Package: geoTS (via r-universe)

October 18, 2024

Version 0.1.8

Date 2022-11-16

Title Methods for Handling and Analyzing Time Series of Satellite Images

Author Inder Tecuapetla-Gómez [aut, cre]

Maintainer Inder Tecuapetla-Gómez <itecuapetla@conabio.gob.mx>

Description Provides functions and methods for: splitting large raster objects into smaller chunks, transferring images from a binary format into raster layers, transferring raster layers into an 'RData' file, calculating the maximum gap (amount of consecutive missing values) of a numeric vector, and fitting harmonic regression models to periodic time series. The homoscedastic harmonic regression model is based on G. Roerink, M. Menenti and W. Verhoef (2000) <doi:10.1080/014311600209814>.

LazyData yes

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.5.0), raster (>= 2.9.5)

Imports doParallel (>= 1.0.14), ff (>= 2.2-14), foreach (>= 1.4.4), parallel (>= 3.6.1), robustbase (>= 0.95-0), sp (>= 1.2-0)

NeedsCompilation no

RoxygenNote 7.2.0

Repository CRAN

Date/Publication 2022-11-17 18:10:18 UTC

Contents

`geoTS-package` *Methods for Handling and Analyzing Time Series of Satellite Images*

Description

We provide tools for handling time series of satellite images as well as some statistical methods for spatio-temporal analysis

Tools for handling time series of satellite images

`transfer_bin_raster` transfers data from images originally recorded in a binary format to images in any of the formats allowed by the `raster-package`. Similarly, `transfer_raster_RData` extracts the entries of images originally recorded in `tiff` format, virtually stores them in an `array` object and, finally, this array is saved in an `RData` file. `split_replace` allows us to split `Raster*` objects, which can be arguably large, into smaller chunks. These chunks can be saved in any of the formats allowed by `writeRaster`. Often, satellite images come with missing values (or fill values assigned by other computer programs), `split_replace` allows to replace these values by values of users' convenience; see also `reclassify`. `raster_intersect_sp` allows us to obtain data in the intersection of `Raster*` and `SpatialPolygonsDataFrame` objects.

Methods for analyzing time series of satellite images

`haRmonics` allows us to fit harmonic regression models to numeric vectors; the method `hants` is based on *Roerink et al. (2000)* whereas the method `harmR` is based on *Jakubauskas et al. (2001)*. The `wls_harmR` is the weighted least squares method which requires pre-estimation of heteroscedastic variance; `hetervar` allows for heteroscedastic variance estimation for numeric vectors extracted from time series of satellite imagery.

geoTS include the following datasets:

- `master`: `RasterLayer` with a land mask of eastern Yucatan Peninsula, Mexico.
- `MOD13Q1_NDVI_2000129_009`: A spatial subset of NDVI measurements taken over the eastern Yucatan Peninsula, Mexico in 2000.
- `MOD13Q1_NDVI_Mohinora`: `RasterStack` containing 23 spatial subsets of 16-day NDVI images of **Cerro Mohinora** acquired in 2001.
- `shp_mohinora`: `SpatialPolygonsDataFrame` delimiting the smallest Protected Area of Flora and Fauna in Mexico (**Cerro Mohinora**).

Author(s)

Tecuapetla-Gomez, I. <itecuapetla@conabio.gob.mx>

References

Roerink, G.J., Menenti, M., Verhoef, W. (2000). *Reconstructing clodfree NDVI composites using Fourier analysis of time series*, Int. J. Remote Sensing, **21(9)**, 1911–1917.

Jakubauskas, M., Legates, D., Kastens, J. (2001). *Harmonic analysis of time-series AVHRR NDVI data*, Photogrammetric Engineering and Remote Sensing, **67(4)**, 461–470.

The Matlab implementation of HANTS can be found here.

`haRmonics` *Harmonic analysis for time series*

Description

Fits harmonic regression models, that is, computes amplitudes and phase angles in the typical harmonic regression framework. When `method=harmR` the ordinary least squares method is used, when `method=wls_harmR` then, weighted least squares are employed. Based on these estimates a harmonic regression function is fitted. Also fits `hants`, a popular iterative algorithm that computes amplitudes and phase angles in the harmonic regression framework. As part of the iterative algorithm, observations are being excluded from the design matrix of the regression model if the distance between them and the fitted curve exceeds the value of the parameter `fitErrorTol`. `hants` is based on implementations with the same name written in Fortran and Matlab computer languages.

Usage

```
haRmonics(
  y,
  method = c("harmR", "wls_harmR", "hants"),
  sigma = NULL,
  ts = 1:length(y),
  lenPeriod = length(y),
  numFreq,
  HiLo = c("Hi", "Lo"),
  low,
  high,
  fitErrorTol,
  degreeOverDeter,
  delta
)
```

Arguments

y	numeric vector containing time series on which harmonic regression will be fitted. Missing values are not allowed.
method	character specifying algorithm to apply: harmR (default), wls_harmR (heteroscedastic model) or hants.
sigma	numeric vector of length lenPeriod containing variance estimates. Default set NULL.
ts	numeric vector of length(y) with the sampling points for y. Default is $ts[i] = i, i = 1, \ldots,$ length(y).
lenPeriod	numeric giving the length of the base period, reported in samples, e.g. days, dekads, months, years, etc.
numFreq	numeric indicating the total number of frequencies to be used in harmonic regression. For technical reasons, 2*numFreq+1 must be lesser than length(y).
HiLo	character indicating whether high or low outliers must be rejected when method=hants.
low	numeric giving minimum valid value of fitted harmonic regression function when method=hants.
high	numeric giving maximum valid value of fitted harmonic regression function when method=hants.
fitErrorTol	numeric giving maximum allowed distance between observations and fitted curve; if difference between a given observation and its fitted value exceeds fitErrorTol then this observation will not be included in the fitting procedure in the next iteration of the algorithm.
degreeOverDeter	numeric; iteration stops when number of observations equals number of observations for curve fitting plus degreeOverDeter; the latter in turns is by definition length(y) minus min(2 * numFreq+1, length(y)).
delta	numeric (positive) giving a (small) regularization parameter to prevent non-invertible hat matrix (see **Details**), probably caused by high amplitudes.

Details

Methods harmR and wls_harmR do not allow missing values and utilize parameters y, lenPeriod, numFreq and delta only.

Method hants utilizes all the parameters presented above. This method does not allow missing values. Missing values in y must be substituted by values considerably out of observations range.

Value

A list containing:

a.coef	a numeric vector with estimates of cosine coefficients
b.coef	a numeric vector with estimates of sine coefficients
amplitude	a numeric vector with amplitude estimates.
phase	a numeric vector with phase estimates.
fitted	a numeric vector with fitted values via harmonic regression.

Note

`lenBasePeriod` was used until version 0.1.3, this argument has been replaced by `lenPeriod`.

References

Roerink, G.J., Menenti, M., Verhoef, W. (2000). *Reconstructing cloudfree NDVI composites using Fourier analysis of time series*, Int. J. Remote Sensing, **21(9)**, 1911–1917.

Jakubauskas, M., Legates, D., Kastens, J. (2001). *Harmonic analysis of time-series AVHRR NDVI data*, Photogrammetric Engineering and Remote Sensing, **67(4)**, 461–470.

The Matlab implementation of HANTS can be found here.

Examples

```
y <- c(5, 2, 5, 10, 12, 18, 20, 23, 27, 30, 40, 60, 66,
70, 90, 120, 160, 190, 105, 210, 104, 200, 90, 170,
50, 120, 80, 60, 50, 40, 30, 28, 24, 20, 15, 10)
# ------------------------------------------------------------------------
fit_harmR <- haRmonics(y = y, numFreq = 3, delta = 0.1)
fitLow_hants <- haRmonics(y = y, method = "hants", numFreq = 3, HiLo = "Lo",
                          low = 0, high = 255, fitErrorTol = 5, degreeOverDeter = 1,
                          delta = 0.1)
fitHigh_hants <- haRmonics(y = y, method = "hants", numFreq = 3, HiLo = "Hi",
                           low = 0, high = 255, fitErrorTol = 5, degreeOverDeter = 1,
                           delta = 0.1)
plot(y, pch = 16, main = "haRmonics fitting")
lines(fit_harmR$fitted ,lty = 4, col = "green")
lines(fitLow_hants$fitted, lty = 4, col = "red")
lines(fitHigh_hants$fitted, lty = 2, col = "blue")
# ------------------------------------------------------------------------
# Substituting missing value by a number outside observations range
# ------------------------------------------------------------------------
y1 <- y
y1[20] <- -10

fitLow_hants_missing <- haRmonics(y = y1, method = "hants", numFreq = 3, HiLo = "Lo",
                                  low = 0, high = 255, fitErrorTol = 5, degreeOverDeter = 1,
                                    delta = 0.1)
fitHigh_hants_missing <- haRmonics(y = y1, method = "hants", numFreq = 3, HiLo = "Hi",
                                   low = 0, high = 255, fitErrorTol = 5, degreeOverDeter = 1,
                                     delta = 0.1)
fit_harmR_missing <- haRmonics(y = y1, numFreq = 3, delta = 0.1)

plot(y1, pch = 16, main = "haRmonics fitting (missing values)", ylim = c(-1,210))
lines(fitLow_hants_missing$fitted, lty = 4, col = "red")
lines(fitHigh_hants_missing$fitted, lty = 2, col = "blue")
lines(fit_harmR_missing$fitted, lty = 4, col = "green")
```

`hetervar` *Heteroscedastic variance estimation for remotely-sensed data*

Description

Variance of some remotely-sensed Earth data is time-varying. Utilizing the observations per period (season, year), this function allows for estimation of variability in data either as numeric vector or matricial form

Usage

```
hetervar(
  x,
  m = NULL,
  lenPeriod = 23,
  method = c("standard", "robust-mad", "robust-Qn")
)
```

Arguments

`x`	numeric vector
`m`	matrix with `nrow` equal to the number of periods (seasons or years) analyzed and `ncol` equal to the number of observations per period
`lenPeriod`	numeric giving the number of observations per period. Default, 23.
`method`	character specifying whether `standard` variance, the median absolute deviation (`robust-mad`) or the more efficient robust variance estimator (`robust-Qn`) should be used

Details

Designed for data extracted from time series of satellite imagery. Then, it is expected that `length(x)` be a multiple of `lenPeriod`. When `m` is provided, `ncol(m)` must be equal to `lenPeriod`. Default of `lenPeriod` corresponds to the temporal resolution of some MODIS products.

Method `standard` invokes `sd` whereas `robust-mad` uses the median absolute deviation of `mad` and `robust-Qn` utilizes the robust scale estimator implemented in `Qn`.

This function does not allow missing values.

Value

A numeric vector of length `lenPeriod`

See Also

`sd`, `mad`, `Qn`

master *Land Mask of eastern Yucatan Peninsula*

Description

A RasterLayer with a spatial subset covering eastern Yucatan Peninsula of Mexico. A land mask is a binary layer where 1=Land, 0=Water.

master.tif

A RasterLayer object with 500 rows, 600 columns. Each cell has a resolution of 250m.

matrixToRaster *Creates a RasterLayer object from a matrix*

Description

Transforms a matrix into a RasterLayer object.

Usage

```
matrixToRaster(matrix, raster = NULL, projection = NULL)
```

Arguments

matrix	a matrix object. See **Details**.
raster	a RasterLayer object whose extent and projection are used to create a raster from matrix.
projection	a character vector providing a coordinate reference system. Required when ncol(matrix)=3.

Details

When ncol(matrix)=3, this function assumes that the first two columns of argument matrix provide coordinates to create a RasterLayer, hence argument projection must be provided. When argument matrix has only 2 columns, then the argument raster must be provided because its coordinates and projection will be used to rasterize matrix.

Value

A RasterLayer

Note

In previous versions, raster argument was written in capital letters.

See Also

Raster-class

---

maxLagMissVal *Get maximum lag of missing values*

Description

This function computes the maximum amount of consecutive missing values in a vector. This quantity is also known as maximum lag, run, or record, and can be used as a rough estimate of the quality of a dataset.

Usage

```
maxLagMissVal(x, type = c("NA", "numeric"), value)
```

Arguments

x	numeric vector.
type	character specifying the type of missing value to consider. Default is type="NA"; when type="numeric", value must be provided.
value	numeric giving a figure to be used to fill missing values; often as part of a pre-processing, missing values in a dataset (vector, time series, etc.) are fill in with pre-established values.

Value

A list containing:

maxLag	numeric giving the maximum lag of missing values in x
x	numeric vector with the original data
value	a numeric when type=numeric, NA otherwise

See Also

rle

Examples

```
v <- c(NA, 0.12, 0.58, 0.75, NA, NA, NA, 0.46, 0.97, 0.39,
       NA, 0.13, 0.46, 0.95, 0.30, 0.98, 0.23, 0.98,
       0.68, NA, NA, NA, NA, NA, 0.11, 0.10, 0.79, 0.46, 0.27,
       0.44, 0.93, 0.20, 0.44, 0.66, 0.11, 0.88)
maxLagMissVal(x=v, type="NA")

w <- c(23,3,14,3,8,3,3,3,3,3,3,10,14,15,3,10,3,3,6)
maxLagMissVal(x = w, type = "numeric", value = 3)
```

MOD13Q1_NDVI_2000129_009 *MOD13Q1 NDVI binary file*

Description

Spatial subset of a MOD13Q1 NDVI layer, in binary format, covering eastern Yucatan Peninsula, Mexico. NDVI stands for Normalized Difference Vegetation Index; NDVI = (NIR-RED)/(NIR+RED) where NIR and RED are the Near Infrared and Red bands of the MODIS product, respectively. More information about the MODIS mission can be found here.

MOD13Q1_NDVI_2000129_009.bin

This image contains NDVI measurements for the 129-th Julian day of 2000; the subscript *_009* signifies that this was the 9th NDVI observation of 2000.

MOD13Q1_NDVI_Mohinora *16-day MOD13Q1 NDVI RasterStack*

Description

A RasterStack containing 23 layers of NDVI for 2001. This RasterStack is a spatial subset covering the Protected Area of Flora and Fauna **Cerro Mohinora** at Chihuahua, Mexico.

MOD13Q1_NDVI_Mohinora.tif

A RasterStack object with 59 rows, 93 columns, 5487 cells and 23 layers.

raster_intersect_sp *Intersects raster and sp objects*

Description

Straightforward application of crop and mask to extract the data in the intersection of Raster* and SpatialPolygonsDataframe objects.

Usage

```
raster_intersect_sp(
  x,
  y,
  features,
  save = FALSE,
  dirToSave,
  baseName = "x_intersect_y",
  format = "GTiff"
)
```

Arguments

x	Raster* object
y	SpatialPolygonsDataFrame object
features	integer vector. Should some specifying features (polygons) of y be intersected with x? When not provided, whole y is intersected with x.
save	logical. Should the output be saved? Default, FALSE.
dirToSave	character specifying where to save output. Required when save=TRUE.
baseName	character. What should be the base name of output file? Default, x_intersect_y.
format	character specifying output file format. See writeFormats for all supported formats. Default, "GTiff".

Details

When save=TRUE, writeRaster is used with argument datatype=dataType(subset(x,1)).

Value

An object of class identical to that of argument x

Examples

```
raster_path = system.file("extdata", "MOD13Q1_NDVI_Mohinora.tif", package = "geoTS")
rasterSTACK <- stack(raster_path)
dir.create(path=paste0(system.file("extdata", package="geoTS"), "/output_raster_inter_sp"),
          showWarnings=FALSE)
dirToSave = paste0(system.file("extdata", package="geoTS"), "/output_raster_inter_sp")

raster_intersect_sp(x=rasterSTACK, y=shp_mohinora, save=TRUE, dirToSave=dirToSave,
                    baseName="mohinora_NDVI_2001")
```

---

shp_mohinora *SpatialPolygonsDataFrame for Cerro Mohinora*

Description

A RData file containing a SpatialPolygonsDataFrame object delimiting **Cerro Mohinora** at Chihuhua, the smallest Protected Area of Flora and Fauna in Mexico.

Usage

```
data(shp_mohinora)
```

Format

An object of class SpatialPolygonsDataFrame.

split_replace *Splits a Raster* object into smaller chunks and allows to replace cell values*

Description

This function will split a Raster* object into smaller chunks. The size of these chunks (number of cells) is controlled by `partPerSide`, `h` or `v`. Additionally, it allows to replace cell values (`valToReplace`) within Raster* object by another value of user's choice (`replacedBy`). When `save = TRUE`, the resulting `cellsToProcess` Raster* objects are saved in directory `outputPath`.

Usage

```
split_replace(
  raster,
  partPerSide,
  h,
  v,
  outputPath,
  name,
  save = TRUE,
  replace = FALSE,
  valToReplace,
  replacedBy,
  dataType,
  format = "GTiff",
  parallelProcessing = FALSE,
  numCores = 20,
  cellsToProcess,
  ...
)
```

Arguments

`raster`	Raster* object.
`partPerSide`	integer indicating number of cells in which `raster` will be split in each direction (horizontally and vertically). Use when `nrow(raster)` and `ncol(raster)` are multiples of `partPerSide`.
`h`	integer indicating number of horizontal cells in which `raster` will be split.
`v`	integer indicating number of vertical cells in which `raster` will be split.
`outputPath`	character with full path name where the resulting Raster* objects will be saved.
`name`	character with the name to assign to final products.
`save`	logical, should the output be saved, default is `TRUE`.
`replace`	logical, default `FALSE`, when `TRUE`, `valToReplace` and `replacedBy` must by specified.

valToReplace	indicates a value to be replaced across raster cells.
replacedBy	indicates the value by which valToReplace is replaced.
dataType	character, output data type. See dataType.
format	character, output file type, default "GTiff". See writeFormats.
parallelProcessing	logical, default FALSE, when TRUE raster splitting is done in parallel. See **Details**.
numCores	numeric indicating the number of cores used in parallel processing.
cellsToProcess	numeric vector indicating which smaller cells should be processed/saved. See **Details**.
...	additional arguments used by writeRaster.

Details

Before processing any of the cellsToProcess the temporary raster directory is re-directed. Basically, prior to process the i-th cell, at outputPath a new subdirectory is created, which, in turn, is erased automatically once the i-th cell has been processed. As a result of several tests we found that this measure avoids memory overflow.

When partPerSide is used, cellsToProcess = 1:(partPerSide^2). When h and v are used, cellsToProcess = 1:(ncells(raster)/(h*v)). Since the code assumes that nrow(raster) and ncol(raster) are multiples of partPerSide or h and v, respectively, the user must be careful when selecting these parameters.

For parallelProcessing the backend doParallel is employed.

Value

At outputPath the user will find length(cellsToProcess) Raster* files

See Also

writeRaster, aggregate, rasterOptions

---

transfer_bin_raster *Transfer values from a binary image file to a raster file*

Description

Get the values of a binary file (in integer format) and transfer them to a raster file. All formats considered in writeRaster are allowed.

Usage

```
transfer_bin_raster(
  inputPath,
  outputPath,
  master,
  what = integer(),
  signed = TRUE,
  endian = "little",
  size = 2,
  format = "GTiff",
  dataType = "INT2S",
  overwrite = TRUE
)
```

Arguments

inputPath	character with full path name of input file(s).
outputPath	character with full path name (where the raster files will be saved).
master	character with full path name of a raster file; extent and projection of this file are applied to this function output.
what	See readBin. Default integer().
signed	See readBin. Default TRUE.
endian	See readBin. Default "little".
size	integer, number of bytes per element in the byte stream, default 2. See readBin.
format	character, output file type. See writeFormats.
dataType	character, output data type. See dataType.
overwrite	logical, default TRUE, should the resulting raster be overwritten.

Value

At the designated path (outputPath) the user will find TIF file(s).

Examples

```
inputPath = system.file("extdata", package = "geoTS")
masterFile = system.file("extdata", "master.tif", package = "geoTS")
transfer_bin_raster(inputPath = inputPath, outputPath = inputPath,
                    master = masterFile, what = integer(),
                    signed = TRUE, endian = "little", size = 2,
                    format = "GTiff", dataType = "INT2S", overwrite = TRUE)
```

`transfer_raster_RData` *Transfer values from a Raster* object to an RData file*

Description

Get the values of a Raster*, storage them into an `array` and finally save the array in an `RData` which allows for compatibility with multiple R functions as well as great portability.

Usage

```
transfer_raster_RData(
  inputFile,
  outputPath,
  transferOneFile = TRUE,
  vmode = c("integer", "single", "double")
)
```

Arguments

`inputFile`	character with full path name of input file.
`outputPath`	character with full path name (where the `RData` file will be saved). No need to provide extension `.RData`.
`transferOneFile`	logical, default `TRUE` indicates that one file will be transferred. `FALSE` indicates that more than one file will be transferred. See **Details**.
`vmode`	a character specifying the type of virtual storage mode `vmode` needed. Only `integer`, `single` and `double` are allowed.

Details

Prior to embark the user in a transfer that may not be successful due to the lack of RAM, this function provides an estimate of the amount of bytes to be used in the transfer process. The estimate is obtained by multiplying the number of rows by the number of columns by the number of layers of the `Raster*` object to transfer by the amount of bites used by `vmode` (32-bit float for `integer` or `single` and 64-bit float for `double`). A question is displayed in the console requesting whether the process should continue. Should the user decide not to continue with the importation `transfer_raster_RData` returns the message `"Did not transfer anything"`.

When `transferOneFile=FALSE`, it is assumed that the system has enough RAM to support full files transfer -no question is asked in the console. This option is useful when this function is used within a `for` loop.

Value

At the designated path (`outputPath`) the user will find an `RData` file.

See Also

vmode

Examples

```
inputFile = system.file("extdata", "master.tif", package = "geoTS")
outputPath = paste0(system.file("extdata", package = "geoTS"), "/master")
transfer_raster_RData(inputFile = inputFile, outputPath = outputPath,
                      vmode = "single")
```

Index