz file

Usage

```
getNlTileZipLclNameVIIRS(nlYearMonth, tileNum)
```

Arguments

nlYearMonth	The nlYearMonth in which the tile was created
tileNum	The index of the tile as given in nlTileIndex

Value

A character string filename of the compressed .tgz VIIRS tile

Examples

```
## Not run: getNlTileZipLclNameVIIRS("201401", "1")
 #returns "./tiles/VIIRS_2014_01_75N180W.tgz"
```

getNlUrlOLS *Function to return the url of the OLS tile to download*

Description

Function to return the url of the OLS tile to download given the year

Usage

```
getNlUrlOLS(nlYear)
```

Arguments

nlYear	The year of the tile for which to return the tile download URL

Value

character string Url of the OLS tile file

Examples

```
## Not run: tileUrl <- getNlUrlOLS("1999")
```

getNlUrlVIIRS *Function to return the url of the VIIRS tile to download*

Description

Function to return the url of the VIIRS tile to download given the year, month, and nlTile index

Usage

```
getNlUrlVIIRS(nlYearMonth, tileNum)
```

Arguments

nlYearMonth	character string yearmonth in "YYYYMM' format e.g. "201401"
tileNum	The integer index of the tile to download as given by getNlTiles("VIIRS")

Value

Character string Url of the VIIRS tile file

Examples

```
## Not run: tileUrl <- getNlUrlVIIRS("201401", "1")
```

`getPolyFname` *Returns the directory name of the unzipped shapefile downloaded from GADM.ORG without the path*

Description

Returns the directory name of the unzipped shapefile downloaded from www.GADM.ORG without the path

Usage

```
getPolyFname(ctryCode)
```

Arguments

`ctryCode`	character the ISO3 code of the country

Value

character path to zip

Examples

```
## Not run: getPolyFnameZip("KEN")
 #returns "path/to/"
```

`getPolyFnamePath` *Get the path of the unzipped polygon directory downloaded from GADM.ORG*

Description

Get the path of the unzipped polygon directory downloaded from GADM.ORG. Note the polygons are in ESRI Shapefile format thus when unzipped create a directory with the name <ctrycode>_adm_shp e.g. KEN_adm_shp. The directory will contain a number of files including the .shp file. `rgdal::readOGR` can read a shapefile polygon when given the directory path. It will determine which files to read.

Usage

```
getPolyFnamePath(ctryCode)
```

Arguments

`ctryCode`	character the ISO3 code of the country

Value

character path to polygon directory

Examples

```
getPolyFnamePath("KEN")
 #returns "path/to/"

#@export only due to exploreData() shiny app
```

`getPolyFnameZip` *Get the filename of the polygon zip file as downloaded from* `http://www.GADM.ORG`

Description

Get the filename of the polygon zip file as downloaded from `http://www.GADM.ORG`

Usage

```
getPolyFnameZip(ctryCode)
```

Arguments

`ctryCode`	character the ISO3 code of the country

Value

character path to zip

Examples

```
## Not run: getPolyFnameZip("KEN")
 #returns "path/to/"
```

getTilesCtryIntersectVIIRS

Get a list of tiles that a country polygon intersects with

Description

Create a dataframe mapping each country in the rworldmap to the VIIRS tiles which they intersect with and thus need to be retrieved to process their nightlight imagery. Since some functions use this dataframe for long-term processing, omitCountries can eliminate countries that should be excluded from the list hence from processing. Countries can be added in the omitCountries function. Default is "none".

Usage

```
getTilesCtryIntersectVIIRS(ctryCode)
```

Arguments

ctryCode	The country's ISO3 code

Value

None

Examples

```
getTilesCtryIntersectVIIRS("KEN")
```

insertNlDataCol

Insert an aggregate nightlight data column in a country nightlights dataframe

Description

Insert an aggregate nightlight data column in a country nightlights dataframe. The number of elements in the vector MUST match the number of rows in the country dataframe.

Usage

```
insertNlDataCol(ctryNlDataDF, dataCol, statType, nlPeriod, nlType)
```

Arguments

ctryNlDataDF	dataframe with the country data to save
dataCol	the numeric vector to be inserted as a column
statType	the stat which produced the dataCol vector
nlPeriod	the nlPeriod that the dataCol belongs to
nlType	the type of nightlight data i.e. "OLS" or "VIIRS"

Value

Character full path to the cropped VIIRS country raster for a country and a given year and month

Examples

```
## Not run: ctryNlDataDF <- insertNlDataCol(ctryNlDataDF, dataCol, "sum", "201409", "VIIRS")

## Not run: ctryNlDataDF <- insertNlDataCol(ctryNlDataDF, dataCol, "mean", "2012", "OLS")
```

---

listCtryNlData *List available data*

Description

List available data. If source is "local" it lists data cached locally. If source is remote lists available data on the remote repository.

Usage

```
listCtryNlData(ctryCodes = NULL, nlPeriods = NULL, nlTypes = NULL,
  source = "local")
```

Arguments

ctryCodes	A character vector of ctryCodes to filter by
nlPeriods	A character vector of nlPeriods to filter by
nlTypes	A character vector of nlTypes to filter by
source	Character string. Whether to check data availability "local" or "remote" Not in use.

Value

a list of countries and the periods and stats for each

Examples

```
#list all data
listCtryNlData()

#list all data available for KEN
listCtryNlData(ctryCodes = "KEN")

#list all VIIRS data available for ECU
listCtryNlData(ctryCodes = "ECU", nlTypes = "VIIRS")

#list available OLS data for KEN and RWA in 2012 & 2013
listCtryNlData(ctryCodes = c("KEN","RWA"), nlPeriods = c("2012", "2013"), nlTypes = "OLS")
```

---

listCtryNlRasters *List available cropped country rasters*

Description

List available data. If source is "local" it lists data cached locally. If source is remote lists available data on the remote repository.

Usage

```
listCtryNlRasters(ctryCodes = NULL, nlPeriods = NULL, nlTypes = NULL,
  source = "local")
```

Arguments

ctryCodes	A character vector of ctryCodes to filter by
nlPeriods	A character vector of nlPeriods to filter by
nlTypes	A character vector of nlTypes to filter by
source	Character string. Whether to check data availability "local" or "remote". Default is "local".

Value

a list of existing cropped rasters

Examples

```
#list all rasters
listCtryNlRasters()

#list all rasters available for KEN
listCtryNlRasters(ctryCodes = "KEN")

#list all VIIRS rasters available for ECU
```

```
listCtryNlRasters(ctryCodes = "ECU", nlTypes = "VIIRS")

#list available OLS rasters for KEN and RWA in 2012 & 2013
listCtryNlRasters(ctryCodes = c("KEN","RWA"), nlPeriods = c("2012", "2013"), nlTypes = "OLS")
```

`listNlTiles` *List locally cached tiles*

Description

List the tiles which have been downloaded previously and are currently cached in the local tiles folder

Usage

```
listNlTiles(nlTypes = NULL, nlPeriods = NULL, source = "local")
```

Arguments

`nlTypes`	A character vector of nlTypes to filter by
`nlPeriods`	A character vector of nlPeriods to filter by
`source`	Character string. Whether to check data availability. "local" or "remote". Default is "local".

Value

a list of locally cached nlTiles or NULL

Examples

```
#list all tiles
listNlTiles()

#list all VIIRS tiles
listNlTiles(nlTypes = "VIIRS")

#list all VIIRS tiles available in the years 2014-2015. Note VIIRS data
#starts in 201401
listNlTiles(nlTypes = "VIIRS", nlPeriods = nlRange("201401", "201512"))

#filter data
listNlTiles(nlTypes = "OLS", nlPeriods = c("2012", "2013"))
```

`mapAllCtryPolyToTilesVIIRS` *Create a mapping of all countries and the tiles they intersect*

Description

This is simply another name for mapCtryPolyToTilesVIIRS with ctryCodes="all"

Usage

```
mapAllCtryPolyToTilesVIIRS(omitCountries = pkgOptions("omitCountries"))
```

Arguments

`omitCountries`	A character vector or list of countries to leave out when processing. Default is "none"

Value

None

Examples

```
#no countries omitted
## Not run: mapAllCtryPolyToTilesVIIRS()

#no countries omitted
## Not run: mapAllCtryPolyToTilesVIIRS(omitCountries="none")

#include countries that take long to process
## Not run: mapAllCtryPolyToTilesVIIRS(omitCountries=c("error", "long"))
```

`mapCtryPolyToTilesVIIRS` *Create a mapping of all countries and the tiles they intersect*

Description

Create a dataframe mapping each country in the rworldmap to the VIIRS tiles which they intersect with and thus need to be retrieved to process their nightlight imagery. Since some functions use this dataframe for long-term processing, omitCountries can eliminate countries that should be excluded from the list hence from processing. Countries can be added in the omitCountries function. Default is "none".

Usage

```
mapCtryPolyToTilesVIIRS(ctryCodes = "all",
  omitCountries = pkgOptions("omitCountries"))
```

Arguments

ctryCodes	A character vector or list of countries to map. Default is "all"
omitCountries	A character vector or list of countries to leave out. Default is "none"

Value

ctryCodeTiles A data frame of countries and the tiles they intersect with as give by getNlTiles("VIIRS")

Examples

```
#map all countries
## Not run: mapCtryPolyToTilesVIIRS()

#map all countries, no countries omitted
## Not run: mapCtryPolyToTilesVIIRS(ctryCodes="all", omitCountries="none")

#will not omit countries that do not have polygons on GADM
## Not run: mapCtryPolyToTilesVIIRS(omitCountries=c("error", "missing"))
```

---

masqOLS *Extract raster pixel values within the boundaries of a polygon*

Description

Extract raster pixel values within the boundaries of a polygon

Usage

```
masqOLS(shp, rast, i)
```

Arguments

shp	the country Polygon layer as SpatialPolygon
rast	the clipped country raster
i	the index of the polygon in the country polygon layer (shp)

Value

numeric vector of radiances

Examples

```
## Not run:
ctryPoly <- readOGR(getPolyFnamePath("KEN"), getCtryShpLyrName("KEN", 1))
ctryRaster <- raster(getCtryRasterOutputFname("KEN", "OLS", "1999"))
temp <- NULL
KenAdm1Sum <- NULL
for (i in 1:5)#length(ctryPoly@polygons))
{
 temp$name <- as.character(ctryPoly@data$NAME_1[i])
 temp$sum <- sum(masqOLS(ctryPoly, ctryRaster, i), na.rm=T)

 KenAdm1Sum <- rbind(KenAdm1Sum)
}

## End(Not run)
```

---

masqVIIRS *extract data from a raster using one polygon in a multipolygon*

Description

extract data from a raster using one polygon in a multipolygon. Modified from https://commercedataservice.github.io/tutorial_viirs_part1/

Usage

```
masqVIIRS(ctryPoly, ctryRast, idx)
```

Arguments

ctryPoly	the country Polygon layer as SpatialPolygon
ctryRast	the clipped country raster
idx	the index of the polygon in the country polygon layer (shp)

Value

numeric vector of radiances

Examples

```
## Not run:
ctryPoly <- rgdal::readOGR('path/to/polygon.shp')

ctryRaster <- raster::raster('path/to/raster.tif')

#get the sum of nightlight pixels in the first polygon in a multipolygon
sumPolygon1 <- sum(masqVIIRS(ctryPoly, ctryRaster, 1), na.rm=T)
```

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

---

myZonal *Calculate zonal statistics. Used internally*

Description

Calculate zonal statistics. Used internally by zonalpipe. Modified from http://www.guru-gis.net/efficient-zonal-statistics-using-r-and-gdal/

Usage

```
myZonal(x, z, stats, digits = 0, na.rm = TRUE, ...)
```

Arguments

x	the country raster
z	the zonal country polygon layer
stats	a character list of statistics to calculate
digits	round off to how many decimals
na.rm	how to handle NAs
...	Other params to pass to the stats function

Value

numeric value result of the given stat function

---

nlCleanup *Clean up the environment after processing (Not yet implemented)*

Description

Clean up the environment after processing (Not yet implemented)

Usage

```
nlCleanup()
```

Examples

```
## Not run: nlCleanup()
```

nlInit *Initialize some important variables and create directory structure*

Description

Initialize some important variables and create directory structure

Usage

```
nlInit(omitCountries = "none")
```

Arguments

omitCountries	character string/vector CtryCodes to exclude from processing

Examples

```
## Not run: nlInit()
```

nlRange *Create a range of nlPeriods*

Description

Create a range of nlPeriods. Autodetects the type of nlPeriod and creates a character vector of nlPeriods filling in the intermediate nlPeriods. NOTE: Both start and end range must be valid and of the same type i.e. "OLS" years or "VIIRS" yearMonths

Usage

```
nlRange(startNlPeriod, endNlPeriod)
```

Arguments

startNlPeriod	the nlPeriod start
endNlPeriod	the nlPeriod end

Value

character vector of nlPeriods

Examples

```
#get OLS years between 2004 and 2010
nlRange("2004", "2010")

#get VIIRS yearMonths between Jan 2014 and Dec 2014
nlRange("201401", "201412")
```

---

pkgOptions *Set or get options for the Rnightlights package*

Description

Set or get options for the Rnightlights package

Usage

```
pkgOptions(...)
```

Arguments

...	Option names to retrieve option values or [key]=[value] pairs to set options.

Value

if an option name is supplied as a parameter this returns the value, else a list of all options is returned.

Supported options

The following options are supported

- cropMaskMethod(character) The method to use to clip the nightlight raster tiles to the country boundaries
- deleteTiles(character) whether to delete tiles after processing may be useful where diskspace is a concern
- dirNlData(character) The directory to store the extracted data files in
- dirNlDataPathcharacter) The root directory storing the package data
- dirNlTiles(character) The directory in which to store the downloaded VIIRS raster tiles
- dirPolygon(character) The directory to store the downloaded country administration level polygons
- dirRasterOutput(character) The directory in which to store the clipped country rasters
- dirRasterWeb(character) The directory in which to store the rasters resampled for web display
- dirZonals(character) The directory in which to store the zonal statistics country polygon

- downloadMethod(character) The download method to use
- extractMethod(character) The method to use to extract data from the rasters
- gdal_cachemax(character) The maximum memory gdal should use in gdal_rasterize
- ntLtsIndexUrlOLS(character) The url with the OLS tile index
- ntLtsIndexUrlVIIRS(character) The url with the VIIRS tile index
- numCores(character) The number of processor cores to use when extractMethod = "raster"
- omitCountries(character) The countries to exclude in processing
- stats(character) The statistics to calculate for country regions. The default are sum and mean. Any other aggregate statistics can be included. Also any aggregate function accessible in the current environment can be added.
- tmpDir(character) Change the temporary directory for processing rasters. Not in use

Examples

```
#retrieve the current cropMaskMethod
pkgOptions("cropMaskMethod")

#set the cropMaskMethod
pkgOptions(cropMaskMethod="gdal")

#retrieve all options
pkgOptions()
```

---

pkgReset *Reset global options for the Rnightlights package*

Description

Reset global options for the Rnightlights package

Usage

```
pkgReset()
```

Examples

```
#get cropMaskMethod
pkgOptions("cropMaskMethod") #returns default "rast"

#set cropMaskMethod to "gdal"
pkgOptions(cropMaskMethod="gdal") #sets to "gdal"

#check cropMaskMethod has changed
pkgOptions("cropMaskMethod") #returns "gdal"
```

```
#reset pkgOptions
pkgReset()

#check cropMaskMethod has been reset
pkgOptions("cropMaskMethod") #returns default "rast"
```

`plotCtryWithTilesVIIRS` *Plot a country polygon against a background of the VIIRS tiles and world map*

Description

Plot a country polygon as defined in the **rworldmap** package along with the VIIRS nightlight tiles for a visual inspection of the tiles required for download in order to process a country's nightlight data. Output corresponds to that of `getCtryNlTiles()`

It utilizes `rworldmap::rwmgetISO3()` to resolve country codes as well as names

Usage

```
plotCtryWithTilesVIIRS(idx)
```

Arguments

`idx`	character string or integer either the index of the country polygon in `rworldmap::getMap()` or the 3-letter ISO3 country code e.g. "KEN" or a common name of the country e.g. "Kenya" as found valid by `rworldmap::rwmgetISO3()`

Value

None

Examples

```
#by ctryCode
plotCtryWithTilesVIIRS("KEN")

#by index in rworldmap
plotCtryWithTilesVIIRS(115)

#by index passed as char string
plotCtryWithTilesVIIRS("24")
```

`processNLCountry` *Processes nightlights for an individual country in a particular nlPeriod*

Description

Given a `countryCode`, `yearMonth` and preferred processing methods `cropMaskMethod` and `extractMethod`, this function will first check if the data already exists in the cache. First it will check if the data file exists and if it does not it will create a dataframe of the country data containing only the administrative properties and move to processing. If the data file exists it will check to see if the particular year month already exists. If it exists, it will exit with a message. If it does not exist, it will load the country data file and move on to processing.

Usage

```
processNLCountry(ctryCode, nlPeriod, nlType,
  cropMaskMethod = pkgOptions("cropMaskMethod"),
  extractMethod = pkgOptions("extractMethod"),
  fnStats = pkgOptions("stats"))
```

Arguments

`ctryCode`	character string The ctryCode of interest
`nlPeriod`	character string The nlPeriod of interest
`nlType`	character string The nlType of interest
`cropMaskMethod`	("rast" or "gdal") Whether to use rasterize or gdal-based functions to crop and mask the country rasters
`extractMethod`	("rast" or "gdal") Whether to use rasterize or gdal-based functions to crop and mask the country rasters
`fnStats`	the statistics to calculate. If not provided will calculate the stats specified in `pkgOptions("stats")`

Details

Processing consists of:

1. Reading in the country polygon in ESRI Shapefile format
2. Reading in the tiles that the particular country intersects with and then clipping the tile(s) to the country boundary
3. Extract the data from the clipped raster and compute various statistics at the lowest admin level in the country.
4. Finally, these stats are appended to the data frame and written to the data file.

NOTE: processNLCountry() assumes that all inputs are available and will not attempt to download them. It should ideally be called from the function processNlData() which does all the preparation for processing. processNlData() which can process multiple countries and time periods will download all the required tiles and polygons prior to calling processnlcountry. getCtryNlData can also be used with the option ignoreMissing=FALSE which will call processNlData in the background.

Value

None

Examples

```
#calculate only the sum of VIIRS radiances for Dec 2014 using gdal
#for both cropMask and extraction for KEN
## Not run: processNLCountry("KEN", "201412", "VIIRS", "gdal", "gdal", "sum")
```

---

processNlData *Downloads nightlight tiles and country polygons and calls the function to process them*

Description

Downloads nightlight tiles and country polygons in preparation for the appropriate functions to process them. Given a list of countries and nlPeriods and an nlType, processNlData will first determine which countries and periods do not actually have data i.e. have not been previously processed. From the list of countries and prediods that need processing, it determines which nightlight tiles require to be downloaded. At the same time, it downloads any country polygons which have not already been downloaded.

Usage

```
processNlData(ctryCodes = getAllNlCtryCodes("all"),
  nlPeriods = getAllNlPeriods(nlType), nlType = "VIIRS",
  stats = pkgOptions("stats"))
```

Arguments

ctryCodes	the list of countries to be processed
nlPeriods	the nlPeriods of interest
nlType	the type of nightlights to process i.e. "OLS" or "VIIRS". Default "VIIRS"
stats	the statistics to calculate. If not provided will calculate the stats specified in pkgOptions("stats")

Details

processNlData then calls processNlCountry with the nlType supplied as a parameter. The processing is essentially the same for all nlTypes.

This is the main entry-point to the package and the main user-facing function. However, it works in batch mode and caches all data without returning any data to the user. However, it will return TRUE/FALSE depending on whether it completed successfully. Since it saves state regularly it can be run multiply in case of failure until it finally returns TRUE. This is where the only constraints are downloading and processing errors due to bandwidth or other resource constraints. A good example is running a long-running batch on an AWS EC2 spot-priced machine where the server may be decommissioned at any time. See more in the examples.

Value

None

Examples

```
## Not run:
#Example 1: process VIIRS nightlights for all countries and all periods available e.g. to create
    #a local cache or repo

    #Recommend running nlInit() to improve performance. It stores some global variables
    #so that they do not have to be re-evaluated multiply
    nlInit()

    processNlData() #process VIIRS nightlights for all countries and all periods

#Example 2: process nightlights for all countries in 2012 only

    nlInit() #for performance. See Example 1

   nlPeriods <- getAllNlYears("VIIRS") #get a list of all nightlight periods to present-day

    nlPeriods <- nlPeriods[grep("^2012", nlPeriods)] #filter only periods in 2012

    processNlData(nlPeriods=nlPeriods)

#Example 3: process VIIRS nightlights for countries KEN & RWA in 2014 Jan to 2014 May only

    nlInit()

    cCodes <- c("KEN", "RWA")

    nlPeriods <- getAllNlPeriods("VIIRS")

    nlPeriods <- nlPeriods[grep("^20140[1-5]", nlPeriods)]

    processNlData(ctryCodes=cCodes, nlPeriods=nlPeriods)

#Example 4: process VIIRS nightlights for countries KEN & RWA in 2014 Oct to 2014 Dec only
```

```
    processNlData(ctryCodes=c("KEN", "RWA"), nlPeriods=c("201410", "201411", "201412"))

#Example 5: process all nightlights, all countries, all stats in one thread

   processNlData()

#Example 6: process all nightlights, all countries, all stats with each
#   year in a separate thread. Create a separate R script for each year as follows:

    library(Rnightlights)

    nlInit()

    nlPeriods <- getAllNlYears("VIIRS")

    nlPeriods_2014 <- nlPeriods[grep("^2014", nlPeriods)]

    processNlData(nlPeriods=nlPeriods_2014)

    #Run the script from the command line as:

    #R CMD BATCH script_name_2014.R

## End(Not run)
```

`removeDataPath` *Deletes a root data path all sub-directories*

Description

Deletes a root data path and all sub-directories. It can be the current directory or a previously used data path. It will only delete if it has the default folder structure of a root data path.

Usage

```
removeDataPath(dataPath = file.path(getNlDataPath(), ".Rnightlights"))
```

Arguments

`dataPath`	The path to the root folder to be deleted

Value

None

Examples

```
## Not run: removeDataPath(getNlDataPath())
```

saveCtryNlData *Save a data frame of a country's data to the appropriate location*

Description

Saves the data frame created from processNlCountry* to the appropriate location. Note: This function does not perform any validation error checking and will overwrite existing data. Use with caution.

Usage

```
saveCtryNlData(ctryNlDataDF, ctryCode)
```

Arguments

ctryNlDataDF	dataframe with the country data to save
ctryCode	the ctryCode to which the data belongs

Value

None

Examples

```
## Not run: saveCtryNlData(ctryNlDataDF, ctryCode)
```

setNlDataPath *Sets the root path to the package data directory*

Description

By default, this function will set the root path to `~/.Rnightlights/`.

Usage

```
setNlDataPath(dataPath)
```

Arguments

dataPath	The path

Value

Returns (invisibly) the old root path.

Examples

```
## Not run: setNlDataPath("/new/path")
```

`setupDataPath` *Interactively allows the user to set up the default root path*

Description

Interactively allows the user to set up the default root path where data is cached

Usage

```
setupDataPath(newDataPath = tempdir(), ...)
```

Arguments

`newDataPath`	Default root path to set
`...`	Not used.

Value

character string Returns (invisibly) the root path, or @NULL if running a non-interactive session.

See Also

Internally, @see "setNlDataPath" is used to set the root path. The "`base::interactive`" function is used to test whether `R` is running interactively or not.

`tileIdx2Name` *Get the name of a tile given its index*

Description

Get the name of a VIIRS tile as given by getNlTiles() given its index

Usage

```
tileIdx2Name(tileNum, nlType)
```

Arguments

`tileNum`	index as given by getNlTiles()
`nlType`	the nlType of interest

Value

Character name of the tile

Examples

```
tileIdx <- tileName2Idx("00N060W", "VIIRS")
```

`tileName2Idx` *Get the index of a tile given its name*

Description

Get the index of a VIIRS tile as given by getNlTiles() given its name

Usage

```
tileName2Idx(tileName, nlType)
```

Arguments

`tileName`	name as given by getNlTiles()
`nlType`	the nlType of interest

Value

Integer index of the tile

Examples

```
tileIdx <- tileName2Idx("00N060W", "VIIRS")
```

`tilesPolygonIntersectVIIRS`
Get the list of VIIRS tiles that a polygon intersects with

Description

Get the list a VIIRS tiles that a polygon intersects with

Usage

```
tilesPolygonIntersectVIIRS(shpPolygon)
```

Arguments

shpPolygon	a SpatialPolygon or SpatialPolygons

Value

Character vector of the intersecting tiles as given by getNlTiles("VIIRS")

Examples

```
## Not run:
ctryShapefile <- dnldCtryPoly("KEN")
ctryPoly <- rgdal::readOGR(getPolyFnamePath("KEN"), getCtryShpLyrName("KEN",0))
tileList <- tilesPolygonIntersectVIIRS(ctryPoly)

## End(Not run)
```

---

validCtryCode *Check if a month number is valid for a given nightlight type*

Description

Check if a month number is valid for a given nightlight type. Note month num is only valid for "VIIRS" nightlight type

Usage

```
validCtryCode(ctryCode)
```

Arguments

ctryCode	the ISO3 country code to validate

Value

TRUE/FALSE

Examples

```
## Not run: validCtryCode("KEN")
 #returns TRUE

## Not run: validCtryCode("UAE")
 #returns FALSE. "United Arab Emirates" ISO3 code = "ARE"
```

validCtryNlDataDF *Check if a country dataframe is valid*

Description

Check if a country dataframe is valid

Usage

```
validCtryNlDataDF(ctryNlDataDF)
```

Arguments

ctryNlDataDF	the country dataframe

Value

TRUE/FALSE

Examples

```
## Not run: validCtryNlDataDF(nlCtryDataDF)
 #returns TRUE
```

validNlMonthNum *Check if a month number is valid for a given nightlight type*

Description

Check if a month number is valid for a given nightlight type. Note month num is only valid for "VIIRS" nightlight type

Usage

```
validNlMonthNum(monthNum, nlType = "VIIRS")
```

Arguments

monthNum	the month in "MM" format e.g. Jan="01", Feb="02"
nlType	type of nightlight either "VIIRS" or "OLS"

Value

TRUE/FALSE

Examples

```
## Not run: validNlMonthNum("01","VIIRS")
 #returns TRUE

## Not run: validNlMonthNum("13","VIIRS")
 #returns FALSE

## Not run: validNlMonthNum("01","OLS")
 #returns FALSE
```

`validNlPeriod` *Check if an nlPeriod is valid for a given nightlight type*

Description

Check if an nlPeriod is valid for a given nightlight type

Usage

```
validNlPeriod(nlPeriod, nlType)
```

Arguments

`nlPeriod`	the nlPeriod of interest
`nlType`	type of nightlight either "VIIRS" or "OLS"

Value

TRUE/FALSE

Examples

```
validNlPeriod("201401","VIIRS")
 #returns TRUE

validNlPeriod("201203","VIIRS")
 #returns FALSE

validNlPeriod("2012","OLS")
 #returns TRUE
```

validNlPeriodOLS *Check if an OLS nlYear is valid*

Description

Check if an OLS nlYear is valid

Usage

```
validNlPeriodOLS(nlYear)
```

Arguments

nlYear	the year in "YYY" format e.g. "2012"

Value

TRUE/FALSE

Examples

```
## Not run: validNlPeriodOLS("2015")
 #returns FALSE

## Not run: validNlPeriodOLS("2004")
 #returns TRUE

## Not run: validNlPeriodOLS("201201")
 #returns FALSE
```

validNlPeriodVIIRS *Check if a VIIRS nlYearMonth is valid*

Description

Check if a VIIRS nlYearMonth is valid Note this function is only valid for the "VIIRS" nightlight type

Usage

```
validNlPeriodVIIRS(nlYearMonth)
```

Arguments

nlYearMonth	the yearmonth in "YYYYMM" format e.g. "201410", "201701"

Value

TRUE/FALSE

Examples

```
## Not run: validNlPeriodVIIRS("201512")
 #returns TRUE

## Not run: validNlPeriodVIIRS("201513")
 #returns FALSE; invalid month 13

## Not run: validNlPeriodVIIRS("201401")
 #returns FALSE #VIIRS starts in "201401"
```

`validNlTileNameVIIRS` *Check if a tile index name is valid for a given nightlight type*

Description

Check if a tile number is valid for a given nightlight type. Note tile num is only valid for "VIIRS" nightlight type

Usage

```
validNlTileNameVIIRS(tileName)
```

Arguments

`tileName`	the name of the tile

Value

TRUE/FALSE

Examples

```
validNlTileNameVIIRS("00N060W")
 #returns TRUE
```

validNlTileNumVIIRS *Check if a tile index number is valid for a given nightlight type*

Description

Check if a tile number is valid for a given nightlight type. Note tile num is only valid for "VIIRS" nightlight type

Usage

```
validNlTileNumVIIRS(nlTileNum)
```

Arguments

nlTileNum	the index of the tile

Value

TRUE/FALSE

Examples

```
validNlTileNumVIIRS("1")
 #returns TRUE

validNlTileNumVIIRS("9")
 #returns FALSE
```

validNlYearNum *Check if a year is valid for a given nightlight type*

Description

Check if a year is valid for a given nightlight type

Usage

```
validNlYearNum(yearNum, nlType)
```

Arguments

yearNum	the year of interest
nlType	type of nightlight either "VIIRS" or "OLS"

Value

TRUE/FALSE

Examples

```
## Not run: validNlYearNum("2014","VIIRS")
```

`validStat` *Check if a statistic given is valid*

Description

Check if a statistic given is valid

Usage

```
validStat(stat)
```

Arguments

`stat`	the statistic to check

Value

TRUE/FALSE

Examples

```
## Not run: validStat("sum")
 #returns TRUE
```

`ZonalPipe` *Create a zonal file if it does not exist and calculate the zonal stats*

Description

Create a zonal file if it does not exist and calculate the zonal stats by calling the myZonal function. Modified from http://www.guru-gis.net/efficient-zonal-statistics-using-r-and-gdal/

Usage

```
ZonalPipe(ctryCode, ctryPoly, path.in.shp, path.in.r, path.out.r, path.out.shp,
  zone.attribute, stats)
```

Arguments

`ctryCode`	the ctryCode of interest
`ctryPoly`	the SpatialPolygonsDataFrame country polygon to process
`path.in.shp`	The path to the country shapefile
`path.in.r`	The path to the raster tile
`path.out.r`	The path where to save the output zonal raster
`path.out.shp`	The path to save the output zonal shapefile (Ignored)
`zone.attribute`	The zonal attribute to calculate
`stats`	The stats to calculate

Value

TRUE/FALSE

Index