

Package ‘gapfill’

July 22, 2025

Type Package

Title Fill Missing Values in Satellite Data

Version 0.9.6-1

Date 2021-02-11

Maintainer Florian Gerber <flora.fauna.gerber@gmail.com>

Description Tools to fill missing values in satellite data and to develop new gap-fill algorithms. The methods are tailored to data (images) observed at equally-spaced points in time. The package is illustrated with MODIS NDVI data.

License GPL (>= 2)

URL <https://github.com/florafauna/gapfill>

BugReports <https://github.com/florafauna/gapfill/issues>

Depends R (>= 3.1), ggplot2 (>= 2.2.1)

Imports fields, foreach (>= 1.4), Rcpp (>= 0.12.1), quantreg (>= 5.0)

Suggests roxygen2, spam, testthat, abind

Enhances raster, doParallel, doMPI

LinkingTo Rcpp

LazyData true

RoxyenNote 7.1.1

NeedsCompilation yes

Author Florian Gerber [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-8545-5263>>)

Repository CRAN

Date/Publication 2021-02-12 10:10:05 UTC

Contents

gapfill-package	2
Array2Matrix	3
ArrayAround	4
EstimateQuantile	5
Extend	6
Gapfill	8
Image	12
Index	14
ndvi	15
Score	16
Subset-Predict	17
Validate	20
Index	22

gapfill-package	<i>Overview</i>
-----------------	-----------------

Description

The package provides tools to fill-in missing values in satellite data. It can be used to gap-fill, e.g., MODIS NDVI data and is helpful when developing new gap-fill algorithms. The methods are tailored to data (images) observed at equally-spaced points in time. This is typically the case for MODIS land surface products and AVHRR NDVI data, among others.

The predictions of the missing values are based on a subset-predict procedure, i.e., each missing value is predicted separately by (1) selecting subsets of the data that are in a neighborhood around the missing point and (2) predicting the missing value based on the subset.

The main function of the package is [Gapfill](#).

Features

- Gap-filling can be executed in parallel.
- Users may define new [Subset](#) and [Predict](#) functions and run alternative prediction algorithms with little effort. See [Extend](#) for more information and examples.
- Visualization of space-time data are simplified through the ggplot2-based function [Image](#).

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

References

F. Gerber, R. de Jong, M. E. Schaepman, G. Schaepman-Strub, and R. Furrer (2018) in IEEE Transactions on Geoscience and Remote Sensing, pp. 1-13, doi: [10.1109/TGRS.2017.2785240](#).

See Also

[Gapfill](#), [Subset-Predict](#), [Extend](#), [Image](#).

Array2Matrix

Convert an Array with 4 Dimensions into a Matrix

Description

Converts the array, `a`, with 4 dimensions, `c(d1, d2, d3, d4)`, into a matrix with `d1*d2` rows and `d3*d4` columns.

Usage

```
Array2Matrix(a)
```

Arguments

`a` Array with 4 dimensions.

Value

A matrix. If `a` has the attribute `mp`, the transformed attribute is returned as well. See [ArrayAround](#) for more information about `mp`.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

See Also

[Index](#), [ArrayAround](#).

Examples

```
a <- array(data = 1:16, dim = c(2, 2, 2, 2))
Array2Matrix(a = a)
attr(a, "mp") <- c(1, 2, 2, 1)
Array2Matrix(a = a)

Array2Matrix(ArrayAround(data = a, mp = c(1, 1, 1, 1),
                          size = c(1, 1, 2, 2)))
```

ArrayAround

Subset an Array with 4 dimensions

Description

Given an array data with 4 dimensions, a subset around the element with coordinates mp ("missing position") is extracted. The size of the subset in all four directions from mp is specified by size. ArrayAroundRandom returns a subset around a random location in data.

Usage

```
ArrayAround(data, mp, size)
```

```
ArrayAroundRandom(
  data,
  size = c(0L, 0L, 0L, 0L),
  target = c("all", "missing", "observed"),
  verbose = TRUE
)
```

Arguments

data	Array with 4 dimensions.
mp	Integer vector of length 4 indexing an element in data.
size	Integer vector of length 4, that provides the size of the subset in all four dimensions (around mp).
target	One of c("all", "missing", "observed"). Indicates from which subset of data a random location is sampled.
verbose	Logical vector of length 1. If TRUE, messages are printed.

Value

Array with 4 dimensions corresponding to the specified subset. The attribute mp of the returned array is an integer vector of length 4 giving mp relative to the returned array.

Note

When size = c(0, 0, 0, 0), the returned subset consists of one value (the value of data indexed with mp.)

When mp is near the boundaries of data, the returned subset may be smaller than indicated by the argument size and the attribute mp may indicate an element near the boundaries of the subset.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

Examples

```

a <- array(1:16, c(2, 2, 2, 2))
ArrayAround(data = a, mp = c(1, 1, 1, 1), size = c(0, 0, 0, 0))
## returns the first element a[1,1,1,1]

ArrayAround(data = a, mp = c(2, 2, 2, 2), size = c(0, 0, 0, 0))
## returns the last element a[2,2,2,2]

ArrayAround(data = a, mp = c(1, 1, 1, 1), size = c(1, 0, 0, 0))
## returns a[1:2,1,1,1]

ArrayAround(data = a, mp = c(1, 1, 1, 1), size = c(1, 1, 1, 1))
## returns a

ArrayAroundRandom(a)
ArrayAroundRandom(a, size = c(1, 2, 1, 2))

```

EstimateQuantile

Estimate the Quantile of a Missing Value

Description

Helper function for [Predict](#). The function estimates the quantile of the missing value at position mp from the data a relative to its image a[,mp[3],mp[4]].

Usage

```
EstimateQuantile(a, mp, nQuant, predictionInterval = FALSE)
```

Arguments

a	Numeric array with 4 dimensions.
mp	Integer vector of length 4 indexing the position of the missing value to predict.
nQuant	Integer vector of length 1. Minimum number of non-missing values in a[mp[1], mp[2], ,] required to estimate the quantile. If a[mp[1], mp[2], ,] contains less non-missing values, the neighboring values of a[mp[1], mp[2], ,] are also taken into account.
predictionInterval	Logical vector of length 1. If TRUE, the estimated quantile together with lower and upper bounds of an approximate 90% uncertainty interval is returned.

Value

If predictionInterval is FALSE, a numeric vector of length 1 being the estimated quantile of the missing value a[mp[1], mp[2], mp[3], mp[4]] is returned. Otherwise, a numeric vector of length 3 containing the estimated quantile and the lower and upper bounds of an approximate 90% uncertainty interval is returned.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

References

F. Gerber, R. de Jong, M. E. Schaepman, G. Schaepman-Strub, and R. Furrer (2018) in IEEE Transactions on Geoscience and Remote Sensing, pp. 1-13, doi: [10.1109/TGRS.2017.2785240](https://doi.org/10.1109/TGRS.2017.2785240).

See Also

[Predict](#).

Examples

```
a <- Subset(data = ndvi, mp = c(1, 3, 1, 2), i = 0)
EstimateQuantile(a = a, mp = attr(a, "mp"), nQuant = 2)
```

Extend

Implement an Alternative Gap-fill Algorithm

Description

By default, the [Gapfill](#) function uses the [Subset](#) and [Predict](#) functions to predict missing values. To implement alternative gap-fill procedures, these functions can be replaced by user defined ones and passed to the [Gapfill](#) function via the arguments `fnSubset` and `fnPredict`. The example section below gives two such extensions:

Example 1: Illustration of the concept. The prediction is the mean of the subset around a missing value.

Example 2: An algorithm using the [Score](#) and the [lm](#) functions.

Details

To work properly the user-defined `Subset` function needs to have the arguments:

data: The input data array.

mp: Numeric vector of length 4 specifying the index of the currently treated missing value.

i: Integer vector of length 1. Number of non-successfully tried subsets.

The function user-defined [Predict](#) function, needs to have the arguments:

a: Return value of the `Subset` function.

i: Integer vector of length 1. Number of non-successfully tried subsets.

Both functions may take additional arguments. The default values of these arguments can be changed via the `...` arguments of [Gapfill](#).

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

References

F. Gerber, R. de Jong, M. E. Schaepman, G. Schaepman-Strub, and R. Furrer (2018) in IEEE Transactions on Geoscience and Remote Sensing, pp. 1-13, doi: [10.1109/TGRS.2017.2785240](https://doi.org/10.1109/TGRS.2017.2785240).

See Also

[Gapfill](#), [Subset-Predict](#), [Score](#), [lm](#).

Examples

```
## Not run:
## Example 1: mean -----
## define a predict function
PredictMean <- function (a, i) mean(a, na.rm = TRUE)

out1 <- Gapfill(data = ndvi, fnPredict = PredictMean)
Image(out1$fill)

## start with a smaller subset
args(Subset)
out2 <- Gapfill(data = ndvi, fnPredict = PredictMean,
               initialSize = c(0, 0, 1, 6))
Image(out2$fill)

## require at least "nNotNA" non-NA values
## return predicted value and number of iterations i
PredictMean2 <- function (a, i, nNotNA) {
  if (sum(!is.na(a)) < nNotNA)
    return (c(NA, NA))
  c(mean(a, na.rm = TRUE), i)
}
out3 <- Gapfill(data = ndvi, fnPredict = PredictMean2, nPredict = 2,
               initialSize = c(0, 0, 1, 6), nNotNA = 0)
stopifnot(identical(c(out2$fill), c(out3$fill[,,,1])))
Image(out3$fill[,,,2]) # number of used iterations i

out4 <- Gapfill(data = ndvi, fnPredict = PredictMean2, nPredict = 2,
               initialSize = c(0, 0, 1, 6), nNotNA = 50)
Image(out4$fill[,,,1]) # fill values
Image(out4$fill[,,,2]) # number of used iterations i

## Example 2: Score() and lm() -----
PredictLm <- function (a, i, nNotNA = 50, minScores = 2){
  if (sum(!is.na(a)) < nNotNA)
    return (NA)
  am <- Array2Matrix(a)
  sx <- Score(t(am))
```

```

    lsx <- length(sx)
    if (lsx < minScores)
      return (NA)
    sy <- Score(am)
    lsy <- unique(length(sy))
    if (lsy < minScores)
      return (NA)
    df <- data.frame(z = c(am),
                     sx = rep(sx, ncol(am)),
                     sy = rep(sy, each = nrow(am)))
    newdata <- df[IndexTwoOne(attr(am, "mp"), dim(am)),]
    m <- lm(z ~ sx * sy, data = df)
    predict(m, newdata = newdata)
  }

## test PredictLm() by running it
## manually for one missing value
mp <- IndexOneFour(which(is.na(ndvi))[1], dim(ndvi))
a <- Subset(data = ndvi, mp = mp, i = 0)
PredictLm(a = a, i = 0)

## run PredictLm() on ndvi data
out5 <- Gapfill(data = ndvi, fnPredict = PredictLm,
                nNotNA = 50)
Image(out5$fill)

## End(Not run)

```

Gapfill

Main Function for Gap-Filling

Description

The function fills (predicts) missing values in satellite data. We illustrate it with MODIS NDVI data, but it can also be applied to other data, that is recorded at equally spaced points in time. Moreover, the function provides infrastructure for the development of new gap-fill algorithms. The predictions of the missing values are based on a subset-predict procedure, i.e., each missing value is predicted separately by (1) selecting a subset of the data to a neighborhood around the missing value and (2) predicting the values based on that subset.

Usage

```

Gapfill(
  data,
  fnSubset = Subset,
  fnPredict = Predict,
  iMax = Inf,
  nPredict = 1L,
  subset = "missing",

```



```

clipRange = c(-Inf, Inf),
dopar = FALSE,
verbose = TRUE,
...
)

```

Arguments

data	Numeric array with four dimensions. The input (satellite) data to be gap-filled. Missing values should be encoded as NA. When using the default Subset and Predict functions, the data should have the dimensions: x coordinate, y coordinate, seasonal index (e.g., day of the year), and year. See the ndvi dataset for an example.
fnSubset	Function to subset the data around a missing value. See Subset and Extend for more information.
fnPredict	Function to predict a missing value based on the return value of fnSubset. See Predict and Extend for more information.
iMax	Integer vector of length 1. The maximum number of iterations until NA is returned as predicted value.
nPredict	Integer vector of length 1. Specifies the length of the vector returned from fnPredict. Values larger than 1 may increase memory usage considerably.
subset	If "missing" (default), all missing values of data are replaced by predictions. If "observed", all observed values of data are replaced by predictions. If a logical array of dimension dim(data) or a vector with positive integers, the elements data[subset] are replaced by predictions. Note that independently of the specified subset all provided values in data are used to inform the predictions.
clipRange	Numeric vector of length 2. Specifies the lower and the upper bound of the filled data. Values outside this range are clipped accordingly. If nPredict is larger than 2, only the first return value of fnPredict will be clipped.
dopar	Logical vector of length 1. If TRUE, the %dopar% construct from the R package foreach is used. This allows the function to predict several missing values in parallel, if a parallel back-end (e.g., from the R package doParallel or doMpi) is available. See the example below and foreach for more information.
verbose	Logical vector of length 1. If TRUE (default), messages are printed.
...	Additional arguments passed to fnSubset and fnPredict.

Details

The predictions of the missing values are based on a subset-predict procedure, i.e., each missing value is predicted separately by (1) selecting a subset of the data to a neighborhood around it and (2) predicting the values based on that subset. The following gives more information on this subset-predict strategy.

Missing values are often unevenly distributed in data. Therefore, the size of a reasonable subset may be different depending on the position of the considered missing value. The search strategy to find that subset is encoded in fnSubset. The function returns different subsets depending on the

argument `i`. The decision whether a subset is suitable and the prediction itself is implemented in `fnPredict`. To be more specific, the subset-predict procedure loops over the following two steps to predict one missing value:

- (1) The function `fnSubset` is provided with the argument `i = i` (where `i <- 0` in the first iteration) and returns a subset around the missing value.
- (2) The function `fnPredict` decides whether the subset contains enough information to predict the missing value. If so, the predicted value is returned. Otherwise, the function returns NA and the algorithm increases `i` by one (`i <- i + 1`) before continuing with step (1).

The procedure stops if one of the following criteria is met:

- `fnPredict` returns a non-NA value,
- `iMax` tries have been completed,
- `fnSubset` returns the same subset two times in a row.

Value

List of length 4 with the entries:

- `fill` contains the gap-filled data. If `nPredict = 1`, `fill` is an array of dimension `dim(data)`, otherwise the array is of dimension `c(dim(data), nPredict)`.
- `mps` integer vector of length equaling the number of predicted values. Contains the (1 dimensional) indices of the predicted values.
- `time` list of length 4 containing timing information.
 - `start` start date and time.
 - `end` end date and time.
 - `elapsedMins` elapsed minutes.
 - `elapsedSecsPerNA` elapsed seconds per predicted value.
- `call` call used to produce the object.

Note

The default `Predict` function implements the prediction of the missing value and can also return lower and upper bounds of an approximated 90% prediction interval. See the help page of `Predict` for more information on the prediction interval. The example section below shows how the prediction interval can be calculated and displayed.

To tailor the procedure to a specific dataset, it might be necessary to adapt the subset and/or the prediction strategy. On the one hand, this can be done by changing the default arguments of `Subset` and `Predict` through the argument `...` of `Gapfill`. See the help of the corresponding functions for more information about their arguments. On the other hand, the user can define a new subset and predict functions, and pass them to `Gapfill` through the arguments `fnSubset` and `fnPredict`. See `Extend` for more information.

The current implementation of `Subset` does not take into account that values at the boundaries of data can be neighboring to each other. For example, if global data (entire sphere) are considered, `data[1,1,,]` is a neighbor of `data[dim(data)[1], dim(data)[2],,]`. Similar considerations

apply when data are available for an entire year. To take this into account, the `Subset` function can be redefined accordingly or the data can be augmented.

There are two strategies to run the gap-filling in parallel. The first one is to set the argument `dopar` of `Gapfill` to `TRUE` and to use an openMP or MPI parallel back-end. The parallel back-end needs to be setup before the call to `Gapfill`. An example using the R package `doParallel` is given below. Note that there exist other parallel back-ends implemented in other packages; such as, e.g., the package `doMpi`. Some parallel back-ends are platform dependent. While this approach shortens the process time by distributing the computational workload, it does not reduce the memory footprint of the procedure. The second strategy, which also reduces memory usage, is to split the data into several independent chunks. Whether data chunks are independent or not depends on the function provided to `fnSubset`. For example, the default `Subset` function never includes data that is further apart from the missing value than 1 seasonal index. Hence, `data[, , 1:3, ]` can be used to gap-fill `data[, , 2, ]`.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

References

F. Gerber, R. de Jong, M. E. Schaepman, G. Schaepman-Strub, and R. Furrer (2018) in IEEE Transactions on Geoscience and Remote Sensing, pp. 1-13, doi: [10.1109/TGRS.2017.2785240](https://doi.org/10.1109/TGRS.2017.2785240).

See Also

[Extend](#), [Subset-Predict](#), [Image](#).

Examples

```
## Not run:
out <- Gapfill(ndvi, clipRange = c(0, 1))

## look at input and output
str(ndvi)
str(out)
Image(ndvi)
Image(out$fill)

## run on 2 cores in parallel
if(require(doParallel)){
  registerDoParallel(2)
  out <- Gapfill(ndvi, dopar = TRUE)
}

## return also the prediction interval
out <- Gapfill(ndvi, nPredict = 3, predictionInterval = TRUE)

## dimension has changed according to 'nPredict = 3'
dim(out$fill)
```

```
## clip values outside the valid parameter space [0,1].
out$fill[out$fill < 0] <- 0
out$fill[out$fill > 1] <- 1

## images of the output:
## predicted NDVI
Image(out$fill[,,,1])
## lower bound of the prediction interval
Image(out$fill[,,,2])
## upper bound of the prediction interval
Image(out$fill[,,,3])
## prediction interval length
Image(out$fill[,,,3] - out$fill[,,,2])

## End(Not run)
```

Image	<i>Image Panels</i>
-------	---------------------

Description

Creates an image panel to visualize data in 4, 3 or 2 dimensional arrays (e.g., space-time data). The function returns a ggplot2 object, which can be modified using ggplot2 (and/or grid) syntax.

Usage

```
Image(
  x = NULL,
  zlim = range(x, na.rm = TRUE),
  col = fields::tim.colors(1000),
  theme = TRUE,
  guides = TRUE,
  na.value = "black",
  panelsByrow = TRUE,
  asRaster = TRUE,
  xlab = "",
  ylab = "",
  colbarTitle = "",
  ...
)
```

Arguments

- x Numeric array with 4, 3, or 2 dimensions containing the data to be plotted.
- zlim Numeric vector of length 2. Gives the upper and lower bound of the plotted values.
- col Vector of colors.

theme	Logical vector of length one. Should the ggplot2 theme be modified?
guides	Logical vector of length one. Should ggplot2 guides be modified?
na.value	Vector of length one. The color to be used for NA values.
panelsByrow	Logical vector of length one. Indicates the ordering of the panels.
asRaster	Logical vector of length one. If TRUE, the ordering of the pixel within images is similar to the plot method of the raster package. Otherwise, the ordering is similar to image .
xlab	Character vector (or expression) of length one giving the x-axis label.
ylab	Character vector (or expression) of length one giving the y-axis label.
colbarTitle	Character vector (or expression) of length one giving the colorbar label.
...	Additional arguments are passed to ggplot.

Value

Object (plot) of class `c("gg", "ggplot2")`.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

See Also

[ndvi](#), [ggplot2](#).

Examples

```
library("abind")
t1 <- array(rep(c(1,0), each = 5), c(5,5))
t1[5,3] <- 2
t2 <- abind(t1, t1, along = 3)
t3 <- abind(t2, t2, along = 4)
Image(t1)
Image(t2)
Image(t3)

## Not run:
Image(ndvi)

p1 <- Image(ndvi, colbarTitle = "NDVI", xlab = "Year", ylab = "DOY",
            panelsByrow = FALSE)
p1

p2 <- Image(ndvi[, , 3, 2], na.value = "white", colbarTitle = "NDVI") +
  theme(strip.text.x = element_blank(),
        strip.text.y = element_blank(),
        panel.border = element_rect(fill = NA, size = 1))
p2

## place modified color bar left
```

```

p2 + guides(fill = guide_colorbar(title = "NDVI",
                                barwidth = 1,
                                barheight = 20,
                                label.position = "right",
                                legend.position = c(0, 0))) +
  theme(legend.position = "right")

## place color bar at bottom
p2 + guides(fill = guide_colorbar(title = "NDVI",
                                barwidth = 7,
                                barheight = .7,
                                label.position = "bottom",
                                legend.position = c(0, 0)),
            direction = "horizontal") +
  theme(legend.position = "bottom")

## End(Not run)

```

Index

Index Conversions

Description

Converts an index from the first length to the second length. For example, assume that `c(2, 2)` indexes an element in a matrix with 2 rows and 5 columns. If the matrix is transformed to a vector, the same element can be accessed with the index `IndexTwoOne(c(2, 2), c(2, 5)) (=4)`.

Usage

```
IndexTwoOne(index, dimTwo)
```

```
IndexOneFour(index, dimFour)
```

Arguments

<code>index</code>	Integer vector of length 2 for <code>IndexTwoOne()</code> and length 1 for <code>IndexOneFour()</code> .
<code>dimTwo</code>	Integer vector of length 2 indicating the dimension of the 2 dimensional array.
<code>dimFour</code>	Integer vector of length 4 indicating the dimension of the 4 dimensional array.

Value

Vector of length 1 for `IndexTwoOne()` and length 4 for `IndexOneFour()`.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

Examples

```
## IndexTwoOne
IndexTwoOne(c(2, 2), c(2, 5))
v <- 1:10
dimTwo <- c(2, 5)
m <- array(v, dimTwo)
stopifnot(v[IndexTwoOne(c(2, 2), dimTwo)] == m[2,2])
```

```
## IndexOneFour
IndexOneFour(13, c(2, 2, 2, 2))
w <- 1:16
dimFour <- c(2, 2, 2, 2)
a <- array(w, dimFour)
stopifnot(a[1,1,2,2] == w[13])
```

ndvi	<i>NDVI Data from Alaska</i>
------	------------------------------

Description

The dataset was created to test gap-fill algorithms. It mimics a subset of the MODIS NDVI data (product MOD13A1) in the region of Alaska. The data product features one image per 16-day time interval, i.e., 24 images per year. The indicated images (see `Image(ndvi)`) were downloaded and stored as a 4 dimensional array. Its dimensions correspond to longitude, latitude, day of the year, and year.

Usage

```
ndvi
```

Format

Numeric array with 4 dimensions. As indicated by the `dimnames` of the array:

- dim 1: longitude,
- dim 2: latitude,
- dim 3: day of the year,
- dim 4: year.

The values are NDVI values, and hence, between 0 and 1. Missing values are encoded as NA.

Source

The actual MOD13A data product is available from NASA EOSDIS Land Processes DAAC, USGS Earth Resources Observation and Science (EROS) Center, Sioux Falls, South Dakota <https://lpdaac.usgs.gov>. MODIS data can be downloaded with the R package MODIS <https://r-forge.r-project.org/projects/modis/>.

Examples

```
str(ndvi)
Image(ndvi)
```

Score

Score Columns of a Matrix Containing NAs by its Values

Description

Helper function for [Predict](#) used to score the columns of a matrix according to their values. The scoring of a given column is done by pair-wise comparisons with all other columns. The comparison of columns is done by pair-wise comparisons of the non-missing values. This procedure is robust to missing values, if all columns of the matrix have a similar (potentially shifted) distribution of values.

Usage

```
Score(mat)
```

Arguments

mat Numeric matrix. May contain NA values.

Value

Numeric vector of length `ncol(mat)`.

Note

Interfaces a C++ function. The R package Rcpp is used.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

References

F. Gerber, R. de Jong, M. E. Schaepman, G. Schaepman-Strub, and R. Furrer (2018) in IEEE Transactions on Geoscience and Remote Sensing, pp. 1-13, doi: [10.1109/TGRS.2017.2785240](https://doi.org/10.1109/TGRS.2017.2785240).

Examples

```

mat <- rbind(c( 1,  2, NA),
             c(NA, NA,  1),
             c( 2, NA,  3),
             c( 1,  5, NA),
             c(NA,  2,  5))

s <- Score(mat)

## manual calculation in R
Mean <- function(x) mean(x, na.rm = TRUE)
sByHand <- c(Mean(c(Mean(mat[,1] > mat[,2]),
                  Mean(mat[,1] > mat[,3]))),
             Mean(c(Mean(mat[,2] > mat[,1]),
                  Mean(mat[,2] > mat[,3]))),
             Mean(c(Mean(mat[,3] > mat[,1]),
                  Mean(mat[,3] > mat[,2]))))
stopifnot(identical(s, sByHand))

```

Subset-Predict

*Subset and Predict Functions***Description**

The Subset and Predict function used in the default configuration of [Gapfill](#). To predict a missing value, the two function are called sequentially as described the help page of [Gapfill](#).

Usage

```

Subset(data, mp, i, initialSize = c(10L, 10L, 1L, 5L))

Predict(
  a,
  i,
  nTargetImage = 5,
  nImages = 4,
  nQuant = 2,
  predictionInterval = FALSE,
  qrErrorToNA = TRUE
)

```

Arguments

data	Numeric array with four dimensions. The input (satellite) data to be gap-filled. Missing values should be encoded as NA. The data should have the dimensions: x coordinate, y coordinate, seasonal index (e.g., day of the year), and year. See the ndvi dataset for an example.
------	--

<code>mp</code>	Integer vector of length 4 encoding the position of the missing value in data to predict.
<code>i</code>	Integer vector of length 1. The number of tried subsets that lead to a NA return value from Predict.
<code>initialSize</code>	Integer vector of length 4, that provides the size of the subset for $i = 0$.
<code>a</code>	Return value of <code>Subset()</code> .
<code>nTargetImage</code>	Integer vector of length 1. Minimum number of non-NA values in the image containing the missing value. If the criterion is not met, NA is returned.
<code>nImages</code>	Integer vector of length 1. Minimum number of non-empty images. If the criterion is not met, NA is returned.
<code>nQuant</code>	Integer vector of length 1. Parameter passed to EstimateQuantile .
<code>predictionInterval</code>	Logical vector of length 1. If FALSE (default), no prediction interval is returned. If TRUE, the predicted value together with the lower and upper bounds of an approximated 90% prediction interval are returned. In that case, the function returns 3 values, and hence, the argument <code>nPredict</code> of gapfill has to be set to 3 in order to store all returned values.
<code>qrErrorToNA</code>	Logical vector of length 1. If TRUE (default), an error in the quantile regression fitting leads to a NA return value. If FALSE, an error in the quantile regression fitting leads to an error and stops the prediction.

Details

The `Subset` function defines the search strategy to find a relevant subset by calling the function [ArrayAround](#). The size of the initial subset is given by the argument `initialSize`. Its default values is `c(5L, 5L, 1L, 5L)`, which corresponds to a spatial extend of 5 pixels in each direction from the missing value and includes time points having the previous, the same or the next seasonal index and are not further apart than 5 years. With an increase of the argument `i`, the spatial extent of the subset increases.

The `Predict` function decides whether the subset `a` is suitable and calculates the prediction (fill value) when a suitable subset is provided. To formulate the conditions that are used to decide if a subset is suitable, consider the subset `a` as a collection of images. More precisely, if `dim(a) = c(d1, d2, d3, d4)`, it can be seen as a collection of `d3*d4` images with an extent of `d1` by `d2` pixels. Using this terminology, we require the following conditions to be fulfilled in order to predict the missing value:

- `a` contains at least `nTargetImage` non-NA values in the image containing the missing value,
- `a` contains at least `nImages` non-empty images.

The prediction itself is based on sorting procedures (see [Score](#) and [EstimateQuantile](#)) and the quantile regression function [rq](#).

If the argument `predictionInterval` is TRUE the `Predict` functions returns the predicted value together with the lower and upper bounds of an approximated 90% prediction interval. The interval combines the uncertainties introduced by [Score](#) and [EstimateQuantile](#).

Value

Subset returns an array with 4 dimensions containing the missing value at the position indicated by the attribute mp.

Predict returns a numeric vector containing the predicted value (and if predictionInterval is TRUE, the lower and upper bounds of the prediction interval), or NA, if no prediction was feasible.

Note

The current implementation of Subset does not take into account that locations at the boundary of data can be neighboring to each other. For example, if global data (entire sphere) are considered, the location data[1, 1, ,] is a neighbor of data[dim(data)[1], dim(data)[2], ,]. Similar considerations apply when data are available for an entire year. To take this into account, the Subset function can be redefined accordingly or the data can be augmented.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

References

F. Gerber, R. de Jong, M. E. Schaepman, G. Schaepman-Strub, and R. Furrer (2018) in IEEE Transactions on Geoscience and Remote Sensing, pp. 1-13, doi: [10.1109/TGRS.2017.2785240](https://doi.org/10.1109/TGRS.2017.2785240).

See Also

[Gapfill](#), [Extend](#), [EstimateQuantile](#), [Score](#), [ndvi](#).

Examples

```
## Assume we choose c(5, 5, 1, 5) as initalSize of the subset
iS <- c(5, 5, 1, 5)
## case 1: initial subset leads to prediction -----
i <- 0
a <- Subset(data = ndvi, mp = c(1, 3, 1, 2), i = i, initialSize = iS)
p <- Predict(a = a, i = i)
p
stopifnot(identical(a, ArrayAround(data = ndvi, mp = c(1, 3, 1, 2),
                                   size = c(5 + i, 5 + i, 1, 5))))
stopifnot(identical(p, Gapfill(data = ndvi, subset = 1807,
                              initialSize = iS, verbose = FALSE)$fill[1807]))

## case 2: two tries are necessary -----
i <- 0
a <- Subset(data = ndvi, mp = c(20, 1, 1, 2), i = i, initialSize = iS)
p <- Predict(a = a, i = i)
p

## Increase i and try again.
i <- i + 1
a <- Subset(data = ndvi, mp = c(20, 1, 1, 2), i = i, initialSize = iS)
```

```

p <- Predict(a = a, i = i)
p
stopifnot(identical(a, ArrayAround(data = ndvi, mp = c(20, 1, 1, 2),
                                   size = c(5 + i, 5 + i, 1, 6))))
stopifnot(identical(p, Gapfill(data = ndvi, subset = 1784,
                              initialSize = iS, verbose = FALSE)$fill[1784]))

```

Validate

Validation with RMSE

Description

The function summarizes the validation scenario and returns the root mean squared error (RMSE) of the predictions. The typical validation procedure is: start with the `trueData`. Remove some validation points to obtain artificially generated `dataObserved`. Predicting the validation points based on `dataObserved` leads to `dataFilled`.

Usage

```

Validate(
  dataObserved,
  dataFilled,
  dataTrue,
  include = rep(TRUE, length(dataObserved))
)

```

Arguments

<code>dataObserved</code>	Numeric vector containing the observed data.
<code>dataFilled</code>	Numeric vector containing the filled (predicted) data. Needs to have the same length as <code>dataObserved</code> .
<code>dataTrue</code>	Numeric vector containing the true data. Needs to have the same length as <code>dataObserved</code> .
<code>include</code>	Logical vector indicating which element to include in the calculation of the RMSE.

Value

Numeric matrix with one 1 row and 6 columns having the entries:

- `nNA`: number of missing values in `dataObserved`,
- `nFilled`: number of predicted values,
- `nNotFilled`: number of not predicted missing values,
- `ratioFilled`: ratio: `nFilled / nNA`,
- `nCrossvali`: number of values for validation,
- `RMSE`: root mean squared error.

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>.

Examples

```
Validate(c(1, NA, 2, NA), c(1, 2, 2, NA), c(1, 1, 2, 2))

## validate gap-fill predictions: consider the ndvi data
Image(ndvi)

## define some validation points vp
## in the image of the day 145 of the year 2004
vp <- 300 + c(5:10) + rep(21 * c(0:5), each = 6)

## remove the vp values from the data
nn <- ndvi
nn[vp] <- NA
Image(nn)

## predict the vp values
out <- Gapfill(nn, subset = vp)
Validate(dataObserved = nn, dataFilled = out$fill,
         dataTrue = ndvi)
```

Index

- * **package**
 - gapfill-package, [2](#)
- alternative (Extend), [6](#)
- Array2Matrix, [3](#)
- ArrayAround, [3](#), [4](#), [18](#)
- ArrayAroundRandom (ArrayAround), [4](#)
- EstimateQuantile, [5](#), [18](#), [19](#)
- Extend, [2](#), [3](#), [6](#), [9–11](#), [19](#)
- extend (Extend), [6](#)
- fnPredict (Subset-Predict), [17](#)
- fnSubset (Subset-Predict), [17](#)
- foreach, [9](#)
- Gap-fill (Gapfill), [8](#)
- gap-fill (Gapfill), [8](#)
- Gapfill, [2](#), [3](#), [6](#), [7](#), [8](#), [17](#), [19](#)
- gapfill, [18](#)
- gapfill (Gapfill), [8](#)
- Gapfill-Package (gapfill-package), [2](#)
- Gapfill-package (gapfill-package), [2](#)
- gapfill-Package (gapfill-package), [2](#)
- gapfill-package, [2](#)
- ggplot2, [13](#)
- Image, [2](#), [3](#), [11](#), [12](#)
- image, [13](#)
- Index, [3](#), [14](#)
- IndexOneFour (Index), [14](#)
- IndexTwoOne (Index), [14](#)
- lm, [6](#), [7](#)
- ndvi, [9](#), [13](#), [15](#), [19](#)
- Predict, [2](#), [5](#), [6](#), [9](#), [10](#), [16](#)
- Predict (Subset-Predict), [17](#)
- rq, [18](#)
- Score, [6](#), [7](#), [16](#), [18](#), [19](#)
- Subset, [2](#), [6](#), [9–11](#)
- Subset (Subset-Predict), [17](#)
- Subset-Predict, [17](#)
- Validate, [20](#)
- validate (Validate), [20](#)