

Package ‘phenopix’

May 9, 2026

Type Package

Title Process Digital Images of a Vegetation Cover

Version 2.4.5

Date 2025-02-10

Maintainer Gianluca Filippa <gian.filippa@gmail.com>

Description A collection of functions to process digital images, depict greenness index trajectories and extract relevant phenological stages.

Depends R (>= 2.15.3)

Imports zoo, plyr, jpeg, stringr (>= 1.0.0), strucchange, parallel, foreach, doParallel, iterators, gtools, raster, sp, terra (>= 1.8.21)

License GPL-2

LazyLoad yes

NeedsCompilation no

Author Gianluca Filippa [aut, cre],
Edoardo Cremonese [aut],
Mirco Migliavacca [aut],
Marta Galvagno [aut],
Matthias Folker [aut],
Andrew D. Richardson [aut],
Enrico Tomelleri [aut]

Repository CRAN

Date/Publication 2025-02-13 16:30:06 UTC

Contents

phenopix-package	3
autoFilter	5
bartlett2009	7
bartlett2009.filtered	7
bartlett2009.fitted	8

bartlett2009.processed	8
BeckFit	9
combineUncertainty	10
convert	11
convertROI	12
CutSeason	13
DrawMULTIROI	14
editExposure	15
ElmoreFit	16
extract	17
extractDateFilename	18
extractParameters	19
extractVIs	20
FitDoubleLogBeck	21
FitDoubleLogElmore	22
FitDoubleLogGu	23
FitDoubleLogKIHeavy	24
FitDoubleLogKILight	25
fitted.phenopix	26
get.options	27
getCoords	28
getExposure	29
greenClusters	30
greenExplore	31
greenProcess	32
GuFit	34
hydrodoy	35
KlostermanFit	36
matchExposure	37
NDVI	39
PasteSeason	40
PhenoBP	40
PhenoDeriv	41
PhenoExtract	42
PhenoGu	44
PhenoKl	45
PhenoPlot	47
PhenoTrs	48
plot.phenopix	49
plotBP	50
plotExplore	51
plotSpatial	52
plotSum	53
plotVI	54
print.phenopix	55
PrintROI	55
resizeImage	56
spatialFilter	57

spatialGreen 58
 SplineFit 59
 splitROI 60
 structureFolder 61
 summarizePhases 61
 summary.phenopix 63
 trainOCR 63
 update.phenopix 64
 updateROI 65

Index 67

phenopix-package *A package for the processing of images of a vegetation cover*

Description

The package provides functions to process digital images, depict greenness index (and a number of other indices) trajectories and extract relevant phenological stages.

Details

Package: phenopix
 Type: Package
 Version: 2.4.3
 Date: 2023-08-09
 License: GPL-2

The package provides functions to process digital images, depict greenness index trajectories and extract relevant phenological stages.

The first step of the work is to set a region of interest on the images. The function DrawROI does the work. More than one ROI can be specified.

Second step is extracting information from the ROI(s). The function extractVIs computes several vegetation indexes on image pixels falling within the ROI(s). The function works recursively within a folder so that all images can be processed and a time series of the computed indexes is extracted. A specific function extractDateFilename retrieves a timestamp from the filename of the images. Time series of green, red and blue chromatic coordinates and brightness are returned for each ROI of the image.

Third step is data filtering. The function autoFilter provides several filtering methods that can be used one at a time or in sequence.

Fourth step is fitting a curve to the data. Five methods have been included in the package. The function SplineFit fits a smoothed cubic spline to the data. The remaining four methods are based on the optimization of different double logistic equations retrieved from the recent literature. An uncertainty calculation is provided, based on the residuals between observed and predicted values. The variability in the residuals is used to generate random noise, that is then added to raw data in a

bootstrap procedure. From this procedure an ensemble of equation parameters and/or of curves are generated.

Fifth step is the extraction of phenological thresholds. Five methods have been included to extract phenological dates. One is based on thresholds on the spline curve and one on breakpoints analysis, whereas other methods work on inflection points of the derivatives in various ways. Uncertainty on curve fitting is extended to threshold extraction, so that also for this step uncertainty is easily estimated.

The package offers a variety of fittings and phenophase methods so to be as flexible as possible in order to apply to very different boreal (sofar) ecosystems, ranging from high latitude/altitude grasslands to tropical forests. Ecosystems that show multiple seasonal peaks can be splitted in subseason with the `CutSeason` function and processed separately. Dedicated plotting functions provide an easy way to look at fitting and thresholding with annotated graphs. The package is being tested on the PHENOCAM dataset (<http://phenocam.sr.unh.edu/webcam/>), and constantly debugged.

Detailed vignettes are available in my researchGate page, specifically a base vignette (https://www.researchgate.net/publication/289355121_phenopix_R_package_vignette) and one dedicated to the spatial analysis (https://www.researchgate.net/publication/289355121_phenopix_R_package_vignette_based_analysis_vignette). Here the user can find a walk-through to understand main functions, and proceed step-by-step through the analysis.

Recent changes:

As from version 2.4 the function `drawROI` was substituted by the function `drawMULTIROI`. This implies few changes: the object `roi.data` is no longer structured as before. Now it is a list with an element for each ROI. For each element, there is an item named 'mask' containing the binary raster mask (dimensions and resolution as pixel size of the original image where ROI was drawn) and polygons (an object of class `SpatialPolygons`). The relevant changes in a processing chain that may occur due to these changes are partially accommodated with the new function `convertROI`.

Author(s)

Gianluca Filippa, Edoardo Cremonese, Mirco Migliavacca, Jeroen Staab

Maintainer: Gianluca Filippa <gian.filippa@gmail.com>

References

Gu L, Post WM, Baldocchi D, Black TA, Suyker AE, Verma SB, Vesala T, Wofsy SC. (2009) Characterizing the Seasonal Dynamics of Plant Community Photosynthesis Across a Range of Vegetation Types. In: Phenology of Ecosystem Processes (Ed: Noormets A, Springer New York), pp 35-58.

Klosterman ST, Hufkens K, Gray JM, Melaas E, Sonnentag O, Lavine I, Mitchell L, Norman R, Friedl MA, Richardson A D (2014) Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using PhenoCam imagery, *Biogeosciences*, 11, 4305-4320, doi:10.5194/bg-11-4305-2014.

Migliavacca, M., Galvagno, M., Cremonese, E., Rossini, M., Meroni, M., Sonnentag, O., Cogliati, S., Manca, G., Diotri, F., Busetto, L., Cescatti, A., Colombo, R., Fava, F., Morra di Cella, U., Pari, E., Siniscalco, C., Richardson, A.D., 2011. Using digital repeat photography and eddy covariance data to model grassland phenology and photosynthetic CO₂ uptake. *Agric. Forest Meteorol.* 151, 1325-1337.

Papale, D., Reichstein, M., Aubinet, M., Canfora, E., Bernhofer, C., Kutsch, W., Longdoz, B., Rambal, S., Valentini, R., Vesala, T., Yakir, D., 2006. Towards a standardized processing of Net Ecosystem Exchange measured with eddy covariance technique: algorithms and uncertainty estimation. *Biogeosciences* 3, 571-583.

Sonnentag, O., Hufkens, K., Teshera-Sterne, C., Young, A.M., Friedl, M., Braswell, B.H., Milliman, T., O'Keefe, J., Richardson, A.D., 2012. Digital repeat photography for phenological research in forest ecosystems. *Agric. Forest Meteorol.* 152, 159-177.

Zhang X, Friedl MA, Schaaf CB, Strahler AH, Hodges JCF, Gao F, Reed BC, Huete A (2003) Monitoring vegetation phenology using MODIS, *Remote Sens. Environ.*, 84, 471-475.

See Also

greenbrown.r-forge.r-project.org

autoFilter

A multiple-approach filtering procedure

Description

5 different filters can be applied to raw green coordinate values. Filters can be applied alone or in sequence, in this case order matters. Available filters are 'night', 'max', 'spline', 'blue' and 'mad'. See details for further explanation.

Usage

```
autoFilter(data, dn=c('ri.av', 'gi.av', 'bi.av'), raw.dn = FALSE,
           brt = 'bri.av', na.fill = TRUE, filter = c("night", "spline", "max"),
           filter.options = NULL, plot = TRUE, ...)
```

Arguments

data	A data.frame containing a POSIX vector
dn	The column positions or colnames for red, green and blue digital numbers, in this order. The default is suited to work with a dataframe in output from ExtractVIs
raw.dn	If TRUE you must provide raw digital numbers (range 0-255), for which position or colnames must be provided in dn. In this case gcc is calculated as g/brt where g is raw green digital number and brt is brightness computed as the sum of r + g + b. If FALSE (default) the function expects that in dn you provide colnames or col positions of relative red green and blue, respectively.
brt	The column position for brightness. Used in the filtering procedure. If you provide raw digital numbers and brt=NULL, it will be calculated from raw red green and blue dn.
na.fill	If FALSE, discarded data are filled with NA. If TRUE a call to na.approx from package zoo linearly interpolates between existing records up to a gap of 10 missing values.

<code>filter</code>	Character names for filters. Chose one (or more) between 'night', 'max', 'spline', 'blue' and 'mad'. The order provided in this argument determines the order of application of filters. See details for further information on filters.
<code>filter.options</code>	If NULL <code>filter.options</code> defaults to set values, otherwise they must be specified in a named list. Filter options are designed to work with a wide range of data, the user must change them with caution. See <code>get.options</code> to change default filter options.
<code>plot</code>	If TRUE a diagnostic plot of the different filtering effects is returned.
<code>...</code>	Further options, currently not used

Details

This function takes a `data.frame` with raw digital numbers of red green and blue found in `dn`, converts them in the respective coordinates (example for red: $rcc=r/(r+g+b)$). If `raw.dn=FALSE`, the function takes directly color's chromatic coordinates provided in `dn`, i.e. relative values. In the `dataframe` a POSIX vector must be provided, which will subsequently be converted in numeric day of year (`doy`). Afterwards, the filtering procedure starts. Filters are applied in the order provided in `filter`. Night filter removes records under a certain `gcc` value (as specified in `filter.options`). The default is 0.2. Blue filter is intended to remove bad images and is very aggressive. It is suggested only for very low quality images. The daily mean and standard deviation on `bcc` is computed and a `sd` threshold is computed as the quantile of standard deviations with `prob = 0.05`. An envelope is then computed as daily mean `bcc` +/- the calculated threshold. Raw data outside this envelope are discarded. The `mad` filter is applied according to Papale et al 2006 (biogeosciences) created to remove spikes on FLUXNET data. The `max` filter is based on Sonnentag et al (2012) and computes the 90% of the curve based on a three days moving window. The `spline` filter is based on Migliavacca et al (2011). Default values in function's arguments are suited to data in output from the function `ExtractVIs`. Note that computing relative greenness index within the function `autoFilter`, i.e., providing raw `dn` and with `raw.dn` set to TRUE will produce slightly different values than 'gi.av' in output from `ExtractVIs`. This results from the difference between computing pixel based brightness (and color indices) or ROI average brightness as it happens in `autoFilter` with `raw.dn = TRUE`.

Value

A multivariate `zoo` object with raw data and a column with `gcc` values after filtering. Colnames for filtered data have the same name as the applied filter.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Sonnentag et al.
 Migliavacca et al.
 Papale et al.

Examples

```
## Not run:
data(bartlett2009)
## with raw.dn ==TRUE and column position for chromatic coordinates
par(mfrow=c(3,1), oma=c(5,4,4,2), mar=rep(0,4))
filtered.data <- autoFilter(bartlett2009, dn=c(5:7),
  filter=c('night', 'spline', 'max'),
  filter.options=NULL, raw.dn=TRUE)
## same as before but returning unfilled data
filtered.data <- autoFilter(bartlett2009, dn=c(5:7),
  filter=c('night', 'spline', 'max'),
  filter.options=NULL, na.fill=FALSE, raw.dn=TRUE)
## all filters in action (check the plot)
filtered.data <- autoFilter(bartlett2009, dn=c(5:7),
  filter=c('night','blue','mad','spline','max'),
  filter.options=NULL, raw.dn=TRUE)
## check filter names
names(filtered.data)

## End(Not run)
```

 bartlett2009

Bartlett 2009 raw data

Description

Raw data from the PHENOCAM database, site: Bartlett, year:2009.

Usage

```
data(bartlett2009)
```

Examples

```
data(bartlett2009)
## maybe str(bartlett2009) ; plot(bartlett2009) ...
```

 bartlett2009.filtered *Bartlett 2009 dataset filtered*

Description

A yearly time series of filtered green chromatic coordinates from the PHENOCAM database, site Bartlett, year 2009

Usage

```
data(bartlett2009.filtered)
```

Examples

```
data(bartlett2009.filtered)
## maybe str(bartlett2009.filtered) ; plot(bartlett2009.filtered) ...
```

```
bartlett2009.fitted
```

Bartlett 2009 dataset with computed fitting and uncertainty estimation

Description

A list of predicted values, equation parameters and their uncertainty from fitting the Klosterman equation to Bartlett 2009 filtered data.

Usage

```
data(bartlett2009.fitted)
```

Examples

```
data(bartlett2009.fitted)
## maybe str(bartlett2009.fitted) ; plot(bartlett2009.fitted) ...
```

```
bartlett2009.processed
```

Bartlett 2009 dataset processed by greenExplore function

Description

A complex list accessible via multiple generic functions like plot, print, summary, extract

Usage

```
data(bartlett2009.processed)
```

Examples

```
## Not run:
data(bartlett2009.processed)
plot(bartlett2009.processed)
## maybe str(bartlett2009.fitted) ; plot(bartlett2009.fitted) ...

## End(Not run)
```

BeckFit	<i>Fit a double logistic equation to a time series according to Beck et al. (2006)</i>
---------	--

Description

This function fits a double logistic curve to observed values using the function as described in Beck et al. (2006) (equation 3). It can also provide and uncertainty estimation. Rather internal function. See [greenProcess](#)

Usage

```
BeckFit(ts, uncert = FALSE, nrep = 100, ncores='all',  
sf=quantile(ts, probs=c(0.05, 0.95), na.rm=TRUE))
```

Arguments

ts	A ts or zoo object with gcc data. <code>index(ts)</code> must be numeric days of year (doys) or a POSIXct vector
uncert	Should uncertainty be estimated?
nrep	Number of relications to estimate uncertainty, defaults to 1000.
ncores	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.

Details

The function estimates parameters of the double logistic equation from Beck et al. 2006 and provides an uncertainty estimation. Parameters are estimated by a call to the function [FitDoubleLogBeck](#) from the [greenbrown](#) package. Uncertainty is computed by adding noise to the raw data and by estimating again the parameters. Noise is added according to the standard deviation of the residuals (fitted - observed). The procedure is repeated nrep times.

Value

A list containing the following items.

fit	A list as returned by the function FitDoubleLogBeck
uncertainty	A list containing a zoo data.frame with the uncertainty predicted values, and a dataframe containing the respective uncertainty curve parameters

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Beck, P.S.A., C. Atzberger, K.A. Hodga, B. Johansen, A. Skidmore (2006): Improved monitoring of vegetation dynamics at very high latitudes: A new method using MODIS NDVI. - Remote Sensing of Environment 100:321-334.

Examples

```
## Not run:
library(zoo)
data(bartlett2009.filtered)
## fit without uncertainty estimation
fitted.beck <- BeckFit(bartlett2009.filtered)
days <- as.numeric(format(index(bartlett2009.filtered), '
plot(days, bartlett2009.filtered)
lines(fitted.beck$fit$predicted, col='red')
## look at fitting parameters
fitted.beck$fit$params
## look at fitting equation, where t is time
fitted.beck$fit$formula

## End(Not run)
```

combineUncertainty *An evolution of [greenProcess](#) for the combination of uncertainty after processing*

Description

The combineUncertainty uses [greenProcess](#) to fit all available double logistic equations in the phenopix package and extracts thresholds with all available methods. Then uncertainties can be combined and returned by using [summarizePhases](#) and plotted with [plotSum](#). See [greenProcess](#).

Usage

```
combineUncertainty(ts, which='all', nrep=50, ncores='all')
```

Arguments

ts	A ts or zoo object with gcc data. index(ts) must be numeric days of year (doys) or a POSIXct vector
which	It can be 'all' (default) and all 4 double logistic fits will be calculated (beck, elmore, elosterman, gu), or a vector of subsets of the four fits
nrep	Number of relications to estimate uncertainty for each single fitting, defaults to 50.
ncores	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.

Details

This function uses [greenProcess](#) to fit all available double logistic equations in the phenopix package and extracts thresholds with all available methods. Then uncertainties can be combined and returned by using [summarizePhases](#) and plotted with [plotSum](#). See [greenProcess](#), [summarizePhases](#), [plotSum](#). This function uses a modellistic approach to combine all uncertainties from all available phenopix fittings, as to get an ensemble of phases with different methods, without necessarily choosing any of them.

Value

A named list with dataframes for each phenophase method with all replication for each of the included fitting methods. These data can then be combined with the companion functions [summarizePhases](#) and [plotSum](#). See examples for details.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
  require(zoo)
  data(bartlett2009.filtered)
  combined.fit <- combineUncertainty(na.approx(bartlett2009.filtered), nrep=100)
# 100 replications for each fitting
  names(combined.fit) # a dataframe for each phenoMethod + a list with all fittings
  fit.summary <- summarizePhases(combined.fit, across.methods=TRUE)
## again a list with one element for each fitting method + two additional items
## if across.methods is TRUE, which combines gu + klosterman phenophase methods
## in a single method, and the same happens for trs and derivatives
  plotSum(bartlett2009.filtered, fit.summary, which='klosterman')
## a plot with original timeseries + phenophases and their uncertainty

## End(Not run)
```

 convert

Convert in data.frame a zoo or ts object

Description

This function converts an object of class `ts` or `zoo` in a `data.frame` with a column named `time` retrieved from `index(x)`. It is designed for those unfamiliar with time series objects.

Usage

```
convert(x, year=NULL)
```

Arguments

x	An object of class <code>ts</code> or <code>zoo</code>
year	A vector of length one with year of observation. If provided, a column with time in POSIX format is also returned

Details

An object of class `data.frame` is returned with a column `doy` and optionally `time` at the end of the table.

Value

A dataframe

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

convertROI	<i>Convert a region of interest object from the old structure to the new (phenopix > 2.4)</i>
------------	--

Description

This function allows to convert the `roi.data` stored in the `roi.data.Rdata` object built with `phenopix` older than version 2.4 into the new structure better described in the details section. The change was needed because a new method for drawing a region of interest was conceived, which includes the possibility to draw multiple polygons for a single ROI. The new function that does the job is [DrawMULTIROI](#), which fully substitutes the old `DrawROI`.

Usage

```
convertROI(path_img_ref, path_ROIs, file.type='.jpg')
```

Arguments

<code>path_img_ref</code>	Path in your folder for the reference image
<code>path_ROIs</code>	Path where <code>roi.data</code> object created with <code>phenopix < 2.4</code> was built.
<code>file.type</code>	It must match the syntax of your file extension (e.g. <code>.jpg</code> , <code>.JPG</code> , <code>.JPEG</code>). Multiple types are allowed by concatenation with <code>c()</code> .

Details

This function allows to convert the `roi.data` stored in the `roi.data.Rdata` object built with phenopix older than version 2.4 into the new structure better described in the details section. The changed was needed because a new method for drawing a region of interest was conceived, which includes the possibility of drawing multiple polygons for a single ROI. There is a second reason why `DrawROI` had to disappear: it included C++ code from the package `SDMTools` which is currently orphaned. The new function that allows multiple polygons in a single ROI is `DrawMULTIROI`, which fully substitutes the old `DrawROI`.

Value

A list containing the following:

<code>mask</code>	A binary mask (a raster object)
<code>polygons</code>	An object of class <code>SpatialPolygons</code>

Additionally, a jpeg image is returned with the ROI(s) drawn. The same object that is returned is saved in the path specified in `path_ROIs`

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

CutSeason

Cut time series by visual estimation

Description

This function allows to cut a yearly time series where multiple cycles are apparent into separated time series. This is done by visual estimation using `locator` function.

Usage

```
CutSeason(data, plot = FALSE)
```

Arguments

<code>data</code>	A <code>data.frame</code> , ideally a yearly time series of daily values.
<code>plot</code>	Should a plot be returned? If so different colors for different splits are returned.

Details

This function allows to split a season of data according to visual estimation. The required time series is plotted and a call to `locator` enables the user to click on the split. A list is returned with a number of `data.frame`s. See `locator` for details on how to switch off the on-screen locator, which depends upon the graphic device. Note that all breaks must be provided, i.e. at the beginning and end of the sub-seasonal cycle(s).

Value

A list with `n` `data.frames`, where `n` is the number of splits

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[PasteSeason](#)

DrawMULTIROI

Draw a Region of Interest

Description

This function allows to draw one (or more) region(s) of interest (ROI) on a jpeg image. After phenopix version 2.4 this function allows to draw a region of interest potentially consisting of different polygons.

Usage

```
DrawMULTIROI(path_img_ref, path_ROIs, nroi = 1,  
roi.names, file.type='.jpg')
```

Arguments

<code>path_img_ref</code>	Path in your folder for the reference image
<code>path_ROIs</code>	Path where to store image with ROI and ROI coordinates.
<code>nroi</code>	The number of ROIs you want to draw.
<code>roi.names</code>	A character vector with ROI names.
<code>file.type</code>	It must match the syntax of your file extension (e.g. <code>.jpg</code> , <code>.JPG</code> , <code>.JPEG</code>). Multiple types are allowed by concatenation with <code>c()</code> .

Details

The function allows to draw one or more ROIs on an image or to load saved ROIs. The function uses `locator` to locate points, closes the polygon and stores an `RData` with coordinates. The use of `locator` is restricted to only some graphic devices. The function attempts to open an `X11()` device. In Mac OS the polygon is closed by typing `ESC` key. See `locator` for details. After package version 2.4, this function allows to build a ROI based on multiple polygons. As from this version of the package the way the `roi.data` object is structured changes substantially compared to previous versions. The old `drawROI()` function is no longer present. The new function [convertROI](#) allows to convert `roi.data` objects generated with former `DrawROI` function (phenopix < 2.4) into the new `roi.data` structure.

Value

A list containing the following:

mask A binary mask (a raster object)
polygons An object of class SpatialPolygons

Additionally, a jpeg image is returned with the ROI(s) drawn. The same object that is returned is saved in the path specified in path_ROIs

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

editExposure *Change erroneous exposure values by hand*

Description

This function allows to fix wrong exposure values by hand. Use in combination with [getExposure](#). See [getExposure](#)

Usage

```
editExposure(exposures, image.path, which)
```

Arguments

exposures A data.frame with the exposures as in output from [getExposure](#)
image.path The path to the folder where the images are stored, from which exposures were extracted
which A vector with row positions in your data.frame where exposures must be converted.

Details

After the extraction of exposure via the [getExposure](#) function, some of them will likely require manual correction. This function makes this process easy by plotting on screen the actual exposure and the estimated. If they don't agree you can manually correct it by typing into the R console.

Value

The corrected exposures data.frame.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

ElmoreFit	<i>Fit a double logistic equation to a time series according to Elmore et al. (2012)</i>
-----------	--

Description

This function fits a double logistic curve to observed values using the function as described in Elmore et al. (2012) (equation 4). It can also provide and uncertainty estimation.

Usage

```
ElmoreFit(ts, uncert = FALSE, nrep = 100, ncores='all',
sf=quantile(ts, probs=c(0.05, 0.95), na.rm=TRUE))
```

Arguments

ts	A ts or zoo object with gcc data. <code>index(ts)</code> must be numeric days of year (doys) or a POSIXct vector
uncert	Should uncertainty be estimated?
nrep	Number of replications to estimate uncertainty, defaults to 100.
ncores	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. <code>greenProcess</code> sf is automatically calculated.

Details

The function estimates parameters of the double logistic equation from Elmore et al. 2012 and provides an uncertainty estimation. Parameters are estimated by a call to the function `FitDoubleLogElmore` from the `greenbrown` package. Uncertainty is computed by adding noise to the raw data and by estimating again the parameters. Noise is added according to the standard deviation of the residuals (fitted - observed). The procedure is repeated `nrep` times.

Value

A list containing the following items.

fit	A list as returned by the function <code>FitDoubleLogElmore</code>
uncertainty	A list containing a zoo data.frame with the uncertainty predicted values, and a dataframe containing the respective uncertainty curve parameters

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Elmore, A.J., S.M. Guinn, B.J. Minsley and A.D. Richardson (2012): Landscape controls on the timing of spring, autumn, and growing season length in mid-Atlantic forests. - *Global Change Biology* 18, 656-674.

See Also

[FitDoubleLogElmore](#)

Examples

```
## Not run:
library(zoo)
data(bartlett2009.filtered)
## fit without uncertainty estimation
fitted.elmore <- ElmoreFit(bartlett2009.filtered)
days <- as.numeric(format(index(bartlett2009.filtered), '
plot(days, bartlett2009.filtered)
lines(fitted.elmore$fit$predicted, col='red')
## look at fitting parameters
fitted.elmore$fit$params
## look at fitting equation, where t is time
fitted.elmore$fit$formula

## End(Not run)
```

extract

Extract items from an object of class phenopix

Description

This function allows to extract items from an object of class phenopix.

Usage

```
extract(x, what)
```

Arguments

x	An object of class phenopix
what	One between "data", "fitted", "metrics", "metrics.uncert", "curve.params", "curve.uncert", "curve.params.uncert"

Details

This function allows to extract items from an object of class phenopix. "data" extracts row data "fitted" extracts fitted data "metrics" extracts metrics data.frame "metrics.uncert" extracts metrics uncertainty dataframe "curve.params" extracts curve parameters "curve.uncert" extracts a family of fitted curves estimated by the uncertainty procedure "curve.params.uncert" extracts a family of curve parameters estimated by the uncertainty procedure

Value

A dataframe

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

extractDateFilename *Extract dates from filenames*

Description

This function extracts dates from filenames.

Usage

```
extractDateFilename(filename, date.code)
```

Arguments

filename	The filename where to retrieve time stamp
date.code	The format of your date in filename, see details.

Details

This function allows the extraction of the date (hour, doy, dayfract) from the filename. The only mandatory rules are (1) that site name come first and date after and (2) sitename and date must be separated by an underscore. In date.code provide the format of your date, using lower letters for year (y) month (m) and day (d) and upper letters for hour (H) and minute (M). As an example: If your file is named: 'sitename_2012_03_03_15-30.jpg' than your date.code is "yyyy_mm_dd_HH-MM". If your file is named 'sitename_12.03.03.1530.jpg' than your date.code is "yy.mm.dd.HHMM" If hours and minutes are missing in your filename, conversion defaults to 12:00.

Value

A POSIX string containing date,Hour,DOY,DOY.dayfract of the entire images time series

Author(s)

Edoardo Cremonese <e.cremonese@arpa.vda.it>

extractParameters	<i>Extract fitting parameters from a pixel-based analysis</i>
-------------------	---

Description

This function allows to extract fitting parameters from pixel-based analysis of a seasonal ensemble of images in one or multiple rois.

Usage

```
extractParameters(list, update=NULL, ...)
```

Arguments

<code>list</code>	A list as in output from <code>spatialGreen</code> .
<code>update</code>	One between 'trs', 'derivatives', 'klosterman', 'gu' or NULL (the default). See update from this package for further details.
<code>...</code>	Further arguments to the update function. Currently, you can specify a <code>trs</code> argument for update method <code>trs</code> for specific thresholds. See <code>PhenoExtract</code> for further details.

Details

This function allows to extract curve parameters and thresholds from pixel based analysis. Depending on how you have run `spatialGreen` function, `extractParameters` behaves differently (but returns the same results!). In case you haven't saved each pixel's fit with `spatialGreen` (i.e., with `save == FALSE` and `assign == TRUE`), this function will extract parameters from a list as in output from `spatialGreen`. IN case you HAVE saved each pixel's fit (i.e. `save == TRUE` in function `spatialGreen`) `extractParameters` loads recursively in a folder (`path`) and builds a `data.frame` with extracted curve parameters and thresholds. In both cases a `data.frame` is returned, with all curve parameters and thresholds for each pixel (one row for each pixel). Additionally, RMSE (root mean square error) for each fitting is computed and returned as well. The dataframe in output is suitable to be passed to the function `plotSpatial`.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

 extractVIs

Extracts various vegetation indexes in a ROI

Description

This function allows to extract various vegetation indexes in a region of interest of a jpeg image.

Usage

```
extractVIs(img.path, roi.path, vi.path = NULL, roi.name = NULL,
  plot = TRUE, begin = NULL, spatial = FALSE, date.code,
  npixels=1, file.type='.jpg', bind=FALSE, shift.matrix=NULL, ncores='all', log.file=NULL)
```

Arguments

<code>img.path</code>	Path to image folder, or a vector with all image files (full path mandatory)
<code>roi.path</code>	Path to stored RData with ROI, see DrawROI
<code>vi.path</code>	Where to save the output. If NULL data are saved in the working directory.
<code>roi.name</code>	The name of the ROI
<code>plot</code>	Should a diagnostic plot with red, green and blue indexes be returned?
<code>begin</code>	The user can provide a beginning date as a character string in the form 'YYYY-MM-DD'. Images with a time stamp earlier than the provided date will be discarded. Default is NULL and the whole time series will be processed. Useful when updating an existing time series.
<code>spatial</code>	If <code>spatial = FALSE</code> (the default) vegetation indexes are averaged over the entire ROI. If set to <code>FALSE</code> red green and blue digital numbers are extracted for each pixel in the ROI. Since this option builds large objects, it is recommended to use it on resized images, so to have to more than 10k pixels in the ROI.
<code>date.code</code>	Provide the (quoted) exact format of the date embedded in your image names. Example: your image file picked on June 14th, 2012 at 12 pm is named "site_12.06_14.1200.png", than your <code>date.code</code> will be "yy.mm_dd.HHMM".
<code>npixels</code>	This argument allows to sample the images at lower resolution. Specify by an integer the number of pixels you want to aggregate (e.g. if <code>npixels=2</code>), a square of 2 pixels per side is aggregated in one pixel.
<code>file.type</code>	It must match the syntax of your file extension (e.g. .jpg, .JPG, .JPEG). Multiple types are allowed by concatenation with <code>c()</code> .
<code>bind</code>	If <code>TRUE</code> and argument <code>begin</code> is defined, then new VI.data are binded to already existing data and overwritten, after checking for duplicated records (and returning a warning in case there are).
<code>shift.matrix</code>	If non NULL a matrix of two-dimentional shifts to adjust ROI masks to accommodate for field-of-view shifts. This subject will be soon implemented in the package.

ncores	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
log.file	It can be NULL or a path where to generate and refresh a txt file which logs the progress of the filtering procedure

Details

In the pixels contained in a ROI (which are loaded from the given folder), various vegetation indexes are calculated. Raw red, green and blue dn are averaged over the roi. The standard deviation is also calculated. For each pixel the brightness is computed as the sum of $r + g + b$. Then, for each pixel red, green and blue chromatic coordinates (aka relative indices) are calculated as $r/\text{brightness}$. After the calculation, all pixel data are averaged over the ROI. ExtractDateFilename is then called to get a time vector and a doy. The function applies recursively for all images in a folder and allows to get a multivariate time series of the various indexes. A plot is also generated and saved as .png if `plot=TRUE` Option `begin` allows to update the process from a given date without reprocessing the whole time series of images. In this case a separate Rdata will be saved (and not overwritten to an already existing one) with the date of `begin` in the filename. Option `spatial` allows to compute digital numbers for each pixel instead of averaging them over the entire ROI. Note: Brightness and relative color indices are calculated for each pixel and THEN averaged over the ROI. This is different from averaging the brightness and raw colors over the entire ROI and THEN calculate a ROI-based relative color index. This second approach is used within the function `autoFilter` in the case when ROI aggregated raw dn are provided and brightness is calculated from them.

Value

A list with one data.frame for each ROI containing the multivariate time series of various vegetation indices if `spatial = FALSE`. Else, a list with one list for each ROI. Each sublist will be a named list that can be handled with appropriate functions, different from the one used if `spatial = FALSE`.

Author(s)

Edoardo Cremonese <e.cremonese@arpa.vda.it>, Gianluca Filippa <gian.filippa@gmail.com>

FitDoubleLogBeck *A fit*

Description

See FitDoubleLogBeck from package greenbrown

Usage

```
FitDoubleLogBeck(x, t = index(x), tout = t, weighting = TRUE,
  return.par = FALSE, plot = FALSE, hessian=FALSE,
  sf=quantile(x, probs=c(0.05, 0.95), na.rm=TRUE), ...)
```

Arguments

x	A vector or, better, an univariate ts or zoo object.
t	A vector of time (in numeric doys), if not provided index(x) is retrieved.
tout	For gapfilling purposes, a vector of time steps at which the function can be predicted.
weighting	Should fit be weighted?
return.par	Currently unused.
plot	Currently unused.
hessian	Currently unimplemented.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.
...	Further arguments currently unused.

FitDoubleLogElmore *A fit*

Description

See FitDoubleLogElmore from package greenbrown

Usage

```
FitDoubleLogElmore(x, t = index(x), tout = t,
  return.par = FALSE, plot = FALSE, hessian=FALSE,
  sf=quantile(x, probs=c(0.05, 0.95), na.rm=TRUE), ...)
```

Arguments

x	A vector or, better, an univariate ts or zoo object.
t	A vector of time (in numeric doys), if not provided index(x) is retrieved.
tout	For gapfilling purposes, a vector of time steps at which the function can be predicted.
return.par	Currently unused.
plot	Currently unused.
hessian	Currently unimplemented.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.
...	Further arguments currently unused.

FitDoubleLogGu *Fit a double logistic equation to a vector according to Gu et al. (2003)*

Description

This function fits a double logistic curve to observed values using the function as described in Gu et al. (2003).

Usage

```
FitDoubleLogGu(x, t = index(x), tout = t, hessian=FALSE,  
sf=quantile(x, probs=c(0.05, 0.95), na.rm=TRUE), ...)
```

Arguments

x	A vector or, better, an univariate ts or zoo object.
t	A vector of time (in numeric doys), if not provided index(x) is retrieved.
tout	For gapfilling purposes, a vector of time steps at which the function can be predicted.
hessian	Currently unimplemented.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.
...	Further arguments currently unused.

Details

The function estimates parameters of the double logistic equation from Gu et al. 2003. The wrapper function `GuFit` calls this function and additionally allows the calculation of uncertainty. So better use `GuFit`.

Value

A list containing the following items.

predicted	Predicted values from the equation
params	Equation parameters
formula	The equation

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Gu L, Post WM, Baldocchi D, Black TA, Suyker AE, Verma SB, Vesala T, Wofsy SC. (2009) Characterizing the Seasonal Dynamics of Plant Community Photosynthesis Across a Range of Vegetation Types. In: Phenology of Ecosystem Processes (Ed: Noormets A, Springer New York), pp 35-58.

FitDoubleLogKlHeavy *Fit a double logistic function to a vector according to Klosterman et al. (2014)*

Description

This function fits a double logistic curve to observed values using the function as described in Klosterman et al. (2014), equation 7.

Usage

```
FitDoubleLogKlHeavy(x, t = index(x), tout = t, max.iter = 200,
sf=quantile(x, probs=c(0.05, 0.95), na.rm=TRUE), ...)
```

Arguments

x	A vector or, better, an univariate ts or zoo object.
t	A vector of time (in numeric doys), if not provided index(x) is retrieved.
tout	For gapfilling purposes, a vector of time steps at which the function can be predicted.
max.iter	Maximum number of iteration for a single optimization process. See details.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.
...	Further arguments currently unused.

Details

The function estimates parameters of the double logistic equation from Klosterman et al. 2014. The wrapper function `KlostermanFit` calls this function and additionally allows the calculation of uncertainty. So better use `KlostermanFit`. This function performs an optimization similar in concept to the one performed in `FitDoubleLogKlLight` but with additional recursive optimization which is about 10 times more time consuming but allows for a better representation of the data. It is suggested to fit the light version of the equation and if the fit is not good enough, check this function out.

Value

A list containing the following items.

predicted	Predicted values from the equation
params	Equation parameters
formula	The equation

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Klosterman ST, Hufkens K, Gray JM, Melaas E, Sonnentag O, Lavine I, Mitchell L, Norman R, Friedl MA, Richardson A D (2014) Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using PhenoCam imagery, *Biogeosciences*, 11, 4305-4320, doi:10.5194/bg-11-4305-2014.

See Also

[KlostermanFit](#)

FitDoubleLogKLLight	<i>Fit a double logistic equation to a vector according to Klosterman et al. (2014)</i>
---------------------	---

Description

This function fits a double logistic curve to observed values using the function as described in Klosterman et al. (2014), equation 7.

Usage

```
FitDoubleLogKLLight(x, t = index(x), tout = t, hessian=FALSE,
sf=quantile(x, probs=c(0.05, 0.95), na.rm=TRUE), ...)
```

Arguments

x	A vector or, better, an univariate ts or zoo object.
t	A vector of time (in numeric doys), if not provided index(x) is retrieved.
tout	For gapfilling purposes, a vector of time steps at which the function can be predicted.
hessian	Currently unimplemented.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.
...	Further arguments currently unused.

Details

The function estimates parameters of the double logistic equation from Klosterman et al. 2014. The wrapper function [KlostermanFit](#) calls this function and additionally allows the calculation of uncertainty. So better use [KlostermanFit](#). This function performs an optimization similar in concept to the one performed in [FitDoubleLogK1Heavy](#) but faster and in a less accurate manner. It is suggested to fit the light version of the equation and if the fit is not good enough, check out [FitDoubleLogK1Heavy](#).

Value

A list containing the following items.

predicted	Predicted values from the equation
params	Equation parameters
formula	The equation

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Klosterman ST, Hufkens K, Gray JM, Melaas E, Sonnentag O, Lavine I, Mitchell L, Norman R, Friedl MA, Richardson A D (2014) Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using PhenoCam imagery, *Biogeosciences*, 11, 4305-4320, doi:10.5194/bg-11-4305-2014.

fitted.phenopix *Returns predicted values from phenopix objects.*

Description

Returns predicted values from phenopix objects.

Usage

```
## S3 method for class 'phenopix'  
fitted(object,...)
```

Arguments

object	An object of class phenopix
...	further arguments passed to or from other methods.

Details

Returns predicted values from phenopix objects.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
library(zoo)
data(bartlett2009.filtered)
fitted <- greenProcess(bartlett2009.filtered, 'klosterman',
  'klosterman', plot=FALSE)
plot(fitted$data, type='p')
lines(index(fitted$data), fitted(fitted), col='red')

## End(Not run)
```

get.options	Returns default options for function autoFilter.
-------------	--

Description

Returns default options for function autoFilter.

Usage

```
get.options()
```

Details

Returns default options for function autoFilter. It can be assigned, changed and used within the function autoFilter. In night filter the user can change the threshold of GCC value below which records are discarded. In blue filter the user can change the threshold on daily standard deviation in blue channel above which records are discarded. In mad filter the user can change the z parameter. Increasing z means discarding more data. In max filter the user can change w, i.e. the time window (in days) on which the moving maximum quantile is computed, and qt, the quantile that is used (default 0.9). In spline filter the user can change stdup and stddown, upper and lower standard deviation thresholds, respectively and loop_spline, the number of spline interactions. See examples for details.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
data(bartlett2009)
## with raw.dn ==TRUE and column position for chromatic coordinates
filtered.data <- autoFilter(bartlett2009, dn=c(5:7),
  filter=c('night', 'spline', 'max'),
  filter.options=NULL, raw.dn=TRUE)
my.options <- get.options()
## change time window for max filter
```

```
my.options$max.filter$w <- 5
filtered.data2 <- autoFilter(bartlett2009, dn=c(5:7),
  filter=c('night', 'spline', 'max'),
  filter.options=my.options, raw.dn=TRUE)
plot(filtered.data$max.filtered)
lines(filtered.data2$max.filtered, col='red')

## End(Not run)
```

getCoords	<i>Extract coordinates (in pixels) of the exposure string from a given JPEG image</i>
-----------	---

Description

This function allows to extract coordinates (in pixels) of the exposure string from a given JPEG image

Usage

```
getCoords(image)
```

Arguments

image The absolute path to the JPEG image you want to use.

Details

This function will first plot the JPEG image on screen and call `locator()`. You then have to click first on bottomright corner of the rectangle you want to extract, and then close the polygon (see `?locator()` for details). A second plot with the cropped image is then printed on screen, where you have to click on topleft and bottomright corner around the string "Exposure: xxx", where xxx is the value of exposure. Be also sure to make your rectangle large enough to include possibly four digits exposure values (keep the right margin larger than the actual number). Coords will be returned in x and y pixel positions counted from the topleft corner of the image. These coords will be used to crop the image to extract exactly the Exposure string within the function `getExposure()`. Based on the results of `getExposure()` run on the images, you will evaluate whether to adjust the coordinates and run `getExposure` again, based on how well exposure was recognised from the OCR procedure.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[getExposure](#)

getExposure	<i>Extract exposure from stardot (or similar) images</i>
-------------	--

Description

This function allows to extract exposure from a set of binary images. It is currently optimized for stardot cameras.

Usage

```
getExposure(ipath, coords, train.data=FALSE, date.code, sample=NULL, begin=NULL)
```

Arguments

ipath	The absolute path to a folder of JPEG binary images, as converted from RGB with the function <code>binaryConvert()</code>
coords	A named vector with 4 coordinates (x1, x2, y1, y2), as obtained from <code>getCoords()</code> .
train.data	A named list with ten sample numbers and letter E, as obtained from <code>trainOCR()</code> .
date.code	As in function <code>extractDateFilename()</code> in this package, to convert image names in POSIX vector.
sample	If NULL (default) all images in the folder are processed, otherwise an integer specifying how many files must be processed. This is useful if you want to check some extractions without processing the whole folder.
begin	The user can provide a beginning date as a character string in the form 'YYYY-MM-DD'. Images with a time stamp earlier than the provided date will be discarded. Default is NULL and the whole time series will be processed. Useful when updating an existing time series.

Details

This function performs a simplified OCR procedure to recognize numbers in a binary image. To do so, a full RGB image is first converted in binary (b/w) internally. Then the coordinates to crop the Exposure record are retrieved with `getCoords()` and, finally, the computation is done with `getExposure`. Before doing so you have to train the OCR with sample numbers. See `trainOCR()` for details. The procedure is based on simple matrix matching. The Exposure string is splitted into the digits that constitute it and then each digit is compared to the samples from the training. Finally, a dataframe is extracted with filenames in the first column, exposure in the second one and a POSIX vector in the third.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[getCoords](#)

greenClusters	<i>Performs a cluster analysis on data in output from pixel-based phenology</i>
---------------	---

Description

A function to perform a cluster analysis using function `kmeans` on data in output from pixel-based phenology

Usage

```
greenClusters(data.clusters, data.curve=NULL, nclusters, formula=NULL,
plot=TRUE)
```

Arguments

<code>data.clusters</code>	A numeric data.frame on which the k-means clustering has to be run. Each row stores a various number of phenophases for a given pixel.
<code>data.curve</code>	A numeric data.frame with <code>nrows</code> equal to <code>nrows</code> of <code>data.clusters</code> and <code>ncol</code> equal to the number of parameters in the equation in <code>formula</code> . Each row contains the set of parameters to fit the equation in <code>formula</code> , for a given pixel. Or <code>NULL</code> in case of processing with method 'spline'
<code>nclusters</code>	The number of clusters, as in <code>centers</code> in function <code>kmeans</code> .
<code>formula</code>	An expression of the equation used to fit the data in your spatial analysis. It can be retrieved from the object in output from <code>greenProcess()</code> function. Or <code>NULL</code> in case of processing with method 'spline'
<code>plot</code>	If true a plot with the average trajectories for each cluster is returned. Available only with fitted curves

Details

This function allows to perform a k-means clustering based on an input data.frame containing one or more phenophases extracted from a pixel-based analysis (in the columns) for each pixel analysed by the spatial analysis (each row must have phenophases for each pixel). A number of clusters is identified and each pixel is assigned to one cluster. An approximate average curve for each of the identified clusters is also built by averaging all curve parameters belonging to the same cluster.

Value

A named list with components `curves`, `napos` and `clusters`. "`curves`" contains a zoo object with the average curves, one curve for each of the identified clusters, or NA for spline fitting. "`napos`" is a vector with `na` positions, i.e. pixels where the phenophases in the input matrix are missing. "`clusters`" is a numeric vector of length equal to the number of pixels indicating to which cluster each pixel belongs.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

greenExplore

Explore all possible fitting and threshold methods from the package phenopix

Description

This function allows to fast compute all fitting and thresholding methods on a given greenness time series.

Usage

```
greenExplore(x, ...)
```

Arguments

x	A zoo object with a season of filtered greenness data
...	additional arguments passed to greenProcess

Details

This function runs all possible fitting and thresholding methods. It must be used in combination with plotExplore. The two functions were designed to give a preliminary indication of what would be the best combination of fitting curve and thresholds for your set of data. The function returns also the root mean squared error (RMSE) of the various fits, that can be used as a discriminant to choose the fit.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[plotExplore](#)

Examples

```
## Not run:
data(bartlett2009.filtered)
explored.data <- greenExplore(bartlett2009.filtered)
explored.data$rmse # chech fit RMSES
plotExplore(explored.data) ## an annotated plot

## End(Not run)
```

greenProcess

*Fit a curve and extract thresholds from vegetation index time series***Description**

This is a wrapper function that calls fitting functions and threshold functions and returns an object of class phenopix.

Usage

```
greenProcess(ts, fit, threshold=NULL, plot=TRUE, which='light',
  uncert=FALSE, nrep=100,
  envelope='quantiles', quantiles=c(0.1, 0.9), hydro=FALSE,
  sf=quantile(ts, na.rm=TRUE, prob=c(0.05, 0.95)), ncores='all', ...)
```

Arguments

ts	A vector or, better, an univariate ts or zoo object.
fit	A character vector of length 1. Available options are: spline, beck, elmore, klosterman, gu.
threshold	A character vector of length 1. Available options are: trs, derivatives, klosterman, gu.
plot	If TRUE a call to PhenoPlot returns fitted values and thresholds.
which	Only relevant if fit == klosterman, available options are light or heavy. See FitDoubleLogKlHeavy and FitDoubleLogKlLight for details.
uncert	Should uncertainty be estimated?
nrep	Number of replications for the uncertainty estimation.
envelope	One between quantiles and min-max. If quantiles, the uncertainty envelope will be computed as quantiles. Quantiles reported in quantiles will be computed together with the median. If min-max is chosen, min, max and mean of the uncertainty envelope will be returned.
quantiles	Quantiles to be calculated if envelope='quantiles'. The notation is the same as to specify quantiles in the quantile function, i.e. to get 10th and 90th percentile, use c(0.1, 0.9).
hydro	Hydro determines how days of the year are computed. If hydro = FALSE (default) January 1st is DOY 1. If hydro = TRUE October 1st is day 1. This option has been introduced for two purposes. First, for water limited or high temperature limited boreal ecosystems (with summer dormant season) to process a seasonal trajectory with winter peak. Second, for ecosystems in the australian emisphere. If hydro = TRUE all metrics concerned with a day of year must be back converted in order to get actual day of year. Conversions and back conversions can be performed with the function doy2hydrodoy.
sf	Scaling factors used instead of min and max for data normalization.

ncores	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
...	For the plotting function, a number of parameters from generic plot can be specified. See examples.

Details

This function is a wrapper function that allows to fit a curve to a yearly trajectories of greenness and extract phenological thresholds according to a given criterion. Handling this main function may allow the user to forget learning other, rather internal functions. The combination of greenProcess and extract allows to use main capabilities of this package. Virtually all other functions included are called at some points within this function. The object of class phenopix which is created is a rather complex list that can be explored via the extract function. Check it out for further details.

Value

An object of class phenopix with dedicated functions: plot(), print(), summary() and fitted(). The structure is actually a list.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Gu L, Post WM, Baldocchi D, Black TA, Suyker AE, Verma SB, Vesala T, Wofsy SC. (2009) Characterizing the Seasonal Dynamics of Plant Community Photosynthesis Across a Range of Vegetation Types. In: Phenology of Ecosystem Processes (Ed: Noormets A, Springer New York), pp 35-58.

Klosterman ST, Hufkens K, Gray JM, Melaas E, Sonnentag O, Lavine I, Mitchell L, Norman R, Friedl MA, Richardson A D (2014) Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using PhenoCam imagery, Biogeosciences, 11, 4305-4320, doi:10.5194/bg-11-4305-2014.

Zhang X, Friedl MA, Schaaf CB, Strahler AH, Hodges JCF, Gao F, Reed BC, Huete A (2003) Monitoring vegetation phenology using MODIS, Remote Sens. Environ., 84, 471-475.

Examples

```
## Not run:
data(bartlett2009.filtered)
fitted <- greenProcess(bartlett2009.filtered, 'klosterman',
  'klosterman', uncert=TRUE, nrep=5, ncores=2)

## End(Not run)
```

GuFit	<i>Fit a double logistic equation to a time series according to Gu et al. (2003)</i>
-------	--

Description

This function fits a double logistic curve to observed values using the function as described in Gu et al. (2003).

Usage

```
GuFit(ts, uncert = FALSE, nrep = 100, ncores='all',
      sf=quantile(ts, probs=c(0.05, 0.95), na.rm=TRUE))
```

Arguments

ts	A ts or zoo object with gcc data. <code>index(ts)</code> must be numeric days of year (doys) or a POSIXct vector
uncert	Should uncertainty be estimated?
nrep	Number of replications to estimate uncertainty, defaults to 100.
ncores	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. <code>greenProcess</code> sf is automatically calculated.

Details

The function estimates parameters of the double logistic equation from Gu et al. 2009 and provides an uncertainty estimation. Parameters are estimated by a call to the function `FitDoubleLogGu`. Uncertainty is computed by adding noise to the raw data and by estimating again the parameters. Noise is added according to the standard deviation of the residuals (fitted - observed). The procedure is repeated `nrep` times.

Value

A list containing the following items.

fit	A list as returned by the function <code>FitDoubleLogGu</code>
uncertainty	A list containing a zoo data.frame with the uncertainty predicted values, and a dataframe containing the respective uncertainty curve parameters

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Gu L, Post WM, Baldocchi D, Black TA, Suyker AE, Verma SB, Vesala T, Wofsy SC. (2009) Characterizing the Seasonal Dynamics of Plant Community Photosynthesis Across a Range of Vegetation Types. In: Phenology of Ecosystem Processes (Ed: Noormets A, Springer New York), pp 35-58.

See Also

[FitDoubleLogGu](#)

Examples

```
## Not run:
library(zoo)
data(bartlett2009.filtered)
## fit without uncertainty estimation
fitted.gu <- GuFit(bartlett2009.filtered)
## convert
plot(bartlett2009.filtered)
lines(fitted.gu$fit$predicted, col='red')
## look at fitting parameters
fitted.gu$fit$params
## look at fitting equation, where t is time
fitted.gu$fit$formula

## End(Not run)
```

hydrodoy

Converts from and to hydrological day of year

Description

Converts from and to hydrological day of year, i.e. setting Oct 1st as doy 1.

Usage

```
hydrodoy(x, year, reverse=FALSE)
```

Arguments

x	A numeric vector of doys (if reverse = FALSE) or hydrodoys
year	A character or numeric with a year
reverse	If FALSE (default) a doy vector is expected and an hydrodoy vector is returned, if TRUE, the other way around.

Details

This function converts a vector of doys into hydrodoys, i.e. setting Oct 1st as doy 1. The argument year is used to understand if you are computing hydrodoys in a leap year or not. The back conversion is also possible by setting reverse = TRUE.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

KlostermanFit	<i>Fit a double logistic equation to a vector according to Klosterman et al. (2014)</i>
---------------	---

Description

This function fits a double logistic curve to observed values using the function as described in klosterman et al. (2014), eq 7.

Usage

```
KlostermanFit(ts, which = "light", uncert = FALSE, nrep = 100,
ncores='all', sf=quantile(ts, probs=c(0.05, 0.95), na.rm=TRUE))
```

Arguments

ts	A ts or zoo object with gcc data. index(ts) must be numeric days of year (doys) or a POSIXct vector
which	A character to be chosen between 'light' (default) and 'heavy'. See details.
uncert	Should uncertainty be estimated?
nrep	Number of relications to estimate uncertainty, defaults to 100.
ncores	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.

Details

The function estimates parameters of the double logistic equation from Klosterman et al. 2014. Two optimization procedures are available. If which='light' (the default) equation parameters are optimized using the function optim and computation is faster, whereas if which='heavy' the optimization procedure calls the function nsl and is based on a greater number of iterations with different first guesses for parameters. This option is about ten times slower than the light one.

Value

A list containing the following items.

fit	A list as returned by the function FitDoubleLogGu
uncertainty	A list containing a zoo data.frame with the uncertainty predicted values, and a dataframe containing the respective uncertainty curve parameters

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Klosterman ST, Hufkens K, Gray JM, Melaas E, Sonnentag O, Lavine I, Mitchell L, Norman R, Friedl MA, Richardson A D (2014) Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using PhenoCam imagery, *Biogeosciences*, 11, 4305-4320, doi:10.5194/bg-11-4305-2014.

See Also

[FitDoubleLogKlLight](#) [FitDoubleLogKlHeavy](#)

Examples

```
## Not run:
library(zoo)
data(bartlett2009.filtered)
## fit without uncertainty estimation
fitted.kl1 <- KlostermanFit(bartlett2009.filtered, which='light')
fitted.kl2 <- KlostermanFit(bartlett2009.filtered, which='heavy')
## check fitting
plot(bartlett2009.filtered)
lines(fitted.kl1$fit$predicted, col='red')
lines(fitted.kl2$fit$predicted, col='blue')
legend('topleft',col=c('red', 'blue'), lty=1,
       legend=c('light', 'heavy'), bty='n')

## End(Not run)
```

matchExposure

Match exposures retrieved from the header of (stardot) digital images

Description

This function allows to match exposure values extracted from RGB and IR images based on image timestamp

Usage

```
matchExposure(exposure.df=NULL,
              exposure.RGB=NULL, exposure.IR=NULL, pattern.RGB="RGB",
              pattern.IR="IR", tol=1,
              matching.column='timestamp')
```

Arguments

exposure.df	A dataframe where exposures from RGB and IR images are in the same data.frame. It is however suggested to provide separate data.frames for RGB and IR images. The format of the data.frame must be as in output from the <code>getExposure()</code> function.
exposure.RGB	When exposure.df is NULL, this data.frame, as in output from the function <code>getExposure()</code> , will be used to retrieve RGB exposures
exposure.IR	When exposure.df is NULL, this data.frame, as in output from the function <code>getExposure()</code> , will be used to retrieve IR exposures
pattern.RGB	When exposure.df is non NULL the pattern to be matched in order to subset RGB image rows. The pattern is matched by a call to <code>grep1()</code>
pattern.IR	When exposure.df is non NULL the pattern to be matched in order to subset IR image rows. The pattern is matched by a call to <code>grep1()</code>
tol	A tolerance value to be used to match timestamps of RGB and IR images. Available values include 1 (the default; perfect matching, i.e. the RGB image and the associated IR image has the same timestamp in the filename, rounded to minutes), 10, which means that images recorded within 10 minutes between each other are considered as matched. Higher values are allowed but it is suggested to avoid them
matching.column	A character giving the name of the column that must be used for the matching.

Details

This function is designed to receive in input two data.frames as in output from `getExposure()`, one with RGB exposures and one with IR exposures. Alternatively, one can provide a single data.frame with both RGB and IR exposures, and specify patterns of the image name (column 'images' as output from `getExposure()`) to match the two image types. Duplicates in image names are removed prior to the matching.

Value

A data.frame with exposure values for both RGB and IR images, the original timestamps and the rounded time.stamp. The format is suited to enter `NDVI()` function.

Author(s)

Jeroen Staab <jeroen.staab@posteo.de>, Gianluca Filippa <gian.filippa@gmail.com>

See Also

[getExposure NDVI](#)

NDVI

Compute NDVI from Phenocam color indices

Description

This function computes camera NDVI as in Petach et al. (2014)

Usage

```
NDVI(exposure.matched, RGB.VI, IR.VI, spatial=FALSE)
```

Arguments

<code>exposure.matched</code>	A dataframe as in output from <code>matchExposure()</code>
<code>RGB.VI</code>	A dataframe for RGB VI extracted as in output from <code>extractVIs()</code>
<code>IR.VI</code>	A dataframe for IR VI extracted as in output from <code>extractVIs()</code>
<code>spatial</code>	Set this flag to TRUE if you have extracted VIs with <code>spatial == TRUE</code> .

Details

Compute NDVI as in Petach et al. (2014)

Value

A data.frame with NDVI computed after Petach et al. (2014).

Author(s)

Jeroen Staab <jeroen.staab@posteo.de>, Gianluca Filippa <gian.filippa@gmail.com>

See Also

[getExposure NDVI](#)

PasteSeason *A plotting facility for seasonal data with multiple cycles*

Description

After cutting and processing a timeseries with `CutSeason`, this function allows to plot the extracted thresholds in a single plot.

Usage

```
PasteSeason(x)
```

Arguments

`x` A list of objects as in output from `greenProcess`. All subseasonal fittings must be listed in a list and passed to the present function.

Details

This is a plotting facility which allows to plot in a single graph multiple fitting output from `greenProcess`.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[CutSeason](#)

PhenoBP *Extract breakpoints on a time series*

Description

A function to extract breakpoints and confidence intervals on a time series

Usage

```
PhenoBP(x, breaks=3, confidence=0.95, plot=TRUE, ...)
```

Arguments

`x` An univariate ts or zoo object
`breaks` Maximum number of breaks to be detected
`confidence` The confidence level for the uncertainty computation. Defaults to 0.95, i.e. 95% confidence interval (two-tails).
`plot` Should a diagnostic plot be returned?
`...` Arguments passed to plot.

Details

Threshold extraction is performed according to breakpoint analysis. The function used is `breakpoints` from package `strucchange`. The function also computes a confidence interval by a call to `confint`. Uncertainty analysis is therefore different from the approach used for other thresholds (see e.g. `PhenoGu`). Unlike the other thresholding approaches, `PhenoBP` is born to work with raw data. It therefore does not require fitting an equation. However, the function can also be used with fitted data. See example.

Value

A named vector with extracted breakpoints

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[PhenoExtract](#)

Examples

```
## Not run:
data(bartlett2009.filtered)
breaks <- PhenoBP(bartlett2009.filtered, breaks=4)
mean.breaks.doy <- as.numeric(format(as.POSIXct(t(breaks[2,]),
origin='1970-01-01'), '
mean.breaks.doy

## End(Not run)
```

PhenoDeriv

Extract phenophases from a fitted equation with method Derivatives

Description

Extract phenophases from a fitted equation with method Derivatives

Usage

```
PhenoDeriv(x, formula = NULL, uncert = FALSE,
params = NULL, breaks, ...)
```

Arguments

x	A named vector with the parameters of the equation used to fit the data.
formula	Inherited from previous steps of the process.
uncert	Currently unused
params	Inherited from previous steps of the process.
breaks	Currently unused
...	Further arguments, currently not used

Examples

```
## Not run:
data(bartlett2009.fitted)
derivatives.phenophases <- PhenoDeriv(
x=bartlett2009.fitted$fit$predicted, fit=bartlett2009.fitted$fit
)
plot(bartlett2009.fitted$fit$predicted)
abline(v=derivatives.phenophases[c(1:2,4)], col=palette())
mtext(names(derivatives.phenophases[c(1:2,4)]),
at=derivatives.phenophases[c(1:2,4)],
line=-2,
col=palette()[1:3])

## End(Not run)
```

PhenoExtract

*Generic function for the extraction of phenological thresholds***Description**

This function extracts phenological thresholds according to different methods. Methods include 'trs', 'derivatives', 'klosterman', 'gu'. See details for the computation of each method.

Usage

```
PhenoExtract(data, method = "trs", uncert = FALSE,
breaks = 3, envelope = "quantiles",
quantiles = c(0.1, 0.9), plot = TRUE, sf, ...)
```

Arguments

data	A list structured as in output from the fitting procedures, such as GuFit, KlostermanFit, ElmoreFit, BeckFit.
method	One between 'trs', 'derivatives', 'klosterman', 'gu'.
uncert	Should uncertainty on thresholds be computed? It requires that uncertainty be computed in the fitting function. I.e. The function requires the element 'uncertainty' in data being non NULL. If is.null(uncertainty) in the data or this item is set to FALSE, uncertainty won't be computed.

breaks	Currently unused
envelope	One between 'quantiles' and 'min-max'. If 'quantiles', the uncertainty envelope will be computed as quantiles. Quantiles reported in <code>quantiles</code> will be computed together with the median. If 'min-max' is chosen, min, max and mean of the uncertainty envelope will be returned.
quantiles	Quantiles to be calculated if <code>envelope='quantiles'</code> . The notation is the same as to specify quantiles in the <code>quantile</code> function, i.e. to get 10th and 90th percentile, use <code>c(0.1, 0.9)</code> .
plot	Should a diagnostic plot be returned with annotated thresholds? It calls the function <code>PhenoPlot</code>
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. <code>greenProcess</code> <code>sf</code> is automatically calculated. We suggest using <code>quantile(ts, probs=c(0.05, 0.5))</code> if you need to compute it.
...	For the plotting function, a number of parameters from generic <code>plot</code> can be specified. Additionally, a further option to 'trs' method calling <code>PhenoTrs</code> is provided: by setting a <code>trs</code> argument. The default for <code>trs</code> is 0.5, meaning that the phases <code>sos</code> (start of season) and <code>eos</code> (end of season) will be set when <code>gcc</code> reaches 50% of maximum on the increasing (<code>sos</code>) and decreasing (<code>eos</code>) seasonal trajectory.

Details

This is a wrapper function that calls `PhenoTrs` for `method='trs'`, `PhenoDeriv` for `method='derivatives'`, (from package `greenbrown`) or `PhenoGu` for `method='Gu'`, and `PhenoK1` for `method='klosterman'` from this package. Please see help of the single functions for details on the calculation of thresholds.

Value

If `uncertainty=FALSE` a vector of phenology metrics, otherwise a dataframe.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

- Klosterman ST, Hufkens K, Gray JM, Melaas E, Sonnentag O, Lavine I, Mitchell L, Norman R, Friedl MA, Richardson A D (2014) Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using PhenoCam imagery, *Biogeosciences*, 11, 4305-4320, doi:10.5194/bg-11-4305-2014.
- Gu L, Post WM, Baldocchi D, Black TA, Suyker AE, Verma SB, Vesala T, Wofsy SC. (2009) Characterizing the Seasonal Dynamics of Plant Community Photosynthesis Across a Range of Vegetation Types. In: *Phenology of Ecosystem Processes* (Ed: Noormets A, Springer New York), pp 35-58.
- Zhang X, Friedl MA, Schaaf CB, Strahler AH, Hodges JCF, Gao F, Reed BC, Huete A (2003) Monitoring vegetation phenology using MODIS, *Remote Sens. Environ.*, 84, 471-475.

See Also

[PhenoGu](#), [PhenoK1](#), [PhenoDeriv](#), [PhenoTrs](#), [PhenoPlot](#), [PhenoGu](#)

 PhenoGu

Extract phenophases from a fitted equation according to Gu et al. 2009

Description

A function for the extraction of phenological thresholds according to Gu et al. 2009. This is a rather internal function. Use `PhenoExtract` with `method='gu'` instead.

Usage

```
PhenoGu(x, fit, uncert = FALSE, breaks, sf, ...)
```

Arguments

<code>x</code>	A named vector with the parameters of the equation used to fit the data.
<code>fit</code>	A list structured as in output from the fitting procedures, such as <code>GuFit</code> , <code>KlostermanFit</code> , <code>ElmoreFit</code> , <code>BeckFit</code> .
<code>uncert</code>	Currently unused
<code>breaks</code>	Currently unused
<code>sf</code>	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. <code>greenProcess</code> <code>sf</code> is automatically calculated.
<code>...</code>	Further arguments, currently not used

Details

Threshold extraction is performed according to Gu et al (2009). Briefly, from the fitting equation (be it from Klosterman, Elmore, Beck or Gu fitting) and the correspondent parameters, the first derivative is extracted. Peak recovery rate (pr) is defined as the maximum of the first derivative and represent the maximum slope of the increasing part of the curve. Correspondingly peak senescence rate (psr) is the minimum of the first derivative. The recovery line and the senescence line are tangent to the fitting curve with slope equal to pr and psr respectively. Baseline and maxline are further defined as the horizontal lines corresponding to the minimum and the maximum of the curve. The intersection between recovery line and baseline defines the beginning of the growing season, i.e. upturn date (UD). The intersection between recovery line and maxline defines the reacing of the plateau, i.e. stabilization date (SD). The intersection between senescence line and baseline defines the end of the growing season, i.e. recession date (RD). In the original implementation by Gu et al. the intersection between senescence line and maxline would define the downturn date (oldDD). To account for decreasing plateau typical of a number of seasonal trajectories, we have further defined a plateau line, which is a linear fit between SD and oldDD. The plateau line would roughly correspond to the maxline if the plateau is horizontal, whereas it will be a decreasing line in case of a decreasing plateau. Therefore, in our implementatio DD is defined as the intersection between the plateau line and the senescence line.

Value

A named vector of length 9 with upturn date (UD), stabilizazion date (SD), downturn date (DD), recession date (RD), maximum of the fitting curve (maxline), minimum of the fitting curve (baseline), peack recovery rate (pr), peack senescence rate (psr) and the slope of the plateau line (plateau.slope).

Note

Since this threshold extraction is based on a derivable function, it cannot be performed on raw data. Uncertainty estimation with this method on a fitted curve from SplineFit is currently not implemented. Instead you can use PhenoGu in a for loop cycling in the uncertainty dataframe columns.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

Gu L, Post WM, Baldocchi D, Black TA, Suyker AE, Verma SB, Vesala T, Wofsy SC. (2009) Characterizing the Seasonal Dynamics of Plant Community Photosynthesis Across a Range of Vegetation Types. In: Phenology of Ecosystem Processes (Ed: Noormets A, Springer New York), pp 35-58.

See Also

[PhenoExtract](#)

Examples

```
## Not run:
data(bartlett2009.fitted)
gu.phenophases <- PhenoGu(x=bartlett2009.fitted$fit$params,
fit=bartlett2009.fitted$fit, sf=quantile(bartlett2009.fitted$fit$predicted, c(0.1, 0.9)))
plot(bartlett2009.fitted$fit$predicted)
abline(v=gu.phenophases[1:4], col=palette())
mtext(names(gu.phenophases[1:4]), at=gu.phenophases[1:4],
line=-2:-5, col=palette()[1:4])

## End(Not run)
```

PhenoKl

Extract phenophases from a fitted equation according to Klosterman et al. 2014

Description

A function for the extraction of phenological thresholds according to Klosterman et al. 2014. This is a rather internal function. Use PhenoExtract with method='klosterman' instead.

Usage

```
PhenoKl(x, uncert = FALSE, fit, breaks, ...)
```

Arguments

x	A named vector with the parameters of the equation used to fit the data.
uncert	Currently unused
fit	A list structured as in output from the fitting procedures, such as GuFit, KlostermanFit, ElmoreFit, BeckFit.
breaks	Currently unused
...	Further arguments, currently not used.

Details

Threshold extraction is performed according to Klosterman et al (2014) with a modification derived from Zhang et al (2003). Briefly, the rate of curvature (k) as defined in Klosterman et al (2014) is computed and inflection points are evaluated on its derivative ($derK$). The growing season is splitted in its increasing and decreasing parts around the maximum. The same happens to $derK$. Greenup date is defined as the day of maximum $derK$ (a local maximum) before the first minimum in $derK$ in the increasing part of the curve. Maturity is defined as the maximum in $derK$ between the minimum of $derK$ and mid season. Senescence is defined as the first local minimum in the decreasing part of $derK$. Dormancy is defined as the last local minimum in $derK$. Phases are named after Zhang et al (2003).

Value

A named vector of length 4 with the extracted thresholds: Greenup, Maturity, Senescence, Dormancy.

Note

Since this threshold extraction is based on a derivable function, it cannot be performed on raw data. Uncertainty estimation with this method on a fitted curve from SplineFit is currently not implemented. Instead you can use PhenoKl in a for loop cycling in the uncertainty dataframe columns.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

References

- Klosterman ST, Hufkens K, Gray JM, Melaas E, Sonnentag O, Lavine I, Mitchell L, Norman R, Friedl MA, Richardson A D (2014) Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using PhenoCam imagery, *Biogeosciences*, 11, 4305-4320, doi:10.5194/bg-11-4305-2014.
- Zhang X, Friedl MA, Schaaf CB, Strahler AH, Hodges JCF, Gao F, Reed BC, Huete A (2003) Monitoring vegetation phenology using MODIS, *Remote Sens. Environ.*, 84, 471-475.

See Also[PhenoExtract](#)**Examples**

```
## Not run:
data(bartlett2009.fitted)
klosterman.phenophases <- PhenoKl(
  x=bartlett2009.fitted$fit$params,
  fit=bartlett2009.fitted$fit)
plot(bartlett2009.fitted$fit$predicted)
abline(v=klosterman.phenophases[1:4], col=palette())
mtext(names(klosterman.phenophases[1:4]),
  at=klosterman.phenophases[1:4], line=-2:-5,
  col=palette()[1:4])

## End(Not run)
```

PhenoPlot

*A plotting function for phenological thresholds***Description**

This function uses data from a fitted phenological model, the extracted metrics and plots them in an annotated graph.

Usage

```
PhenoPlot(data, metrics, add = FALSE, show.uncert = TRUE, ...)
```

Arguments

data	Fitted data from a fitting function such as GuFit, SplineFit, KlostermanFit, ElmoreFit, BeckFit
metrics	A named vector or a dataframe, depending on the presence or absence of uncertainty estimation
add	If TRUE the plot is superimposed to an existing one
show.uncert	Should uncertainty be shown as error bars around the extracted thresholds? It requires that data in input incorporate the uncertainty estimation.
...	Several argument as in plot can be specified here, see example.

Details

This function allows to plot fitting and thresholding on a season of gcc data. Uncertainty can be also shown

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[PhenoExtract](#)

Examples

```
## Not run:
data(bartlett2009.fitted)
klosterman.phenophases <- PhenoExtract(bartlett2009.fitted,
method='klosterman', uncert=TRUE, plot=FALSE)
PhenoPlot(bartlett2009.fitted, klosterman.phenophases$unc.df)

## End(Not run)
```

PhenoTrs

A phenophase method based on fixed thresholds

Description

See PhenoTrs from package greenbrown for details.

Usage

```
PhenoTrs(x, approach = c("White", "Trs"), trs = 0.5, min.mean = 0.1,
formula = NULL, uncert = FALSE, params = NULL, breaks, ...)
```

Arguments

x	seasonal cycle of one year
approach	approach to be used to calculate phenology metrics. 'White' (White et al. 1997) or 'Trs' for simple threshold.
trs	threshold to be used for approach "Trs"
min.mean	minimum mean annual value in order to calculate phenology metrics. Use this threshold to suppress the calculation of metrics in grid cells with low average values
formula	Returned from previous steps of the process
uncert	Currently unused
params	Returned from previous steps of the process
breaks	Currently unused
...	further arguments (currently not used)

Examples

```
## Not run:
data(bartlett2009.fitted)
trs.phenophases <- PhenoTrs(
  x=bartlett2009.fitted$fit$predicted,
  fit=bartlett2009.fitted$fit)
plot(bartlett2009.fitted$fit$predicted)
abline(v=trs.phenophases[c(1:2,4)], col=palette())

## End(Not run)
```

plot.phenopix	<i>Plotting phenopix objects.</i>
---------------	-----------------------------------

Description

Plotting method for objects of class phenopix

Usage

```
## S3 method for class 'phenopix'
plot(x, y, what, main, ...)
```

Arguments

x	An object of class phenopix
y	It must be NULL
what	A character vector of length 1. Available options are: all (showing both fitting curve and thresholds), fitting (showing only fitting curve) thresholds, and params to show boxplots of thresholds and curve parameters (if appropriate) when the uncertainty is computed.
main	A main title for the plot. If not specified the title will be build with fit name and threshold name.
...	For the plotting function, a number of parameters from generic plot can be specified. Note that graphic properties of fitted lines and thresholds cannot be modified. See examples.

Details

A dedicated plotting function for objects of class phenopix. The default shows observed values as a grey line, the fitted function in black and extracted thresholds in palette() colors. If available, uncertainty is also shown with a family of lightgrey curves and as error bars on the extracted thresholds.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
data(bartlett2009.filtered)
fitted <- greenProcess(bartlett2009.filtered, 'klosterman',
  'gu', plot=FALSE, uncert=TRUE, nrep=5, ncores=2)
plot(fitted) ## default
## slightly more elaborated, with suppression of default title
plot(fitted, type='p', pch=20, ylab='GCC', xlab='DOY', col='grey',
  what='all', main='Fit + thresholds')
## only with the fitting curve
plot(fitted, type='p', pch=20, ylab='GCC', xlab='DOY', col='grey',
  what='fitting', main='Fit only')
## show a boxplot of thresholds extracted from the uncertainty estimation
plot(fitted, what='thresholds')

## End(Not run)
```

plotBP

A function to plot results of function PhenoBP

Description

A function to plot results of function PhenoBP, with confidence intervals and annotated break points

Usage

```
plotBP(ts, breaks, bp.y, ...)
```

Arguments

ts	An univariate zoo series with measured or modelled Gcc data
breaks	An object as in output from PhenoBP function.
bp.y	Optional argument indicating the y coordinate where break point names are written, defaults to <code>quantile(ts, 0.005)</code>
...	Further arguments of plot function.

Details

A function to rapidly plot results from PhenoBP function

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

plotExplore	<i>Plot an object in output from the greenExplore function</i>
-------------	--

Description

This function allows to plot the object in output from the greenExplore() function.

Usage

```
plotExplore(x)
```

Arguments

x An object (a list) in output from greenExplore

Details

The combination of functions greenExplore and plotExplore are intended to provide a first view of all possible fitting and thresholding methods on a season of greenness data. In particular plotExplore combines all fittings and thresholding methods (20 plots) on an annotated graph.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[greenExplore](#)

Examples

```
## Not run:
data(bartlett2009.filtered)
explored.data <- greenExplore(bartlett2009.filtered)
explored.data$rmse # check fit RMSES
plotExplore(explored.data) ## an annotated plot

## End(Not run)
```

plotSpatial	<i>Plot pixel-based analysis results from spatialGreen and extractParameters functions</i>
-------------	--

Description

This function allows to plot results from pixel-based analysis of a seasonal ensemble of images in one or multiple rois.

Usage

```
plotSpatial(data, param, roi.data.path, image.path,
  probs=c(0.01, 0.99), ...)
```

Arguments

data	Either a data.frame or a list of dataframes
param	Character string of length 1 with name of the parameter you want to plot. It must match one of names(data)
roi.data.path	The complete path where to find roi data (e.g. '/home/me/folderROI/roi.data.RData')
image.path	A complete path of an image you want to use as background for plotting your data. E.g. '/home/me/images/my_site201406061200.jpg'
probs	Two numbers in the range [0,1] passed to quantile() function to remove tails of the distribution for the plot.
...	Additional arguments passed to the function plot

Details

This function allows to plot on a reference image results from the pixel based extraction of thresholds and other relevant parameters from pixel based fitting. Multiple ROIs are allowed and handled as well. On top of the plot the density distribution of value for each pixel is shown, if desired. Values are plotted in a black to green scale, with a legend. Data in entrance to this function can be stored in a data.frame (single ROI data) as in output from extractParameters. If you have analysed more than one ROI and you want them to be shown together, put individual data.frames in output from extractParameters in a list and pass the list as data to plotSpatial.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

`plotSum`*A plotting facility to show the combination of uncertainty*

Description

The `combineUncertainty` uses [greenProcess](#) to fit all available double logistic equations in the `phenopix` package and extracts thresholds with all available methods. Then uncertainties can be combined and returned by using [summarizePhases](#) and plotted with [plotSum](#). See [greenProcess](#).

Usage

```
plotSum(ts, sum, which, v=NULL, quantile=TRUE, ...)
```

Arguments

<code>ts</code>	A <code>ts</code> or <code>zoo</code> object with <code>gcc</code> data. <code>index(ts)</code> must be numeric days of year (doys) or a <code>POSIXct</code> vector
<code>sum</code>	An object in output from <code>summarizePhases</code>
<code>which</code>	One between <code>trs</code> , <code>derivatives</code> , <code>klosterman</code> , <code>gu</code>
<code>v</code>	An optional vector of vertical coordinates (in y-axis unit for plot annotation of phase names)
<code>quantile</code>	If <code>TRUE</code> , the plotted uncertainty envelope is based on the quantiles, and not min-max, otherwise min-max envelope is plotted
<code>...</code>	For the plotting function, a number of parameters from generic <code>plot</code> can be specified. Note that graphic properties of fitted lines and thresholds cannot be modified. See examples.

Details

This function is the last step of a processing chain. It uses [greenProcess](#) to fit all available double logistic equations in the `phenopix` package and extracts thresholds with all available methods. Then uncertainties can be combined and returned by using [summarizePhases](#) and plotted with [plotSum](#). See [greenProcess](#), [summarizePhases](#), [plotSum](#). This function uses a model approach to combine all uncertainties from all available `phenopix` fittings, as to get an ensemble of phases with different methods, without necessarily choosing any of them.

Value

A named list with dataframes for each phenophase method with all replication for each of the included fitting methods. These data can then be combined with the companion functions [summarizePhases](#) and [plotSum](#). See examples for details.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
require(zoo)
data(bartlett2009.filtered)
combined.fit <- combineUncertainty(na.approx(filtered.tmp$max.filtered), nrep=100)
# 100 replications for each fitting
names(combined.fit) # a dataframe for each phenoMethod + a list with all fittings
fit.summary <- summarizePhases(combined.fit, across.methods=TRUE)
## again a list with one element for each fitting method + two additional items
## if across.methods is TRUE, which combines gu + klosterman phenophase methods
## in a single method, and the same happens for trs and derivatives
plotSum(bartlett2009.filtered, fit.summary, which='klosterman')
## a plot with original timeseries + phenophases and their uncertainty

## End(Not run)
```

plotVI

Plot RGB DN, RGB indices and brightness

Description

This function allows to reproduce the same plot as returned from extractVIs

Usage

```
plotVI(VI.data, VI.path)
```

Arguments

VI.data	An object as in output from extractVIs (with spatial == FALSE)
VI.path	The path where plots will be saved

Details

Using in input a VI.data object as in output from extractVIs, this function returns a plot similar to the default plot of extractVIs. This function makes it easy to update VI plots after using extractVIs with option begin==TRUE.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

print.phenopix	<i>Print phenopix objects.</i>
----------------	--------------------------------

Description

Print method for objects of class phenopix

Usage

```
## S3 method for class 'phenopix'  
print(x, ...)
```

Arguments

x	An object of class phenopix
...	further arguments passed to or from other methods.

Details

Prints a sintetic summary of the object.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:  
data(bartlett2009.filtered)  
fitted <- greenProcess(bartlett2009.filtered, 'klosterman',  
  'klosterman', plot=FALSE)  
print(fitted)  
  
## End(Not run)
```

PrintROI	<i>Print an already drawn region of interest</i>
----------	--

Description

This function allows to re-draw one (or more) region(s) of interest (ROI) on a jpeg image.

Usage

```
PrintROI(path_img_ref,path_ROIs,which='all',col, file.type='.jpg')
```

Arguments

path_img_ref	Path in your folder for the reference image
path_ROIs	Path where ROI coordinates are stored in a structured list called ROI.Rdata.
which	Which one of your ROIs you want to draw, defaults to ‘all’ and draws all rois.
col	A character vector with colors. If missing it defaults to color palette
file.type	It must match the syntax of your file extension (e.g. .jpg, .JPG, .JPEG). Multiple types are allowed by concatenation with c().

Details

This utility allows to draw your ROIs on your reference image, with different colors for different ROIS and a dedicated legend See DrawROI for details.

Value

It returns a plot that can be saved with your favorite device.

Author(s)

Gianluca Filippa <g.filippa@arpa.vda.it>

resizeImage	<i>Resize an image (and a tROI) to a given pixel resolution</i>
-------------	---

Description

This function allows to resize a sample image and a correspondent ROI to a given pixel resolution to be used as background to spatila analysis plots.

Usage

```
resizeImage(image.in, image.out, roi.in, roi.out, npixels)
```

Arguments

image.in	The complete path to your original image
image.out	The complete path (filename with extension included) where the new, resized image will be saved.
roi.in	The complete path to your original roi.data
roi.out	The complete path (filename with extension included) where the new, resized roi.data.RData will be saved
npixels	As in extractVIs to aggregate more than one pixel

Details

Coupled with spatial analysis and image resize (see `extractVIs()` and specifically argument `npixels` for details), this function allows to update a selected image and the correspondent ROI to a smaller size. This is exactly what is done internally when `extractVIs()` is called with `npixels` different from 1. The resized image can be used (together with the `roi.data` object) to plot results from spatially explicit phase extraction. See the vignette 'spatial' for illustrated examples.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

spatialFilter	<i>Explore all possible fitting and threshold methods from the package phenopix</i>
---------------	---

Description

This function allows to fast compute all fitting and thresholding methods on a given greennes time series.

Usage

```
spatialFilter(spatial.list, filter=c('night','spline', 'max'),
  filter.options=NULL,
  ncores='all',log.file=NULL, NDVI=FALSE)
```

Arguments

<code>spatial.list</code>	An object in output from function <code>extractVIs()</code> with <code>spatial = TRUE</code> .
<code>filter</code>	Character names for filters. Chose one (or more) between 'night', 'max', 'spline', 'blue' and 'mad'. The order provided in this argument determines the order of application of filters. See details for further information on filters.
<code>filter.options</code>	If NULL <code>filter.options</code> defaults to set values, otherwise they must be specified in a named list. Filter options are designed to work with a wide range of data, the user must change them with caution.
<code>ncores</code>	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
<code>log.file</code>	It can be NULL or a path where to generate and refresh a txt file which logs the progress of the filtering procedure
<code>NDVI</code>	If TRUE filter is applied to NDVI values, data must be formatted differently from usual filtering.

Details

This function performs the same task of `autoFilter` but in a pixel-by-pixel analysis

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[autoFilter](#)

spatialGreen

Process pixel-based greenness indexes

Description

This function allows to filter, fit a curve and extract thresholds in a pixel-based analysis exactly as `autoFilter` and `greenProcess` do in a ROI-based analysis, except that uncertainty cannot be estimated (since it would be too computationally intense)

Usage

```
spatialGreen(filtered.data, fit, threshold, ncores='all',
             log.file=NULL)
```

Arguments

<code>filtered.data</code>	A list as in output from <code>spatialFilter()</code> .
<code>fit</code>	A character vector of length 1. Available options are: <code>spline</code> , <code>beck</code> , <code>elmore</code> , <code>klosterman</code> , <code>gu</code> .
<code>threshold</code>	A character vector of length 1. Available options are: <code>spline</code> , <code>derivatives</code> , <code>klosterman</code> , <code>gu</code> .
<code>ncores</code>	Number of processors to be used in parallel computation, defaults to 'all' which will accidentally slow down any other activity on your computer. Otherwise set the number of processors you want to use in parallelization.
<code>log.file</code>	It can be <code>NULL</code> or a path where to generate and refresh a txt file which logs the progress of the filtering procedure

Details

This function allows to fit a curve and extract thresholds in a pixel-based analysis exactly as `greenProcess` does in a ROI-based analysis, except that uncertainty cannot be estimated (since it would be too computationally intense). This function takes as first argument a list as in output from `spatialFilter`. For each pixel in the ROI the function fits a curve (according to options specified in `fit`) and extracts thresholds (as defined in `threshold`). This function performs the same task that `greenProcess` does in a ROI-based analysis, except that uncertainty cannot be estimated (since it would be too computationally intense). For pixel-based analysis, it is recommended to use rather low resolution images or split your region of interest into multiple subROIs (function `splitROI`). A specific vignette for spatial analysis is stored as pdf in the package folder. The user is advised to carefully read it before starting a spatial analysis.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

SplineFit

A function to fit a smoothed spline to Phenological Data

Description

A function to fit a smoothed spline to Phenological Data

Usage

```
SplineFit(ts, uncert = FALSE, nrep = 100, df.factor = 0.05,
ncores='all', sf=quantile(ts, probs=c(0.05, 0.95), na.rm=TRUE))
```

Arguments

ts	A ts or zoo object with gcc data. index(ts) must be numeric days of year (doys)
uncert	Should uncertainty be estimated?
nrep	Number of relications to estimate uncertainty, defaults to 100.
df.factor	Defaults to 0.05, it is multiplied by length(ts) to generate degrees of freedom for the spline fitting. The higher the number of data, the higher should be df factor. For a complete year of data (i.e. length(ts)=365) the default value is optimum.
ncores	Unused argument for compatibility
sf	Scaling factors required to normalize the data prior to the fitting. If the function is called by e.g. greenProcess sf is automatically calculated.

Details

This function fits a smoothed spline to the data. Df for smoothing are set at 0.05*length(ts) by default and df.factor can be modified. Uncertainty is estimated by changing the degrees of freedom of the spline. In particular a sequence from 0.01 and df.factor, of length nrep is used as varying degrees of freedom for the spline fitting.

Value

A list containing the following items.

fit	A list with fitted values and an object named 'params' set to NULL, for simmetry with other fittings
uncertainty	A list containing a zoo data.frame with the uncertainty predicted values, and an object named 'params' set to NULL, for simmetry with other fittings

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
data(bartlett2009.filtered)
fitted <- SplineFit(bartlett2009.filtered, uncert=TRUE, nrep=50)

## End(Not run)
```

splitROI*Splits a ROI into multiple subrois for spatial analysis*

Description

This function allows to split a roi into different subrois for a pixel-based analysis.

Usage

```
splitROI(roi.data, nsplit, names=NULL)
```

Arguments

roi.data	An object with roi data as in output from DrawROI() function
nsplit	The number of sub-rois in which your ROI must be divided into
names	Optional names for roi. If omitted rois will be progressively numbered (i.e, roi1, roi2, roi3, etc.)

Details

This function allows to split a large ROI into smaller ones. This is needed to make computationally feasible the pixel-based analysis with large ROIs and/or high resolution images. As a general indication, approx 10000 pixels per sub-ROI is the upper limit for a computationally feasible spatial analysis, so set nsplit argument accordingly. The number of pixels in a ROI essentially controls the dimension of the VI.data object you generate with extractVIs function. A large VI.data object (typically larger than 200 Mb) will fast saturate your RAM and slow down the spatial analysis.

Value

A names list of length = nsplit and where each element has the same structure as a roi.data extracted by DrawROI.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

structureFolder	<i>Fast structuration of a folder for phenopix process</i>
-----------------	--

Description

This function allows to create subfolders to a folder optimized for the phenopix processing. If assigned to an object, all subfolders can be accessed in a simple and straightforward way.

Usage

```
structureFolder(path, ...)
```

Arguments

path	The main path where the subdirectories will be created
...	further arguments passed to <code>dir.create</code>

Details

This function creates 4 subfolders to a given folder. In IMG you will store images to be processed. In REF you will put your reference image, i.e. the one that you will use to draw your ROI(s). In ROI the RData containing ROI data and one plot for each ROI will be saved. In folder VI you can save your VI.data.Rdata after have run `extractVIs`.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

summarizePhases	<i>An evolution of greenProcess for the combination of uncertainty after processing</i>
-----------------	---

Description

The `combineUncertainty` uses [greenProcess](#) to fit all available double logistic equations in the phenopix package and extracts thresholds with all available methods. Then uncertainties can be combined and returned by using [summarizePhases](#) and plotted with [plotSum](#). See [greenProcess](#).

Usage

```
summarizePhases(list, quantiles=c(0.25, 0.75), across.methods=FALSE)
```

Arguments

<code>list</code>	An object in output from <code>combineUncertainty</code>
<code>quantiles</code>	The envelope used for the uncertainty
<code>across.methods</code>	If TRUE, <code>gu</code> and <code>klosterman</code> phenophase methods are combined in a single phenophase estimation, and the same happens for derivatives and <code>trs</code> .

Details

This function uses [greenProcess](#) to fit all available double logistic equations in the `phenopix` package and extracts thresholds with all available methods. Then uncertainties can be combined and returned by using [summarizePhases](#) and plotted with [plotSum](#). See [greenProcess](#), [summarizePhases](#), [plotSum](#). This function uses a modellistic approach to combine all uncertainties from all available `phenopix` fittings, as to get an ensemble of phases with different methods, without necessarily choosing any of them.

Value

A named list with dataframes for each phenophase method with all replication for each of the included fitting methods. These data can then be combined with the companion functions [summarizePhases](#) and [plotSum](#). See examples for details.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
require(zoo)
data(bartlett2009.filtered)
combined.fit <- combineUncertainty(na.approx(filtered.tmp$max.filtered), nrep=100)
# 100 replications for each fitting
names(combined.fit) # a dataframe for each phenoMethod + a list with all fittings
fit.summary <- summarizePhases(combined.fit, across.methods=TRUE)
## again a list with one element for each fitting method + two additional items
## if across.methods is TRUE, which combines gu + klosterman phenophase methods
## in a single method, and the same happens for trs and derivatives
plotSum(bartlett2009.filtered, fit.summary, which='klosterman')
## a plot with original timeseries + phenophases and their uncertainty

## End(Not run)
```

summary.phenopix	<i>Summary of phenopix objects.</i>
------------------	-------------------------------------

Description

Summary method for objects of class phenopix

Usage

```
## S3 method for class 'phenopix'  
summary(object, ...)
```

Arguments

object	An object of class phenopix
...	further arguments passed to or from other methods.

Details

Prints a sintetic summary of the object.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:  
data(bartlett2009.filtered)  
fitted <- greenProcess(bartlett2009.filtered, 'klosterman',  
  'klosterman', uncert=TRUE, plot=FALSE, nrep=5, ncores=2)  
summary(fitted)  
  
## End(Not run)
```

trainOCR	<i>Extract a training dataset for OCR procedure</i>
----------	---

Description

This function allows to extract a training dataset for OCR procedure performed by getExposure. It is currently optimized for stardot cameras.

Usage

```
trainOCR(image.path, nsamples=100)
```

Arguments

image.path	The absolute path to a folder of JPEG binary images, as converted from RGB with the function <code>binaryConvert()</code>
nsamples	The maximum number of sampled images to be used. No need to change it.

Details

This function allows to prepare a training dataset that will be used in function `getExposure()`. You need to identify 0-9 numbers and the capital letter E (Exposure), which are then used in `getExposure()`. The procedure makes use of locator to subsequently crop your image, so make sure you know how this function works in your OS. When you run the function a first image pops up on your graphic device. You have to click with the mouse on topleft and bottom right of the rectangle you want to crop. I suggest to crop to the entire string of text with all picture information, so to have the largest sample of numbers in it. When you close locator (right-click in Linux-OS, but likely also GUI-dependent), the cropped image will show up providing a zoom to the selection. The title in the plot helps you to remember which numbers you still have to define. Choose a number (the order you choose numbers does not matter) and make a second crop around it (always topleft, bottomright). Close locator. A third zoom will show up, gridded pixel by pixel. Again, crop your number topleft bottomright with a rectangle that exactly includes all pixels of your number. Close locator. In R command line you will be asked to type the number you have just drawn, or letter E. Type the number and press Enter. You will be prompted to a new image where you follow the procedure again to identify other numbers. When you will be done with all numbers and E letter, you will get a named list with 11 elements. Each element will be a binary matrix for each of your numbers, and letter E.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

See Also

[getCoords](#)

update.phenopix *Update phenopix objects.*

Description

Update method for objects of class `phenopix`

Usage

```
## S3 method for class 'phenopix'  
update(object, threshold, envelope, quantiles, uncert, plot, ...)
```

Arguments

object	An object of class phenopix
threshold	One between 'spline', 'derivatives', 'klosterman', 'gu'
envelope	If left blank is recycled from original fitting. See PhenoExtract
quantiles	If left blank is recycled from original fitting. See PhenoExtract
uncert	If left blank is recycled from original fitting. See PhenoExtract
plot	If left blank is recycled from original fitting. See PhenoExtract
...	Plotting arguments. See PhenoExtract

Details

This function allows to update a phenopix object in output from greenProcess to extract different thresholds without refitting the data (which is highly time-consuming when uncertainty is computed). All arguments except threshold may be left blank and will be recycled from the original fit. But the can also be changed. See PhenoExtract where arguments are described in detail.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Examples

```
## Not run:
data(bartlett2009.filtered)
fitted <- greenProcess(bartlett2009.filtered, 'klosterman',
  'klosterman', plot=FALSE)
fitted.gu <- update(fitted, 'gu')

## End(Not run)
```

updateROI

Update pixels included in a ROI of different pixel size

Description

This function allows to apply an already drawn ROI to the same image but with different resolution.

Usage

```
updateROI(old.roi, new.img)
```

Arguments

old.roi	The roi object
new.img	An image array as read by readJPEG

Details

This function allows to extract pixel information in a ROI on images of different quality with respect to the one where the ROI was first drawn. The typical use of this function is to work on resized (smaller) images. Typically you draw a ROI on your best quality image (say 640 x 480 pixels) and process your images to extract vegetation indexes. But if you then want to use the function `spatialGreen` to perform a pixel based analysis you have to reduce image size because the analysis is computationally extremely intense. See `spatialGreen` for more details on computation time. Hence, you typically degradate your image (say to 320 x 240 pixels). The function applies ROI vertices to the new, resized image and returns pixels in the ROI of the new image size.

Author(s)

Gianluca Filippa <gian.filippa@gmail.com>

Index

- * **datasets**
 - bartlett2009, 7
 - bartlett2009.filtered, 7
 - bartlett2009.fitted, 8
 - bartlett2009.processed, 8
- * **package**
 - phenopix-package, 3
- autoFilter, 5, 58
- bartlett2009, 7
- bartlett2009.filtered, 7
- bartlett2009.fitted, 8
- bartlett2009.processed, 8
- BeckFit, 9
- combineUncertainty, 10
- confint, 41
- convert, 11
- convertROI, 12, 14
- CutSeason, 13, 40
- DrawMULTIROI, 12, 14
- editExposure, 15
- ElmoreFit, 16
- extract, 17
- extractDateFilename, 18
- extractParameters, 19
- extractVIs, 20
- FitDoubleLogBeck, 9, 21
- FitDoubleLogElmore, 16, 17, 22
- FitDoubleLogGu, 23, 34, 35
- FitDoubleLogKlHeavy, 24, 37
- FitDoubleLogKlLight, 25, 37
- fitted.phenopix, 26
- get.options, 27
- getCoords, 28, 29, 64
- getExposure, 15, 28, 29, 39
- greenClusters, 30
- greenExplore, 31, 51
- greenProcess, 9–11, 16, 22–25, 32, 34, 36, 40, 43, 44, 53, 59, 61, 62
- GuFit, 34
- hydrodoy, 35
- KlostermanFit, 25, 36
- matchExposure, 37
- NDVI, 39, 39
- PasteSeason, 14, 40
- PhenoBP, 40
- PhenoDeriv, 41, 43, 44
- PhenoExtract, 41, 42, 45, 47, 48
- PhenoGu, 41, 43, 44, 44
- PhenoKl, 43, 44, 45
- phenopix (phenopix-package), 3
- phenopix-package, 3
- PhenoPlot, 44, 47
- PhenoTrs, 43, 44, 48
- plot.phenopix, 49
- plotBP, 50
- plotExplore, 31, 51
- plotSpatial, 52
- plotSum, 10, 11, 53, 53, 61, 62
- plotVI, 54
- print.phenopix, 55
- PrintROI, 55
- resizeImage, 56
- spatialFilter, 57
- spatialGreen, 58
- SplineFit, 59
- splitROI, 60
- structureFolder, 61
- summarizePhases, 10, 11, 53, 61, 61, 62

summary.phenopix, [63](#)

trainOCR, [63](#)

update.phenopix, [64](#)

updateROI, [65](#)