Package 'rts'

October 14, 2022

Type Package

Title Raster Time Series Analysis

Version 1.1-8

Date 2022-09-16

Author Babak Naimi

Depends R (>= 3.5.0), terra, xts

Imports methods, sp (>= 1.4.1), zoo, RCurl, raster

Suggests digest, R.rsp

Maintainer Babak Naimi <naimi.b@gmail.com>

Description This framework aims to provide classes and methods for manipulating and processing of raster time series data (e.g. a time series of satellite images).

License GPL (>= 3)

URL http://r-gis.net

VignetteBuilder R.rsp

NeedsCompilation no

Repository CRAN

Date/Publication 2022-09-16 20:16:07 UTC

R topics documented:

`apply.monthly` *Apply a function over calendar periods*

Description

Apply a specified function to each distinct period in a given raster time series object.

Usage

```
apply.daily(x, FUN, ...)
apply.weekly(x, FUN, ...)
apply.monthly(x, FUN, ...)
apply.quarterly(x, FUN, ...)
apply.yearly(x, FUN, ...)
```

Arguments

`x`	a raster time series (`Raster*TS`) object, created by `rts`
`FUN`	an R function
`...`	additional arguments to FUN

Details

These functions offer Simple mechanism to apply a function to non-overlapping time periods, e.g. weekly, monthly, etc, and return a raster time series object including a raster layer for each period in the original data, produced by `FUN`. The end of each period of time is assigned to the corresponding raster layer in the output.

Value

A raster time series (`Raster*TS`) object

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

endpoints, period.apply,

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file
ndvi
ndvi.y <- apply.yearly(ndvi, mean) # apply mean function for each year
ndvi.y
ndvi.q <- apply.quarterly(ndvi,sd) # apply sd function for each quarter of years
ndvi.q

## End(Not run)
```

---

cellFromXY *Get cell number from row, column or XY*

Description

Get cell number(s) of a Raster*TS object from row and column numbers or X and Y coordinates.

Usage

```
## S4 method for signature 'RasterStackBrickTS,numeric,numeric'
cellFromRowCol(object, row, col)
## S4 method for signature 'RasterStackBrickTS'
cellFromXY(object, xy)

## S4 method for signature 'SpatRasterTS,numeric,numeric'
cellFromRowCol(object, row, col)
## S4 method for signature 'SpatRasterTS'
cellFromXY(object, xy)
```

Arguments

object	SpatRaster
col	integer. column number(s)
row	integer row number(s)
xy	matrix of x and y coordinates

Details

These functions are essentially a wrapper to cellFromRowCol and cellFromXY in **raster** package, work with Raster*TS objects.

Value

row, column or cell number(s). cellFromLine and cellFromPolygon return a list.

Author(s)

Babak Naimi <naimi.b@gmail.com> http://r-gis.net

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

rt <- rts(file) # read the ndvi time series from the specified file
cellFromRowCol(rt,15,20)

cellFromRowCol(rt,c(16:20),c(11:15))

cellFromXY(rt,c(645000,57345000))

## End(Not run)
```

---

endpoints *Locate endpoints by time*

Description

Extract index values of a given Raster*TS object corresponding to the *last* observations given a period specified by on.

Usage

```
endpoints(x, on="months", k=1)
```

Arguments

x	a raster time series (Raster*TS) object created by rts
on	the periods endpoints to find as a character string
k	along every k-th element - see notes

Details

This function is, indeed, endpoints in **xts** that works with Raster*TS objects. It returns a numeric vector corresponding to the *last* observation in each period specified by on, with a zero added to the beginning of the vector, and the index of the last raster in x at the end.

Valid values for the argument on include: “us” (microseconds), “microseconds”, “ms” (milliseconds), “milliseconds”, “secs” (seconds), “seconds”, “mins” (minutes), “minutes”, “hours”, “days”, “weeks”, “months”, “quarters”, and “years”.

Value

A numeric vector of endpoints beginning with 0 and ending with a value equal to the number of raster layers in the x argument.

Note

Windows support for subsecond periods is not supported.

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

endpoints

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

endpoints(ndvi,"years")

endpoints(ndvi,"quarters")

## End(Not run)
```

---

extract *Extract values from raster time series*

Description

Extract values from a Raster*TS object for the spatial locations which can be specified by spatial points, lines, polygons or an Extent (rectangle) object or raster cell number(s).

Details

This function uses the **raster** and **xts** packages to extract the values in space and subset them in time by specifying ISO-8601 compatible range strings. This allows for natural range-based time queries without requiring prior knowledge of the underlying time object used in construction.

When a raw character vector is used for the time, it is processed as if it was ISO-8601 compliant. This means that it is parsed from left to right, according to the following specification:

CCYYMMDD HH:MM:SS.ss+

A full description will be expanded from a left-specified truncated one.

Additionally, one may specify range-based queries by simply supplying two time descriptions seperated by a forward slash:

CCYYMMDD HH:MM:SS.ss+/CCYYMMDD HH:MM:SS.ss

The algorithm to parse the above is `.parseISO8601` from the **xts** package.

Value

An `rts` object.

Methods

`extract(x, y, time)`

Arguments

`x` is a raster time series (`Raster*TS`) object created by `rts`

`y` is a `SpatialPoints*`, `SpatialPolygons*`, `SpatialLines`, `Extent` object, or a vector (representing cell numbers)

`time` is Optional; the time index for which the values in raster should be extracted. It can be numeric, timeBased or ISO-8601 style (see details)

Author(s)

Babak Naimi

`<naimi.b@gmail.com>`

http://r-gis.net

See Also

`[.xts` and `extract`

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

n1 <- extract(ndvi,125)# extract the time series values at cell number 125 for all times

n1

plot(n1)

n2 <- extract(ndvi,125,"/20090101") # extract the time series values at cell number 125
# for all times after 2009-01-01

n2
```

```
plot(n2)

n3 <- extract(ndvi,125,"200901/") # extract the time series values at cell number 125 for
# all times before 2009-01

n4 <- extract(ndvi,10:20,"2008") # extract the values at cell numbers of 10:20 in
# the year of 2008

n4

## End(Not run)
```

`Extract by index` *Exrtract values from raster time series*

Description

This is a short-hand method that acts the same as `extract`. This method extracts values from a `Raster*TS` object for spatial locations which can be specified by spatial points, lines, polygons, or an Extent (rectangle) object or raster cell number(s).

Details

This function uses the **raster** and **xts** packages to extract the values in space and subset them in time by specifying ISO-8601 compatible range strings. This allows for natural range-based time queries without requiring prior knowledge of the underlying time object used in construction.

When a raw character vector is used for `j`, it is processed as if it was ISO-8601 compliant. This means that it is parsed from left to right, according to the following specification:

CCYYMMDD HH:MM:SS.ss+

A full description will be expanded from a left-specified truncated one.

Additionally, one may specify range-based queries by simply supplying two time descriptions seperated by a forward slash:

CCYYMMDD HH:MM:SS.ss+/CCYYMMDD HH:MM:SS.ss

The algorithm to parse the above is `.parseISO8601` from the **xts** package.

Value

`rts`.

Methods

x[i, j]

Arguments

x is a raster time series (Raster*TS) object created by rts

i is a SpatialPoints*, SpatialPolygons*, SpatialLines, Extent object, or a vector (representing cell numbers)

j is optional; the time index for which the values in raster should be extracted. It can be numeric, timeBased or ISO-8601 style (see details)

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

extract and [.xts

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

n1 <- ndvi[125] # extract the time series values at cell number 125 for all available times

n1

plot(n1)

n2 <- ndvi[125,"/20090101"] # extract the time series values at cell number 125 for
# all times after 2009-01-01

n2

plot(n2)

n3 <- ndvi[125,"200901/"] # extract the time series values at cell number 125 for all
# times before 2009-01

n4 <- ndvi[10:20,"2008-05-01"] #extract the values at cell numbers of 10:20 for
# the specified time

n4



## End(Not run)
```

index *Extracting and replacing the index of raster time series*

Description

index is a generic function for extracting the index of a raster time series (Raster*TS) object and replacing it.

index(x) <- value, can be used to replace index with value, a vector of the same length as the number of raster layers in Raster*TS object .

Usage

```
## S3 method for class 'RasterStackBrickTS'
index(x, ...)
```

Arguments

x	A RasterStack or RasterBrick object
...	further arguments passed to methods

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

index.xts

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

index(ndvi)


## End(Not run)
```

ModisDownload *Download, reproject, and mosaic Modis satellite images*

Description

'ModisDownload' downloads a series of MODIS images in a specific date or a period of times, and for specified tile(s). It can also use MODIS Reproject Tool (MRT) software to mosaic the downloaded images, in case of selecting more than one tile, and reproject them to a specified coordinate system. As the format of the source images in LP DAAC is HDF, this tool can also convert them into other formats (i.e. Geotif, hdr).

Usage

```
setNASAauth(username,password,update,...)

setMRTpath(MRTpath,update,...)

ModisDownload(x,h,v,dates, ...)

getMODIS(x,h,v,dates,version='006',forceReDownload=TRUE,ncore='auto')

mosaicHDF(hdfNames,filename,MRTpath,bands_subset,delete=FALSE)

reprojectHDF(hdfName,crs,subset,resample_method,filename,...)

modisProducts(version=NULL)

getNativePixelSize(product)

getNativeTemporalResolution(product)
```

Arguments

username	character; username for authentication
password	character; password for authentication
update	logical; specifies whether .nerc file that has been previously created by setNASAauth, should be updated
x	product name or a number specifies the product row in the data.frame generated by `modisProduct` function. It can also be the http address of the product.
h	together with v specify the position of the Modis image tile(s). Examples: h=15; h=c(14,15); v=c(4:7)
v	see the above descriptopn for h
dates	a character vector with the length of one or two, specifiying an individual date or a range of dates as the form of `c(from, to)`
hdfName	the name of hdf file

crs	coordinate reference system definiation (or a SpatRaster defines definition of new Raster for resampling)
subset	integer vector specifies the layer number for subsetting bands
resample_method	resample method for reprojection based on GDAL, can be one of "near","bilinear","cubic",or "cubicspline" (default: "near")
hdfNames	the names of hdf files
MRTpath	Path to the bin folder into the installed directory of MRT software. It is needed only if you need to mosaic and/or reproject the images. Example: MRTpath="d:/MRT/bin"
filename	the name of new file
version	the product version. The default value is '006'
bands_subset	HDF-EOS input files contain several layers of data (bands). Through this argument you ca select a subset of bands. You need to know how many bands the product has, and which bands you want to be subset. Example: Suppose your image has 6 bands and you only need the second band. The parameter should be entered as: bands_subset="0 1 0 0 0 0".
delete	Logical. If TRUE, the original HDF files will be deleted after mosaic or reproject into new files.
forceReDownload	logical, specify whether the product should be re-downloaded if it had been downloaded before. The downloaded products are cashed using a hash digest of the inputs. If the download is canceled mid-way and the same exact products are downloaded again, it can be useful to avoid re-downloading them.
ncore	numerical; specify the number of cores used to speed up the download; 'auto' can be used instead to detect the number of cores and use half of them
product	The product name; can be extracted from the modisProduct function
...	additional arguments including: - mosaic: logical, if TRUE, the images (several tiles) are mosaic into a single image - proj: Logical. If TRUE, it means that you want to reproject the images. Then reprojectHDF function will be called to reproject the images. - UL: Optional. Upper Left coordinate (x,y) in output coordinate system. Only if you want to spatially subset your images. - LR: Optional. Lower right coordinate(x,y). Only if you want to spatially subset your images. - resample_type: Resampling kernel type for MRT ("CUBIC_CONVOLUTION", "NEAREST_NEIGHBOR", or "BILINEAR"). The default is "NEAREST_NEIGHBOR". - proj_type: Output projection short name. Valid values are "AEA" (Albers EqualArea), "ER" (Equirectangular), "GEO" (Geographic), "IGH" (Interrupted Goode Homolosine), "HAM" (Hammer), "ISIN" (Integerized Sinusoidal), "LA" (Lambert Azimuthal Equal Area), "LCC" (Lambert Conformal Conic), "MERCAT" (Mercator), "MOL" (Molleweide), "PS" (Polar Stereographic), "SIN" (Sinusoidal), "TM" (Transverse Mercator), and "UTM" (Universal Transverse Mercator).

- proj_params: Output projection parameters. This quoted, floating-point list includes up to 15 projection parameters, with each value separated by white space ("p1 p2 ... p15"). If there are fewer than 15 values specified in the list, the remaining values will be set to zero. Integer values will automatically be converted to floating point (See the MRT software manual for the details).

- datum: Specifies the output projection datum. The default is "WGS84"

- utm_zone: Valid only if "UTM" is selected for the proj_type.

- pixel_size: Output pixel size.

Details

To be able to download the data, you need to register on "https://urs.earthdata.nasa.gov/" and get a username and password. To pass the authentication by the website, you need to set the username and password on the machine (only first time) using setNASAauth function. Just specify the username and password in the function for the first time, and then the ModisDownload function can use it everytime you need to download the data.

To have the functionality for Mosaic and reprojecting of the images, you need to first install MRT software on your machine, and introduce its' path through the MRTpath argument. Otherwise, it can only be used for automating the downloading procedure.

The functions ModisDownload uses the functions including getMODIS, mosaicHDF (if needed), and reprojectHDF (if needed). The functions getMODIS, can be used to download HDF files, while mosaicHDF and reprojectHDF can mosaic and reproject the HDF files, respectively.

Author(s)

Babak Naimi & Pablo Alfaro

<naimi.b@gmail.com>

http://r-gis.net

Examples

```
## Not run:
library(raster)

library(RCurl)

# First, you need to register on https://urs.earthdata.nasa.gov/ and get a username and password
# for the first time, set the authentication info:

setNASAauth(username='myusername',password='mypass')

# product list:


modisProducts(version=5)
modisProducts(version=6)
modisProducts(version=NULL) # both versions

#x=3 # or x="MOD14A1"
```

```
# download 4 tiles (h14v04, h14v05, h15v04, h15v05) in single date (2011.05.01)

# Following command only downloads the source HDF images, no mosaic and no projection

ModisDownload(x=x,h=c(17,18),v=c(4,5),dates='2011.05.01',mosaic=F,proj=F,version='006')


# alternatively, you can use getMODIS to download only HDF images:

getMODIS(x=x,h=c(17,18),v=c(4,5),dates='2011.05.01',version='006')

# same as the above command, but downloads all available images in 2011:

ModisDownload(x=x,h=c(17,18),v=c(4,5),dates=c('2011.01.01','2011.12.31'),version='006')

#------

# Downloads selected tiles, and mosaic them, but no projections:

ModisDownload(x=x,h=c(17,18),v=c(4,5),dates=c('2011.05.01','2011.05.31'),
              MRTpath='d:/MRT/bin',mosaic=T,proj=F,version='006')

#--- alternatively, you can first download the HDF images using getMODIS,
#and then mosaic them using mosaicHDF!

# Downloads selected tiles, and mosaic, reproject them in UTM_WGS84, zone 30 projection and
#convert all bands into Geotif format (the original HDF will be deleted!):

ModisDownload(x=x,h=c(17,18),v=c(4,5),dates=c('2011.05.01','2011.05.31'),MRTpath='d:/MRT/bin',
              mosaic=T,proj=T,proj_type="UTM",utm_zone=30,datum="WGS84",
              pixel_size=1000,version='006')

# Same as above command, but only second band out of 6 bands will be kept. (You do not need
#to specify proj_params when "UTM" is selected as proj_type and the zone also is specified,
#but for other types of projections you do).

ModisDownload(x=x,h=c(17,18),v=c(4,5),dates=c('2011.05.01','2011.05.31'),MRTpath='d:/MRT/bin',
              mosaic=T,proj=T, bands_subset="0 1 0 0 0 0", proj_type="UTM",
              proj_params="-3 0 0 0 0 0 0 0 0 0 0 0 0 0 0",utm_zone=30,
              datum="WGS84",pixel_size=1000,version='006')


# Same as above command, but it spatially subsets the images into the specified box (UL and LR):

ModisDownload(x=x,h=c(17,18),v=c(4,5),dates=c('2011.05.01','2011.05.31'),MRTpath='d:/MRT/bin',
             mosaic=T,proj=T,UL=c(-42841.0,4871530.0),LR=c(1026104,3983860),
             bands_subset="0 1 0 0 0 0", proj_type="UTM",
             proj_params="-3 0 0 0 0 0 0 0 0 0 0 0 0 0 0",utm_zone=30,datum="WGS84",
             pixel_size=1000,version='006')
```

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

`period.apply` *Apply a function over specified time intervals*

Description

Apply specified function over each period of date/time defined in INDEX at each grid cell.

Usage

```
period.apply(x, INDEX, FUN, ...)
```

Arguments

x	a raster time series (Raster*TS) object created by rts
INDEX	a numeric vector of endpoints of time/date periods to apply function over
FUN	an argument of type function
...	additional arguments for FUN

Details

This functions subsets the raster data based on the specified time periods (endpoint for each period should be specified in INDEX), and FUN function will be applied to the subsetted values at each grid cell for each period. For each period, a raster will be calculated and the end of the date/time period will be assigned to it in the output raster time series object. If the INDEX is out of range, the function stops working and an error is generated.

Value

RasterStackTS or RasterBrickTS

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

period.apply

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

ndvi

ep <- endpoints(ndvi,'years') # extract the end index on each year period

ndvi.y <- period.apply(ndvi,ep,mean) # apply the mean function on each year

ndvi.y

#---------
ep <- endpoints(ndvi,'quarters') # extract the end index on each quarter of a year

# a function:
f <- function(x) {
  if (min(x) > 0.5) mean(x)
  else 0
}

ndvi.q <- period.apply(ndvi,ep,f) # apply the function f on each quarter


## End(Not run)
```

---

plot *Plot raster time series*

Description

Plot raster layers corresponding to specified times, or plot extracted time-series values at a location (cell) or a number of cells as a `rts` object.

Usage

```
## S4 method for signature 'RasterStackBrickTS,ANY'
plot(x, y, ...)
## S4 method for signature 'rts,ANY'
plot(x, y, ...)
```

Arguments

x	an object of raster time series class (Raster*TS), or an object of rts class.
y	optional. if x is a Raster*TS object, this item defines time range specifying which layers of raster time series should be plotted. if x is an rts object, this item specifies which column (corresponding to a cell) of time-series data should be plotted (default=1). y='all' indicates all series in rts object should be included in plot.
...	additional argument as in plot in raster package or in graphics package.

Details

If x is a Raster*TS object:

This function, first, selects the layers corresponding to the time range specified in y and then call plot function in raster package to plot the selected raster layers. If y is not specified, all layers will be sent to plot function.

Same as in extract and subset functions, When a raw character vector is used for the y, it is processed as if it was ISO-8601 compliant. This means that it is parsed from left to right, according to the following specification:

CCYYMMDD HH:MM:SS.ss+

A full description will be expanded from a left-specified truncated one.

Additionally, one may specify range-based queries by simply supplying two time descriptions seperated by a forward slash:

CCYYMMDD HH:MM:SS.ss+/CCYYMMDD HH:MM:SS.ss

x can be an rts object. rts is a subclass of xts, created by extract function.

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

plot, extract, subset

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

plot(ndvi,1:4) # plot the first 4 layers in the raster time series

plot(ndvi,"/20010101")
```

```
plot(ndvi,"20010101/20010501")

plot(ndvi,"2001-02-01/2001-05-01")

plot(ndvi,"2001-02-01")


n1 <- extract(ndvi,125)# extract the time series values at cell number 125 for all times


plot(n1)

n2 <- extract(ndvi,125,"/20090101") # extract the time series values at cell number 125
# for all times after 2009-01-01

n2

plot(n2)


n3 <- extract(ndvi,125:127,"200901/") # extract the time series values at the specified cell
# numbers for all times before 2009-01

n3

plot(n3) # plot the time-series values for the first series in n3 (i.e cell: 125)

plot(n3,y=1:3) # plot for the 3 series in n3 (i.e cells of 125:127)

plot(n3,y=1:3,col=1)

plot(n3,y=1:3,col=c(1,4,5))

plot(n3,y='all')

## End(Not run)
```

RasterStackBrickTS-class *rts classes*

Description

A raster time series contains a collection of RasterLayer objects, each corresponds to a time/date.

RasterStackTS and RasterBrickTS classes are created by putting together a RasterStack or RasterBrick object, from the **raster** package, and an xts object, from the **xts** package. A RasterStack and RasterBrick represents a collection of RasterLayer objects with the same extent and resolution. An xts object extends the S3 class zoo from the package of the same name. This object

provides the index values that is unique and ordered, and also is a time-based class. Currently acceptable classes include: 'Date', 'POSIXct', 'timeDate', as well as 'yearmon' and 'yearqtr' where the index values remain unique.

`rts` is a subclass of `xts` class.

Slots

Slots for Raster*TS object:

`raster`: object of class `RasterStack` or `RasterBrick`

`rime`: object of class `xts`

See also `Raster-class` for slots in `raster`.

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

Examples

```
showClass("RasterStackTS")
```

`read.rts` *Read raster time Series data from a file*

Description

Read a raster time series object from a file.

Usage

```
read.rts(filename,...)
```

Arguments

`filename`	Filename (character)
`...`	see details

Details

This function reads a raster time series object which has been writen by `write.rts`. Instead of `read.rts`, the `rts` function can be used (usage: `rts(filename)`).

By default, the Raster Time Series is read as the class of `SpatRasterTS`, but the user can provide `cls` argument to specify a different rts class (e.g., `cls='RasterBrickTS'`)

Value

RasterBrickTS

Author(s)

Babak Naimi
<naimi.b@gmail.com>
http://r-gis.net

See Also

write.rts, rts

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

rt <- read.rts(file)

rt

# or alternatively:
rt <- rts(file)

## End(Not run)
```

---

rts *Create a Raster Time Series object*

Description

Constructor function to create a raster time series (Raster*TS) object. rts object can be created from a vector of image files names, a RasterStack or a RasterBrick object (defined in **raster**) together with a vector of time/dates-must be of known time-based class. This function can also be used to read a raster time series file.

Usage

```
rts(x, time,...)
```

Arguments

x	A character vector including names of image/raster files, or RasterStack or RasterBrick object, or the name (character) of a raster time series file
time	a vector holding date/time data with the same length as rasters in Raster* object or name of files in character vector
...	see details

Details

A raster time series object is created by combining a RasterStack or RasterBrick objct, defined in **raster** and a xts object in xts-package. RasterStack or RasterBrick can be created by using stack and brick functions, respectively in raster-package. If a character vector including the name of raster files is used for x, stack function is internally called by rts. time information is handled by xts object. The date/time values in the vector of time should be correspond to the raster files (i.e. first date/time for first raster, ...) and have the same length as the number of rasters in x.

If a name of a raster time series file is provided for the x argument, it acts the same as coderead.rts.

If x is the name of Raster Time Series file (a character), it calls read.rts to read the file. By default, the Raster Time Series is read as the class of SpatRasterTS, but the user can provide cls argument to specify a different rts class (e.g., cls='RasterBrickTS')

Value

RasterStackTS or RasterBrickTS

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

stack, brick, xts

Examples

```
## Not run:
path <- system.file("external", package="rts") # location of files

lst <- list.files(path=path,pattern='.asc$',full.names=TRUE)
lst # list of raster files


r <- stack(lst) # creating a RasterStack object

r

d <- c("2000-02-01","2000-03-01","2000-04-01","2000-05-01") # corresponding dates to 4 rasters
d <- as.Date(d) # or d <- as.POSIXct(d)

rt <- rts(r,d) # creating a RasterStackTS object

rt

plot(rt)
```

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

---

subset *Subset layers in a raster time series object*

Description

Extract layers from a Raster*TS object.

Details

This function can be used to extract a raster layer or a set of raster lasyers based on the time-index using date-like string. The format must left-specied with respect to the standard ISO:8601 time format "CCYY-MM-DD HH:MM:SS". It is also possible to specify a range of times via the index-based subsetting, using ISO-recommended "/" as the range operator. The basic form is *"from/to"*, where both are optional. If either side is missing, it is interpretted as a request to retrieve raster layers from the beginning, or through the end of the raster time series object. Both subset function and '[[' operator do the same thing.

Value

RasterStackTS or RasterBrickTS.

Methods

subset(x, subset, ...)

Arguments

x is a raster time series (Raster*TS) object created by rts

subset is indicates the layers (represented as a vector of numeric or character string relevant to time index, or by a time-based object).

... - same as ... in subset function in package **raster**

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

subset and [.xts

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

s1 <- subset(ndvi,1:5) # subset the first 5 raster layers into a new raster time series object

s1

plot(s1)

s2 <- subset(ndvi,"/2000") # subset all layers till end of year 2000

s2

plot(s2)

s3 <- subset(ndvi,"2000-01-01/2000-05-31")

s3

plot(s3)


## End(Not run)
```

---

subset by index *Subset layers in a raster time series object by index*

Description

Extract layers from a Raster*TS object by index (i.e. double bracket, [[).

Details

This function can be used to extract a raster layer or a set of raster lasyers based on the time-index using date-like string. The format must left-specied with respect to the standard ISO:8601 time format "CCYY-MM-DD HH:MM:SS". It is also possible to specify a range of times via the index-based subsetting, using ISO-recommended "/" as the range operator. The basic form is *"from/to"*, where both are optional. If either side is missing, it is interpretted as a request to retrieve raster layers from the beginning, or through the end of the raster time series object. Both subset function and '[[' operator do the same thing.

Value

RasterStackTS or RasterBrickTS.

Methods

x[[i, ...]]

Arguments

i - indicates the layers (represented as a vector of numeric or character string relevant to time index, or by a time-based object).

... - same as ... in subset function in package **raster**

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

subset

Examples

```
## Not run:
file <- system.file("external/ndvi", package="rts")

ndvi <- rts(file) # read the ndvi time series from the specified file

s1 <- ndvi[["2000-01-01/2000-05-31"]]

s1

plot(s1)


## End(Not run)
```

VHPdownload — *Download AVHRR-based Vegetation and Drought satellite image products*

Description

'VHPdownload' downloads a series of AVHRR and VIIRS images in a specific date or a period of dates. The format of the files are downloaded as GeoTiff, and they can be optionally returned as a Raster Time Series Object.

Usage

```
VHPdownload(x,dates,rts,ncore,...)
```

Arguments

x	product name; it can be either of c('VHI','VCI','SMN','SMT','TCI')
dates	a vector, character or Date, with one or two items, specifiying an individual date or a range of dates as the form of c(from, to)
rts	logical; specifies whether the downloaded files should be returned as a Raster Time Series object
ncore	numeric; specifies the number of cores to use for parallel downloading of the files
...	additional arguments (Not implemented yet.)

Details

This function assists to download Blended Vegetation Health Indices Product (blended VIIRS (2013-present) and AVHRR (1981-2012), below, referred as Blended-VHP or VHP). These images for are available with a weekly temporal resolution and a spatial resolution of 4 KM. Five products are available that are specified with the following abbreviations:

- 'SMT': Smoothed Brightness Temperature - 'SMN': Smoothed NDVI - 'TCI': Temperature Condition Index - 'VHI': Vegetation Health Index - 'VCI': Vegetation Condition Index

Data arrays are in geographic projection (grid with equal latitude and longitude interval).

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

Examples

```
## Not run:
library(raster)

library(RCurl)


# download Vegetation Health Index for two months
vhi <- VHPdownload(x='VHI',dates=c('2015.01.01','2015.02.28'),rts=TRUE) # output is as rts object

vhi

plot(vhi[[1]])

plot(vhi[2120000]) # plot time series at the specified cell number


# to make sure the dates are appropriately specified, use a Date object:

dates <- as.Date(c('2015.01.01','2016.12.31'),format="
```

```
dates

class(dates)

dates <- as.Date(c('2012-01-01','2012-12-31'),format="

dates

# If your machine has multiple cores, you can use parallel downloading to speed up the downloads
# Vegetation Condition Index for two years
vci <- VHPdownload(x='VCI',dates=dates,rts=TRUE,ncore=4)

vci

plot(vci[[1:2]])



## End(Not run)
```

`write.rts` *Write raster time Series data to a file*

Description

Write an entire (`Raster*TS`) object to a file.

Usage

```
write.rts(x, filename, overwrite=FALSE, ...)
```

Arguments

`x`	a raster time series (`Raster*TS`) object created by `rts`
`filename`	Output filename
`overwrite`	Logical. If `TRUE`, "filename" will be overwritten if it exists
`...`	Additional arguments as for `writeRaster`: `datatype` Character. utput data type (e.g. 'INT2S' or 'FLT4S'). See `dataType`. If no datatype is specified, 'FLT4S' is used. `bandorder`: Character. 'BIL', 'BIP', or 'BSQ'.

Details

This function writes a raster time series object into a directory which named as is specified in the `filename` argument. To write the raster data, `writeRaster` in the package **raster** is used. The function writes the time information into a separate ascii file.

Value

This function is used for writing values to a series of files.

Author(s)

Babak Naimi

<naimi.b@gmail.com>

http://r-gis.net

See Also

read.rts, writeRaster

Examples

```
## Not run:
path <- system.file("external", package="rts") # location of files

lst <- list.files(path=path,pattern='.asc$',full.names=TRUE)
lst # list of raster files


r <- stack(lst) # creating a RasterStack object

d <- c("2000-02-01","2000-03-01","2000-04-01","2000-05-01") # corresponding dates to 4 rasters
d <- as.Date(d) # or d <- as.POSIXct(d)

n <- rts(r,as.Date(d)) # creating a RasterStackTS object

write.rts(n,"nf") # writing n into the working directory

rt <- read.rts("nf") # reading nf from the working directory

rt


## End(Not run)
```

Index