Package 'igapfill'

July 22, 2025

Version 0.0.41

Date 2025-05-09

Title Interface for Gap-Filling Earth Observation Datasets

Author Inder Tecuapetla-Gómez [aut, cre]

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

Description Provides functions and a user-friendly console-based interface for the efficient use of the main function of the R package 'gapfill' to fill missing values of satellite images subsets. In addition to the R package documentation, the 'gapfill' methods are introduced in Gerber et al. (2018) <doi:10.1109/TGRS.2017.2785240>.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 2.15.3), raster (>= 3.5-15), terra (>= 1.5-21), gapfill(>= 0.9.6-1)

Imports gtools (>= 3.9.4), numbers (>= 0.8-5), doParallel (>= 1.0.17), foreach (>= 1.5.2), geoTS (>= 0.1.8), iterators (>= 1.0.14), itertools (>= 0.1-3), utils (>= 4.5.0)

Suggests knitr, rmarkdown, heatmaply, htmltools

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.3.2

Repository CRAN

Date/Publication 2025-05-14 12:00:06 UTC

Contents

`igapfill-package` *Interface for filling missing values of Earth Observation Datasets*

Description

By making extensive use of parallel computing, the functions of this package facilitate the application of the spatio-temporal gap-filling method `Gapfill` to time series of satellite images (TSSI).

Details

Only *GeoTiff* files are allowed. In passing, some of the functions of this package can be construed as independent builders of arguments used by `Gapfill` functions.

Datasets

Spatial subsets of the MOD13Q1 v061 NDVI product, in *.tif* format, covering Cerro de Garnica National Park (https://simec.conanp.gob.mx/ficha.php?anp=66&reg=11) located at Michoacan, Mexico. These subsets were collected from February 16, 2000 to December 16, 2024. The spatial and temporal resolutions of these images are 250m and 16 days, respectively.

`garnica_250m_16_days_NDVI.tif`	572 layers of NDVI
`garnica_250m_16_days_pixel_reliability.tif`	Pixel reliability layers corresponding to garnica_250m_16_days_NDVI.tif

Quality assessment summary

The following functions allow to compute the amount of missing values in a TSSI and to determine a sub-set of images to which apply the workflow of this package.

`mvSieve`	Computes amount of missing values in a TSSI
`minmaxBlock`	Determines sub-set of images with minimal (or maximal) missing values

Workflow

The following functions allow to define the required directory/folders structure employed by this package. Some of these functions are also useful for better data handling.

create_dirs	Sets up directory tree
dimsReport	Summary of dimensions of images to process
sort_split	Split large TSSI into smaller spatio-temporal chunks
waysToSplit	Briefing of ways to divide rows and columns of images

Interface

These are the functions of this package that allow for filling missing values of spatio-temporal subsets of satellite images using Gapfill.

applyGapfill	Parallel computing-based application of Gapfill
parallel_mosaic	Parallel rasterization and mosaicking (when required) of output of applyGapfill
igapfill	Console-based wrap-up of applyGapfill and parallel_mosaic

Miscellaneous

These functions can be used to obtain some of the arguments required by applyGapfill. In addition to this, these functions can also be employed to define some arguments of Gapfill.

get_3Darray	Assambles 3D array
get_4Darray	Assambles 4D array
get_LAT	Gets latitude information of RasterStack
get_LON	Gets longitude information of RasterStack

Author(s)

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

References

Gerber, F., de Jong, R., Schaepman, M.E., Schaepman-Strub, G., Furrer, R. (2018). *Predicting missing values in spatio-temporal remote sensing data*, IEEE Transactions on Geoscience and Remote Sensing, 1–13.

applyGapfill *Gapfilling for Earth Observation datasets*

Description

Making use of parallel computing this function allows for the application of Gapfill to a set of satellite images. It is mandatory that these images have been configured for processing according to the workflow of this package previously.

Usage

```
applyGapfill(
  inputDir,
  outputDir,
  progressDir,
  lat,
  lon,
  days,
  years,
  numCores = 6,
  scale = 1e-04,
  clipRange = c(-1, 1),
  addArgToReport = TRUE
)
```

Arguments

inputDir	character. Full path name of directory containing files configured to be processed with Gapfill. See **Note**.
outputDir	character. Full path name of directory where output will be saved. See **Note**.
progressDir	character. Full path name of directory where a file reporting on the status of the process will be saved. See **Note**.
lat	character vector. See get_LAT.
lon	character vector. See get_LON.
days	numeric vector indicating what days are being considered. See get_4Darray.
years	integer vector indicating what years are being considered. See get_4Darray.
numCores	numeric. How many cores should be employed in parallel computing?
scale	numeric. See Gapfill. Default is 1e-4. See **Note**.
clipRange	numeric vector of length 2. See Gapfill. Default is c(-1,1). See **Note**.
addArgToReport	logical. Should a copy of the input arguments be passed onto the progress report file? Default is TRUE.

Details

Should the users have not yet created it, this function allows to create the directory/folders structure employed by this package workflow. When users opt for creating such a directory structure, additional arguments are required at the console. When users acknowledge the existence of such directory structure or when they decide to create such structure independently, this function returns the message *"Try again later passing the required parameters"*.

Value

Should the user decide to employ this function, at the output directory (`outputDir`) there will be `.RData` files containing the output of parallel-based calls to `Gapfill`. There will be as many `.RData` files as files are in any of the sub-directories indicated by `inputDir`.

Note

Within the workflow of this package, `inputDir`, `outputDir` and `progressDir` must be equal to the sub-directories */splits*, */output*, and */progressReports*, created by `create_dirs`, respectively. Many satellite products come with a scale factor of `1e4`, using `scale=1e-4` maps pixel values to the interval `(-1,1)` which is the default for argument `clipRange`.

See Also

`create_dirs`, `foreach`, `Gapfill`, `makeCluster`, `registerDoParallel`

`create_dirs` *Creates a set of directories which are an essential part of the general workflow of this package*

Description

Using the directory provided by `path` as root this function creates the sub-directory */gapfill* with sub-folders */gapfill/filled*, */gapfill/master*, */gapfill/output*, */gapfill/progressReports* and */gapfill/splits*.

Usage

```
create_dirs(path, startYear, endYear)
```

Arguments

`path`	character, full path indicating the directory containing the images to process with `Gapfill`.
`startYear`	numeric, indicates the starting time-point, on the annual scale, of a time series of satellite images to process.
`endYear`	numeric, indicates the ending time-point, on the annual scale, of a time series of satellite images to process.

Value

At the location indicated by path, the abovementioned directories will be created.

`dimsReport` *Dimensions and splitting characteristics of images to process*

Description

This function returns a few messages at the console. These messages report on *(i)* the dimensions of the images located at path and *(ii)* the ways in which these dimensions can be split.

Usage

```
dimsReport(path, mes)
```

Arguments

path	character, full path indicating the directory containing the images to process with Gapfill.
mes	character, when not provided the default message is *"The images located at 'path' have:"*.

Value

At the console there will be a series of messages, no further actions will be taken.

See Also

divisors, waysToSplit

`garnica` *Time series images of Cerro de Garnica, Mexico (2000-2024)*

Description

Spatial subsets of the MOD13Q1 v061 NDVI product, in *.tif* format, covering Cerro de Garnica National Park (https://simec.conanp.gob.mx/ficha.php?anp=66&reg=11) located at Michoacan, Mexico. These subsets were collected from February 16, 2000 to December 16, 2024. The spatial and temporal resolutions of these images are 250m and 16 days, respectively.

Details

Due to the amount and characteristics of these images, they are stored in two files which we describe in what follows:

garnica_250m_16_days_NDVI.tif

This file contains 572 layers of Normalized Difference Vegetation Index (NDVI). By definition, NDVI = (NIR-RED)/(NIR+RED) where NIR and RED are the Near Infrared and Red spectral bands, respectively. Although by definition, the NDVI is a value belonging to the (-1, 1) interval, NASA distributes this product in integer (`INT2S`) data type; the scale parameter is 1e4.

More information about this MODIS product can be found here.

garnica_250m_16_days_pixel_reliability.tif

This file contains 572 pixel reliability layers. That is, in any of these layers, the value of any pixel is either -1 (Fill/No data), 0 (Good quality), 1 (Marginal data), 2 (Snow/Ice) or 3 (Cloudy).

Note

About the naming convention

Any layer in these files has the following naming convention:

MOD13Q1.AYYYYDDD.h08v07.061.YYYYDDDHHMMSS.250m_16_days_PRODUCT.tif

where:

- MOD13Q1 is the product short name
- AYYYYDDD is the Julian date of acquisition
- h08v07 is the tile identifier
- 061 is the product version
- YYYYDDDHHMMSS is the Julian date of production
- 250m is the spatial resolution
- 16_days is the temporal resolution
- PRODUCT is either NDVI or pixel_reliability

`get_3Darray` *Assembles a 3D-array*

Description

This function returns a 3D-array which is an auxiliary object when invoking `applyGapfill`.

Usage

```
get_3Darray(path)
```

Arguments

`path`	character with full path name of a directory containing files that can be read as RasterStacks

Value

An array with three dimensions

Note

This function may be useful when employing `Gapfill` independently of the current package.

See Also

`array`, `Gapfill`

`get_4Darray` *Assembles a 4D-array*

Description

This function returns a 4D-array which is an auxiliary object when invoking `applyGapfill`.

Usage

```
get_4Darray(listPath, i, lon, lat, days, years)
```

Arguments

`listPath`	list, containing lists with names of files to be assembled as a 4D array.
`i`	numeric indicating i-th entry of `listPath` to be processed.
`lon`	character vector whose entries indicate longitude coordinates.
`lat`	character vector whose entries indicate latitude coordinates.
`days`	numeric vector indicating what days are being considered. Length of this object must be equal to length of `listPath`.
`years`	integer vector indicating what years are being considered. Length of this object must be equal to length of `listPath`.

Details

Each entry of `listPath` must contain files associated with images registered during the same year. `lon` and `lat` can be obtained with the functions `get_LON` and `get_LAT`, respectively. `days` must be provided by the user, otherwise it will be set to `1:length(years)`.

Value

An array of 4 dimensions: longitude, latitude, days and years

Note

This function may be useful when employing `Gapfill` independently of the current package.

See Also

create_dirs, get_3Darray

---

get_LAT *Gets RasterStack's latitude information*

Description

This function constructs the input for one of the four dimensions required by Gapfill, namely, *latitude*.

Usage

```
get_LAT(stack)
```

Arguments

stack	RasterStack

Value

Character vector of length equal to `nrow(stack)`.

Note

This function may be useful when employing Gapfill independently of the current package.

See Also

get_3Darray, get_LON

---

get_LON *Gets RasterStack's longitude information*

Description

This function constructs the input for one of the four dimensions required by Gapfill, namely, *longitude*.

Usage

```
get_LON(stack)
```

Arguments

stack	RasterStack

Value

Character vector of length equal to `ncol(stack)`.

Note

This function may be useful when employing `Gapfill` independently of the current package.

See Also

`get_3Darray`, `get_LAT`

`igapfill` *Console-based interface for filling missing values of Earth Observation datasets*

Description

Command-line user-friendly application that allows the application of `Gapfill` to a set of satellite images.

Usage

```
igapfill(saveArguments = TRUE)
```

Arguments

`saveArguments`	logical. Should the arguments defined during the execution of this function be added to a progress report file? Default is `TRUE`.

Details

This function is a wrap-up of `create_dirs`, `sort_split` and `applyGapfill` allowing users to provide some of the arguments employed by these functions.

Value

At the specified location there will be `.RData` files containing the output of parallel-based calls to `Gapfill`.

See Also

`sort_split`, `detectCores`, `applyGapfill`

`minmaxBlock` *Minimal or maximal* 2×2 *(non-zero) matrix of missing values*

Description

Finds 2×2 (non-zero) block with the minimum or maximum amount of missing values within a general missing values matrix derived from satellite images.

Usage

```
minmaxBlock(sieve, type = c("min", "max"), rank)
```

Arguments

`sieve`	matrix
`type`	character. Default is `"min"`.
`rank`	numeric. See **Details**.

Details

In what follows we describe the case `type="min"`. This function searches for the minimal 2×2 (non-zero) sub-matrix within a *general* `sieve` matrix. *blockMissingness* is defined as $\log(cumsum(a_{i,j}))$ for $1 \leq i, j \leq 2$. The minimal block is defined as that block with the minimum blockMissingness. The `cumsum` function is preferred rather than other quantities, such as `cumprod` or `det`, because `sieve` could have a large amount of zeros.

In the first stage of the search, a vector with the sorted non-zero values of `sieve` is calculated. Consider the i-th entry of this sorted vector. This value corresponds to some (maybe more than once) cell within `sieve`. Notice that, with the exception of the edges of the sieve, this cell belongs to four 2×2 matrices. The blockMissingness of each of these 4 matrices is calculated. The matrix with the smallest blockMissingness is called a *localMinBlock*.

The procedure just described is applied to each of the `rank` entries of the sorted vector; the *globalMinBlock* is that localMinBlock with the smallest blockMissingness.

The case `type="max"` is analogous to the one above but the searches is now for the cell with the largest blockMissingness and the search now runs (in descending order) over the last `rank`-th entries of the sorted vector.

The argument `rank` is defined as follows. Let `sorted_vector` be a numeric vector with the non-zero, ordered (in ascending order) values of `sieve`. When `type="min"`, `rank` defines the first `rank`-th values of `sorted_vector`. When `type="max"`, `rank` defines the last `rank`-th values of `sorted_vector`.

Value

A list containing:

`rows`	a numeric vector given the rows of `sieve` where the minimal 2×2 block is found.

cols	a numeric vector given the cols of sieve where the minimal 2×2 block is found.
block	a 2×2 sub-matrix of sieve, the actual minimal block.
blockMissingness	a numeric with the blockMissingness of the minimal block.

See Also

mvSieve

---

mvSieve *Sieve of amount of missing values in a time series of satellite images*

Description

This function computes the number of pixels with missing values (no data) in each element of a time series of satellite images. For practical purposes, this function assumes that the images have been stored in a set of different sub-directories; each sub-directory can represent a period/season/year.

Usage

```
mvSieve(dirs, filesPerDir, startPeriod, endPeriod, colNames = month.name)
```

Arguments

dirs	character vector given sub-directory names from which images will be read.
filesPerDir	numeric indicating how many images are stored in each directory.
startPeriod	numeric indicating in which period the time series first image was recorded.
endPeriod	numeric indicating in which period the time series last image was recorded.
colNames	character vector. Default month.name which assumes that filesPerDir=12.

Value

An $n \times m$ matrix where n is equal to length(dirs) and m is equal to filesPerDir. By default, the row names of this matrix are equal to startPeriod:endPeriod and the column names are equal to colNames.

See Also

minmaxBlock

parallel_mosaic *Mosaic from gapfilled objects*

Description

Rasterizes the output of `applyGapfill` and the resulting raster objects are glued together in the manner of a mosaic.

Usage

```
parallel_mosaic(
  inputDirImages,
  inputDirRData,
  inputDirMaster,
  outputDir,
  progressReportDir,
  numCores,
  scaleFactor = 10000,
  dataType = "INT4S"
)
```

Arguments

`inputDirImages`	character. Full path name of directory containing files to be gap-filled.
`inputDirRData`	character. Full path name of directory containing `RData` files obtained as the result of using `Gapfill`. See **Note**.
`inputDirMaster`	character. Full path name of directory containing splits of a *master* file. See **Details**.
`outputDir`	character. Full path name of directory where output will be saved. See **Note**.
`progressReportDir`	character. Full path name of directory where a file reporting on the See **Note**.
`numCores`	numeric. How many cores should be employed in parallel computing?
`scaleFactor`	integer. Default is `1e4`. See **Note**.
`dataType`	character. See `writeRaster` for further details. Default is `INT4S`. See **Note**.

Details

This function may be useful when employing `Gapfill` independently of the current package.

The term *master* refers to a raster with no missing values and whose coordinate reference system is used to rasterize objects such as matrices.

Value

At `outputDir` the user will find `n` *Gtiff* files, where `n` is equal to the number of files in `inputDirImages`.

Note

Within the workflow of this package, `inputDirRData`, `inputDirMaster`, `outputDir` and `progressReportDir` must be equal to the sub-directories */output*, */master*, */filled*, and */progressReports*, respectively. These folders can be created by `create_dirs`. Many satellite products come with a scale factor of `1e4` and are distributed in formats equivalent to `INT4S`. After `applyGapfill` the objects to rasterize/mosaic must be scaled back, therefore the default values for arguments `scaleFactor` and `dataType`.

See Also

`mosaic`, `applyGapfill`.

`sort_split` *Console-based application to sort and split spatio-temporal chunks of images*

Description

An application of `split_replace` to split/divide/configure images in parts to which a subsequent call of `applyGapfill` can be easily handled by regular computer systems.

Usage

```
sort_split(path, startYear, endYear, nrow_split, ncol_split)
```

Arguments

`path`	character with full path name to a directory containing a set of files to be split.
`startYear`	numeric indicating the starting time-point, on the annual scale, of a time series of satellite images.
`endYear`	numeric indicating the ending time-point, on the annual scale, of a time series of satellite images.
`nrow_split`	numeric, in how many equal parts the images' rows must be split? See **Details.**
`ncol_split`	numeric, in how many equal parts the images' cols must be split? See **Details.**

Details

This function asks the user a series of inputs on-the-fly. Should the user allow it, these inputs will be used as arguments in a subsequent call to `split_replace`.

`create_dirs` defines a specific directory structure used by `sort_split`, hence, it is highly recommended to use `create_dirs` in advance. Also, it is highly recommended to use `sort_split` before using `applyGapfill`.

Value

When the user decides not to proceed with this function, a `NULL` is returned at the console. Otherwise, the resulting splits (.tif files) will be saved at the sub-directories defined by `paste0(path, "/gapfill/splits")` and a corresponding final _invisible_ message is displayed at the console.

Note

The "sort" part of the function means that in case that the file list created from `path` is not originally ordered, then `sort_split` will sort it out internally.

See Also

`waysToSplit`, `dimsReport`, `split_replace`.

`waysToSplit` *Brief report on ways to split a Raster* object*

Description

Based on the passed arguments, this function returns three messages at the console. First, the number of *splits* for the Raster* object. Second, the number of rows and third, the number of columns of each split.

Usage

```
waysToSplit(h, v, raster)
```

Arguments

`h`	numeric, parts in which number of columns of `raster` will be split
`v`	numeric, parts in which the number of rows of `raster` will be split
`raster`	Raster* object to be split

Details

For an abuse of language, here we use the term *split* to signify *cell* or *crop*, which in the context of handling geo-referenced data structures are more common terms.

Value

At the console, there will be a summary indicating the number of *splits* for the Raster* object as well as the number of rows and columns of each *split*.

Index