Package: phenex (via r-universe)

October 26, 2024

Type Package

Title Auxiliary Functions for Phenological Data Analysis

Version 1.4-5

Date 2017-05-24

Author Maximilian Lange, Daniel Doktor

Maintainer Daniel Doktor <daniel.doktor@ufz.de>

Description Provides some easy-to-use functions for spatial analyses of (plant-) phenological data sets and satellite observations of vegetation.

Depends R (>= 3.0)

Imports methods, foreach, DEoptim

License GPL (>= 2)

NeedsCompilation yes

Date/Publication 2017-05-29 11:35:40 UTC

Repository https://daniel-dok.r-universe.dev

RemoteUrl https://github.com/cran/phenex

RemoteRef HEAD

RemoteSha 1d9577f124b041c44e5e15c2687351bfad7ef12d

Contents

`analyzeBits` *Bit Analysis*

Description

Analyses bits of a 16-bit integer

Usage

```
analyzeBits(value, mode=1, bitpos=0)
```

Arguments

`value`	A 16-bit integer value
`mode`	If mode is 0, the algorithm returns the bit on position 'bitpos'. If mode is 1 (default), the algorithm returns the most significant bit, if mode is 2 it returns the second significant bit and if mode is 3 the return value is the integer value of the last four bits
`bitpos`	An integer value between 0 and 15 determining the position of the bit to read when mode is 0.

Details

This routine analyses 16-bit integer values to get the indicators of MEDOKADS data

Value

An integer value respective to parameter 'mode'.

Author(s)

Daniel Doktor, Maximilian Lange

References

Koslowsky, D., Billing, H. and Friedrich, K. (2005): MEDOKADS: A long-term data set for detection and monitoring for desertification risks in the mediterranean. In *RGLDD Conference*.

Examples

```
value <- -32768
res <- analyzeBits(value, mode=3)
res
```

---

avhrr *AVHRR Daily Dataset*

Description

This data set gives an example of daily NDVI data from the MEDOKADS data sets.

Usage

```
avhrr
```

Format

A vector containing 365 NDVI values from satellite observations.

Source

MEDOKADS Data Set

References

Koslowsky, D., Billing, H. and Friedrich, K. (2005): MEDOKADS: A long-term data set for detection and monitoring for desertification risks in the mediterranean. In *RGLDD Conference*.

---

avhrrcomp *Example of AVHRR Composite Data*

Description

This data set gives an example of composite NDVI data from the MEDOKADS data sets.

Usage

```
avhrrcomp
```

Format

A vector containing 36 NDVI values from satellite observations..

Source

MEDOKADS Data Set

References

Koslowsky, D., Billing, H. and Friedrich, K. (2005): MEDOKADS: A long-term data set for detection and monitoring for desertification risks in the mediterranean. In RGLDD Conference.

`bise` *Best index slope extraction*

Description

Reduces noise in NDVI time-series. Second interpretation of bise algorithm.

Usage

```
bise(x, slidingperiod, growthFactorThreshold, cycleValues)
```

Arguments

`x`	An object of class 'NDVI' containing raw NDVI values.
`slidingperiod`	Sliding Period of the BISE-algorithm, default value is 40.
`growthFactorThreshold`	Maximum allowed increase per day as factor, default value is 0.1 (increase of 10 percent).
`cycleValues`	A boolean value determining whether the end of the ndvi timeserie is combined with its beginning or not (default value is true). If false, gaps or low ndvi values at the beginning of timeserie influence the resulting timeserie.

Details

Knowledge regarding the phenological cycle in temperate climates and its temporal evolution is used to detect and eliminate cloud contaminated observations. As the algorithm is searching forward within daily NDVI observations over 1 year, decreases are only accepted if no higher value is found within a so called sliding period. A period of 40 days proved best for our study area but might have to be modified when study areas in different climates are investigated.

Value

An object of class 'NDVI' containing raw and corrected NDVI values.

Author(s)

Daniel Doktor, Maximilian Lange

References

Viovy, N., Arino, O. and Belward, A.S. (1992). The Best Index Slope Extraction (BISE) - a method for reducing noise in NDVI time-series. *International Journal of Remote Sensing*, **13**, 1585-1590.

See Also

modelNDVI, NDVI

Examples

```
# load data
data(avhrr)

# create NDVI object
ndvi <- new("NDVI", values=avhrr.ndvi/10000, year=as.integer(1995))

# correct values (bise)
ndvi.bise <- bise(ndvi, slidingperiod=40, growthFactorThreshold=0.1)

#plot
plot(ndvi.bise)
```

---

correctedValues *Corrected Value Accessor*

Description

Access to corrected values of NDVI object.

Usage

```
correctedValues(x)
```

Arguments

x	An object of class 'NDVI' containing raw and corrected NDVI values.

Value

Returns a vector containing corrected NDVI values.

Author(s)

Lange, Maximilian and Doktor, Daniel

See Also

NDVI, modelNDVI, bise, runningAvg

Examples

```
# load data
data(avhrr)

# create NDVI object, correct and model values
ndvi.list <- modelNDVI(ndvi.values=avhrr.ndvi/10000, year.int=1995,
  correction="bise", method="LinIP", MARGIN=2,
  doParallel=FALSE, slidingperiod=40)
ndvi <- ndvi.list[[1]]

#get modelled values
biseValues <- correctedValues(ndvi)
```

---

date2doy *Date to Julian Day Converter*

Description

Converts a date into a Julian day

Usage

```
date2doy(date)
```

Arguments

date	Date YYMMDD as integer

Value

The Julian Day (day of year) of the date

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
date <- 891208
doy <- 0
doy <- date2doy(date)
doy
```

`detectSeasons` *Season Detection*

Description

Detects seasons in timeseries.

Usage

```
detectSeasons(x, minValRange, ...)
```

Arguments

`x`	An object of class 'NDVI' containing raw NDVI values.
`minValRange`	Range in which to search for lowest bise value around detected season start.
`...`	Optional parameters passed to internal bise call.

Value

An object of class 'NDVI', containing "seasons".

Author(s)

Daniel Doktor, Maximilian Lange

See Also

`NDVI`, `seasons`, `bise`

Examples

```
# load data
data(avhrr)

# generate ndvi object
ndvi <- new("NDVI", values=rep(avhrr.ndvi/10000,5), year=NA)

# extract seasons
ndvi <- detectSeasons(ndvi)
seasons(ndvi)
```

`integrateTimeserie` *Integration of timeserie*

Description

Extracts the integral of the vegetation index between start and end date.

Usage

```
integrateTimeserie(x, start, end, n)
```

Arguments

`x`	An object of class 'NDVI' containing modelled NDVI values.
`start`	A list containing the starting date(s) for integration as 'mean' and its standard deviation(s) as 'sd'. Use a list with multiple entries (as vector) for 'mean' and 'sd' if the NDVI object contains multiple seasons.
`end`	A list containing the end date(s) for integration as 'mean' and its standard deviation(s) as 'sd'. Use a list with multiple entries (as vector) for 'mean' and 'sd' if the NDVI object contains multiple seasons.
`n`	The number 'n' of normal distributed values to create around start and end date.

Value

A list containing the integral(s) as 'mean' and a standard devation(s) 'sd'.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

`NDVI`, `integrate`

Examples

```
# load data
data(avhrr)

# create NDVI object, correct and model NDVI values
ndvi <- modelNDVI(ndvi.values=avhrr.ndvi/10000, year.int=1995,
   correction="bise", method="LinIP", MARGIN=2,
   doParallel=FALSE, slidingperiod=40)[[1]]

# extract greenup DOY
greenup <- phenoPhase(ndvi, phase="greenup", method="local", threshold=0.55, n=1000)
senesc <- phenoPhase(ndvi, phase="senescence", method="local", threshold=0.55, n=1000)
```

```
# extract green season integrated vegetation index
gsivi <- integrateTimeserie(ndvi, start=greenup, end=senesc, n=1000)
```

`leapYears` *Leap Year Check*

Description

Checks whether the given years are leap years.

Usage

```
leapYears(year)
```

Arguments

`year`	A vector of years as integer values.

Value

A vector of boolean values determining whether the given years are leap years.

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
year <- c(1995, 2000, 2005, 2010)
leaps <- leapYears(year)
leaps
```

`modelledValues` *Modelled Value Accessor*

Description

Access to modelled values of NDVI object.

Usage

```
modelledValues(x)
```

Arguments

`x`	An object of class 'NDVI' containing raw and modelled NDVI values.

Value

Returns a vector containing modelled NDVI values.

Author(s)

Lange, Maximilian and Doktor, Daniel

See Also

NDVI, modelNDVI, modelValues

Examples

```
# load data
data(avhrr)

# create NDVI object, correct and model values
ndvi.list <- modelNDVI(ndvi.values=avhrr.ndvi/10000, year.int=1995,
  correction="bise", method="LinIP", MARGIN=2,
  doParallel=FALSE, slidingperiod=40)
ndvi <- ndvi.list[[1]]

#get modelled values
model <- modelledValues(ndvi)
```

---

modelNDVI *Main function for NDVI correction and modelling*

Description

Fits a suite of functions/models to raw NDVI or selected NDVI observations (after BISE).

Usage

```
modelNDVI(ndvi.values, year.int, multipleSeasons=FALSE, correction="bise",
 method="LinIP", MARGIN=2, doParallel=FALSE, silent=TRUE, ...)
```

Arguments

ndvi.values	A vector or matrix containing NDVI observations with values in the interval (-1,1).
year.int	Observation year
multipleSeasons	Determines wether a season detection should be performed or not. Setting 'multipleSeasons' to TRUE forces following algorithms modelling NDVI and extracting phenological phases to be performed once per detected season.

`correction` A character string determining which method will be used for correction of NDVI values. Should be either:

"**none**": no correction will be performed.

"**ravg**": Smoothing with running average. Default window size is 7 and can be modified by parameter 'window.ravg'. See also `runningAvg`.

"**bise**": Best index slope extraction after *Viovy et. al* (1992). Second interpretation of bise algorithm. Can be modified with parameter 'slidingperiod'. Default to 40, optimised for the area of Central Europe. This might has to be modified when vegetation dynamics of other climates/habitats are analysed. An maximum increase threshold is defined via parameter 'growthFactorThreshold' with default to 0.1 (10 percent increase per day allowed). The timeserie is cycled via parameter 'cycleValues', which is a boolean value determining whether the end of the ndvi timeserie is combined with its beginning or not (default value is true). If false, gaps or low ndvi values at the beginning of timeserie influence the result.

`method` Determines which model will be fitted to the corrected NDVI-timeserie (if the corrected timeserie is not available, the raw one out of 'values' will be used).

"**LinIP**": A linear interpolation is performed. For interpolation, the end of timeserie is connected to the beginning (e.g. after day 365 follows day 1). Applied in *Badeck et. al* (2004) and *Doktor et. al* (2009).

"**Spline**": A spline interpolation is performed. For interpolation, the end of timeserie is connected to the beginning (e.g. after day 365 follows day 1).

"**DSig**": Fits a double sigmoidal function to NDVI values (according to Zhang et. al (2003)).

"**DSigC**": Fits another double sigmoidal function (own C implementation) to NDVI values.

"**DLogistic**": Fits a double logistic function after *Fischer, Alberte* (1994) to NDVI values.

"**Gauss**": Fits a symmetric or asymmetric (determined by boolean parameter 'asym') gaussian function to NDVI values (own C implementation after *Press, W.H.* (1992)).

"**GaussMix**": Fits a mixture of gaussian functions to NDVI values (own C implementation after *Press, W.H.* (1992)). The number of components is determined by parameter 'components'. If multiple components are given, the algorithm checks which number performs best.

"**Growth**": Fits a plant growth model after *Richter et. al* (1991) to NDVI values.

"**FFT**": Smoothes the corrected or raw NDVI values with fast fourier transformation (implemented in R). The smoothing intensity can be controlled with parameter 'filter.threshold' with default to 3.

"**SavGol**": Smoothes the corrected or raw NDVI values with a Savitzky-Golay filter (own C implementation after *Press, W.H.* (1992)). The smoothing algorithm can be modified with parameters 'window.sav' (window size of filter, default to 7), 'degree' (degree of fitting polynomial, default to 2) and 'smoothing' (repetition quantity, default to 10).

`MARGIN` A vector giving the subscripts which the function will be applied over. E.g., for a matrix '1' indicates rows, '2' indicates columns, 'c(1, 2)' indicates rows and

	columns. Where 'X' has named dimnames, it can be a character vector selecting dimension names. Should be '2' if 'ndvi.values' is a vector instead of a matrix/array.
`doParallel`	This method uses 'foreach'. If a parallel backend is registered, setting 'do.parallel' to 'TRUE' enables parallel processing.
`silent`	A boolean flag determining whether debug information is shown.
`...`	Other parameters passed to correction or modelling function. These are: 'slidingperiod' for correction "bise", 'window.ravg' for correction "ravg", 'asym' for method "Gauss", 'filter.threshold' for method "FFT" and 'degree', 'window.sav' and 'smoothing' for method "SavGol".

Value

Returns an object of type 'NDVI' containing raw data, corrected NDVI values and modelled NDVI values.

Author(s)

Lange, Maximilian and Doktor, Daniel

References

Badeck, F.W., Bondeau, A., Boettcher, K., Doktor, D., Lucht, W., Schaber, J. and Sitch, S. (2004). Responses of spring phenology to climate change. *New Phytologist*, **162**, 295-309.

Doktor, D., Bondeau, A., Koslowski, D. and Badeck, F.W. (2009). Influence of heterogeneous landscapes on computed green-up dates based on daily AVHRR NDVI observations. *Remote Sensing of Environment*, **113**, 2618-2632

Fischer, Alberte (1994). A Model for the Seasonal Variations of Vegetation Indices in Coarse Resolution Data and Its Inversion to Extract Crop Parameters. *Remote Sensing of Environment*, **48**, 220-230.

Press, W.H. (1992). Numerical recipes in C: The Art of Scientific Computing, vol. 1. Cambridge University Press, Cambridge, 2nd edn.

Richter, O., Spickermann, U. and Lenz, F. (1991). A new model for plant-growth. *Gartenbauwissenschaft*, **56**, 99-106.

Viovy, N., Arino, O. and Belward, A.S. (1992). The Best Index Slope Extraction (BISE) - a method for reducing noise in NDVI time-series. *International Journal of Remote Sensing*, **13**, 1585-1590.

Zhang, X.Y., Friedl, M.A., Schaaf, C.B., Strahler, A.H., Hodges, J.C.F., Gao, F., Reed, B.C. and Huete, A. (2003). Monitoring vegetation phenology using MODIS. *Remote Sensing of Environment*, **84**, 471-475.

See Also

`bise`, `runningAvg`,`detectSeasons`, `NDVI`, `phenoPhase`

Examples

```
data(avhrr)
data(modis)

# create NDVI object, correct and model values
ndvi.list1 <- modelNDVI(ndvi.values=cbind(avhrr.ndvi/10000, modis.ndvi/10000),
   year.int=1995, multipleSeasons=FALSE, correction="bise",
   method="LinIP", MARGIN=2, doParallel=FALSE, slidingperiod=40)
ndvi.list2 <- modelNDVI(ndvi.values=cbind(avhrr.ndvi/10000, modis.ndvi/10000),
   year.int=1995, multipleSeasons=FALSE, correction="ravg",
   method="FFT", MARGIN=2, doParallel=FALSE, filter.threshold=7)

# plot Values
for (ndvi.ob in ndvi.list1){ plot(ndvi.ob) }
for (ndvi.ob in ndvi.list2){ plot(ndvi.ob) }
```

`modelValues` *NDVI modelling*

Description

Models NDVI values.

Usage

```
modelValues(x, method, ...)
```

Arguments

`x`	An object of class 'NDVI' containing raw and/or corrected NDVI values.
`method`	Determines which model will be fitted to the corrected NDVI-timeserie (if the corrected timeserie is not available, the raw one out of 'values' will be used). "**LinIP**": A linear interpolation is performed. For interpolation, the end of timeserie is connected to the beginning (e.g. after day 365 follows day 1). Applied in *Badeck et. al* (2004) and *Doktor et. al* (2009). "**Spline**": A spline interpolation is performed. For interpolation, the end of timeserie is connected to the beginning (e.g. after day 365 follows day 1). "**DSig**": Fits a double sigmoidal function to NDVI values (according to *Zhang et. al* (2003)). "**DSigC**": Fits another double sigmoidal function (own C implementation) to NDVI values. "**DLogistic**": Fits a double logistic function after *Fischer, Alberte* (1994) to NDVI values. "**Gauss**": Fits a symmetric or asymmetric (determined by boolean parameter 'asym') gaussian function (own C implementation after *Press, W.H.* (1992)) to NDVI values.

"**Growth**": Fits a growth model after *Richter et. al* (1991) to NDVI values.

"**FFT**": Smoothes the corrected or raw NDVI values with fast fourier transfusion (implemented in R). The smoothing intensity can be controlled with parameter 'filter.threshold' with default to 3.

"**SavGol**": Smoothes the corrected or raw NDVI values with a Savitzky-Golay filter (own C implementation after *Press, W.H.* (1992)). The smoothing algorithm can be modified with parameters 'window' (window size of filter, default to 7), 'degree' (degree of fitting polynomial, default to 2) and 'smoothing' (repetition quantity, default to 10).

... Other parameters passed to modelling function. 'asym' for method "Gauss", 'filter.threshold' for method "FFT" and 'degree', 'window' and 'smoothing' for method "SavGol".

Details

Returns an object of type 'NDVI' containing raw data and/or corrected NDVI values and modelled NDVI values.

Author(s)

Lange, Maximilian and Doktor, Daniel

References

Badeck, F.W., Bondeau, A., Boettcher, K., Doktor, D., Lucht, W., Schaber, J. and Sitch, S. (2004). Responses of spring phenology to climate change. *New Phytologist*, **162**, 295-309.

Doktor, D., Bondeau, A., Koslowski, D. and Badeck, F.W. (2009). Influence of heterogeneous landscapes on computed green-up dates based on daily AVHRR NDVI observations. *Remote Sensing of Environment*, **113**, 2618-2632

Fischer, Alberte (1994). A Model for the Seasonal Variations of Vegetation Indices in Coarse Resolution Data and Its Inversion to Extract Crop Parameters. *Remote Sensing of Environment*, **48**, 220-230.

Press, W.H. (1992). Numerical recipes in C: The Art of Scientific Computing, vol. 1. Cambridge University Press, Cambridge, 2nd edn.

Richter, O., Spickermann, U. and Lenz, F. (1991). A new model for plant-growth. *Gartenbauwissenschaft*, **56**, 99-106.

Viovy, N., Arino, O. and Belward, A.S. (1992). The Best Index Slope Extraction (BISE) - a method for reducing noise in NDVI time-series. *International Journal of Remote Sensing*, **13**, 1585-1590.

Zhang, X.Y., Friedl, M.A., Schaaf, C.B., Strahler, A.H., Hodges, J.C.F., Gao, F., Reed, B.C. and Huete, A. (2003). Monitoring vegetation phenology using MODIS. *Remote Sensing of Environment*, **84**, 471-475.

See Also

NDVI, modelNDVI

Examples

```
# load data
data(avhrr)

# create NDVI object
ndvi <- new("NDVI", values=avhrr.ndvi/10000, year=as.integer(1995))

# correct values (bise)
ndvi <- bise(ndvi, slidingperiod=40)

#model values
ndvi <- modelValues(ndvi, method="LinIP")

# plot
plot(ndvi)
```

`modis` *MODIS Daily Dataset*

Description

This data set gives an example of daily NDVI data from the MOD09 data sets.

Usage

```
modis
```

Format

A vector containing 365 NDVI values based on daily observations of surface reflectances obtained from satellite 'MODIS Terra' with a spatial resolution of 250m.

Source

MOD09GQ Data Set

References

Vermote, E. and Kotchenova, S. (2008): MOD09 (Surface Reflectance) User's Guide. MODIS Land Surface Reflectance Science Computing Facility, 1st Edition.

modiscomp *MODIS Composite Dataset*

Description

This data set gives an example of composite NDVI data from the MOD13 data sets.

Usage

```
modiscomp
```

Format

A vector containing 365 NDVI values based on observations of surface reflectances with 250m spatial and 16 days temporal resoltion obtained from satellite 'MODIS Terra'.

Source

MOD13Q1 Data Set

References

Didan, K., Huete, A., Jacobson, A. and Solano1, R. (2010): MODIS Vegetation Indices (MOD13) C5 User's Guide. Terrestrial Biophysics and Remote Sensing Lab, The University of Arizona. 1st Edition.

NDVI-class *Class "NDVI"*

Description

Class 'NDVI' provides functions to smooth NDVI (Normalized Difference Vegetation Index) time-series obtained from satellite observations.

Objects from the Class

NDVI-Objects can be created by calls of the form `modelNDVI(ndvi.values, ...)`. These objects contain the timeserie as given by 'ndvi.values', smoothed values due to the performed correction (see also `modelNDVI`, parameter 'correction') and modelled values due to the chosen method (see also `modelNDVI`, parameter 'method'). Furthermore, the year of the timeserie can be stored as integer value.

Slots

`year`: The year of the timeseries as integer value or 'NA'.

`seasons`: Season start positions of the timeseries.

`values`: A vector of NDVI-values (between minus one and one as numeric) of length 365 or 366 (respective to length of the year).

`correctedValues`: A vector of corrected NDVI-values. Is set after use of correction methods 'bise' or 'runningAvg'.

`modelledValues`: A vector of corrected NDVI-values. Is set after use of method 'modelValues'.

Methods

bise `signature(x = "NDVI")`: Best index slope extraction after *Viovy et. al* (1992). Second interpretation of algorithm.

This routine tries to restore the temporal NDVI profile, i.e. separate true observations from noise. The sliding period default of BISE-algorithm is 40 days, optimised for the area of Central Europe. This might has to be modified when vegetation dynamics of other climates/habitats are analysed. An maximum increase threshold is defined via parameter 'growthFactorThreshold' with default to 0.1 (10 percent increase per day allowed). The timeserie is cycled via parameter 'cycleValues', which is a boolean value determining whether the end of the ndvi timeserie is combined with its beginning or not (default value is true). If false, gaps or low ndvi values at the beginning of timeserie influence the result.

This method is used by function `modelNDVI` if parameter correction is set to "bise".

checkLength `signature(x = "NDVI")`: Checks the length of the timeserie respective to the length of the year (365 days or 366 for leap years).

correctedValues<- `signature(x = "NDVI")`: Replacement method for slot `correctedValues`.

correctedValues `signature(x = "NDVI")`: Acessor method for slot `correctedValues`.

isLeapYear `signature(x = "NDVI")`: Checks whether the year of the NDVI-object is a leap year.

seasons<- `signature(x = "NDVI")`: Replacement method for slot `seasons`.

seasons `signature(x = "NDVI")`: Acessor method for slot `seasons`.

modelledValues<- `signature(x = "NDVI")`: Replacement method for slot `modelledValues`.

modelledValues `signature(x = "NDVI")`: Acessor method for slot `modelledValues`.

modelValues `signature(x = "NDVI")`: Fits a suite of functions/models to corrected NDVI-timeserie (if the corrected timeserie is not available, the raw one out of 'values' will be used). Parameter 'method' determines the used model:

"**LinIP**": A linear interpolation is performed. For interpolation, the end of timeserie is connected to the beginning (e.g. after day 365 follows day 1). Applied in *Badeck et. al* (2004) and *Doktor et. al* (2009).

"**Spline**": A spline interpolation is performed. For interpolation, the end of timeserie is connected to the beginning (e.g. after day 365 follows day 1).

"**DSig**": Fits a double sigmoidal function to NDVI values (according to *Zhang et. al* (2003)).

"**DSigC**": Fits another double sigmoidal function (own C implementation) to NDVI values.

"**DLogistic**": Fits a double logistic function after *Fischer, Alberte* (1994) to NDVI values.

"**Gauss**": Fits a symmetric or asymmetric (determined by boolean parameter 'asym') gaussian function to NDVI values (own C implementation after *Press, W.H.* (1992)).

"**GaussMix**": Fits a mixture of gaussian functions to NDVI values (own C implementation after *Press, W.H.* (1992)). The number of components is determined by parameter 'components'. If multiple components are given, the algorithm checks which number performs best.

"**Growth**": Fits a growth model after *Richter et. al* (1991) to NDVI values.

"**FFT**": Smoothes the corrected or raw NDVI values with fast fourier transfusion (implemented in R). The smoothing intensity can be controlled with parameter 'filter.threshold' with default to 3.

"**SavGol**": Smoothes the corrected or raw NDVI values with a Savitzky-Golay filter (own C implementation after *Press, W.H.* (1992)). The smoothing algorithm can be modified with parameters 'window.sav' (window size of filter, default to 7), 'degree' (degree of fitting polynomial, default to 2) and 'smoothing' (repetition quantity, default to 10).

This method is used by function `modelNDVI` with modelling method respective to parameter 'method'.

phenoPhase `signature(x = "NDVI")`: Extracts the start of phenological phases out of the modelled NDVI values. Parameter 'phase' determines which phase is extracted:

"**max**": Day of the year with highest NDVI value is returned.

"**min**": Day of the year with lowest NDVI value is returned.

"**greenup**": Day of the year at which greenup takes place is returned. This day refers to the point where the function of modelled NDVI values exceeds a certain threshold.

"**senescence**": Day of the year at which senescence takes place is returned. This day refers to the point where the function of modelled NDVI values exceeds a certain threshold.

Parameter 'method' determines whether a global or local threshold is used for greenup and senescence extraction:

With "**global**" threshold, the day of the year is returned, where NDVI values are first equal or higher as the value of 'threshold'. If the threshold is higher than the values of the timeseries, '-1' will be returned.

With "**local**" threshold, the day of the year is returned, for which NDVI values first reach the value of 'threshold' (interpreted as percentage) between lowest and highest NDVI value of timeserie. The lowest NDVI value is chosen depending on phase selected. For "greenup", the lowest value before day of maximum NDVI value is used. For "senescence", the lowest value after day of maximum NDVI value is used.

The 'threshold' for local or global greenup/senescence-extraction method should have numerical value between '0' and '1'. Global thresholds refer to a fix NDVI value independent of actual NDVI magnitudes. Local thresholds are based on (max - min) * e.g. 0.55. Hence, the respective NDVI value will vary dependent on e.g. land-cover.

plot `signature(x = "NDVI")`: Plots raw data as black dots (slot 'values'), corrected values as red dots (slot 'correctedValues') and modelled values as blue line (slot 'modelledValues').

runningAvg `signature(x = "NDVI")`: This routine performes an averaging with a running window on NDVI values. Default window size is 7 and can be modified by parameter 'window'.

This method is used by function `modelNDVI` if parameter correction is set to "ravg". See also `runningAvg`.

values<- `signature(x = "NDVI")`: Replacement method for slot `values`.

values `signature(x = "NDVI")`: Acessor method for slot `values`.

year<- `signature(x = "NDVI")`: Replacement method for slot `year`.

year `signature(x = "NDVI")`: Acessor method for slot `year`.

rsquare `signature(x = "NDVI")`: Calculates the squared Pearson correlation coefficient out of corrected (if the corrected timeserie is not available, the raw one out of 'values' will be used) and modelled timeserie.

integrateTimeserie `signature(x = "NDVI")`: Calculates the integral of the timeserie between the bounds 'start' and 'end'.

Author(s)

Lange, Maximilian and Doktor, Daniel

References

Badeck, F.W., Bondeau, A., Boettcher, K., Doktor, D., Lucht, W., Schaber, J. and Sitch, S. (2004). Responses of spring phenology to climate change. *New Phytologist*, **162**, 295-309.

Doktor, D., Bondeau, A., Koslowski, D. and Badeck, F.W. (2009). Influence of heterogeneous landscapes on computed green-up dates based on daily AVHRR NDVI observations. *Remote Sensing of Environment*, **113**, 2618-2632

Fischer, Alberte (1994). A Model for the Seasonal Variations of Vegetation Indices in Coarse Resolution Data and Its Inversion to Extract Crop Parameters. *Remote Sensing of Environment*, **48**, 220-230.

Press, W.H. (1992). Numerical recipes in C: The Art of Scientific Computing, vol. 1. Cambridge University Press, Cambridge, 2nd edn.

Richter, O., Spickermann, U. and Lenz, F. (1991). A new model for plant-growth. *Gartenbauwissenschaft*, **56**, 99-106.

Viovy, N., Arino, O. and Belward, A.S. (1992). The Best Index Slope Extraction (BISE) - a method for reducing noise in NDVI time-series. *International Journal of Remote Sensing*, **13**, 1585-1590.

Zhang, X.Y., Friedl, M.A., Schaaf, C.B., Strahler, A.H., Hodges, J.C.F., Gao, F., Reed, B.C. and Huete, A. (2003). Monitoring vegetation phenology using MODIS. *Remote Sensing of Environment*, **84**, 471-475.

See Also

`bise`, `runningAvg`, `modelNDVI`, `phenoPhase`

Examples

```
 ##first example
# load data
data(avhrr)
data(modis)

# create NDVI object
ndvi1 <- new("NDVI", values=avhrr.ndvi/10000, year=as.integer(1995))
ndvi2 <- new("NDVI", values=modis.ndvi/10000, year=as.integer(1995))
```

```
# correct values (bise)
ndvi1.bise <- bise(ndvi1)
ndvi2.bise <- bise(ndvi2)

# or running Average
ndvi1.ravg <- runningAvg(ndvi1)
ndvi2.ravg <- runningAvg(ndvi2)

# model Values
res1 <- modelValues(ndvi1.bise, method="LinIP")
res2 <- modelValues(ndvi1.ravg, method="FFT", filter.threshold=7)
res3 <- modelValues(ndvi2.bise, method="LinIP")
res4 <- modelValues(ndvi2.bise, method="FFT", filter.threshold=7)

# plot Values
plot(res1)
plot(res2)
plot(res3)
plot(res4)

# extract greenup date
greenup <- phenoPhase(res1, phase="greenup", method="local", threshold=0.55, n=1000)
# extract date with highest ndvi
max.ndvi <- phenoPhase(res1, phase="max", n=1000)

## or simpler
data(avhrr)
data(modis)

# create NDVI objects, correct and model values
ndvi.list1 <- modelNDVI(ndvi.values=cbind(avhrr.ndvi/10000, modis.ndvi/10000),
  year.int=1995, correction="bise", method="LinIP", MARGIN=2,
  doParallel=FALSE, slidingperiod=40)
ndvi.list2 <- modelNDVI(ndvi.values=cbind(avhrr.ndvi/10000, modis.ndvi/10000),
  year.int=1995, correction="ravg", method="FFT", MARGIN=2,
  doParallel=FALSE, filter.threshold=7)

# plot Values
for (ndvi.ob in ndvi.list1){ plot(ndvi.ob) }
for (ndvi.ob in ndvi.list2){ plot(ndvi.ob) }

# extract greenup date
greenup <- phenoPhase(ndvi.list1[[1]], phase="greenup",
  method="local", threshold=0.55, n=1000)
# extract senescence date
senescence <- phenoPhase(ndvi.list1[[1]], phase="senescence",
  method="local", threshold=0.55, n=1000)
# extract date with highest NDVI
max.ndvi <- phenoPhase(ndvi.list1[[1]], phase="max", n=1000)

# calculate green season integrated vegetation index
gsivi <- integrateTimeserie(ndvi.list1[[1]], start=greenup, end=senescence, n=1000)
```

phenoPhase *Phenological Phase Extraction*

Description

Extracts phenological metrics based on modelled NDVI values.

Usage

```
phenoPhase(x, phase, method, threshold, n)
```

Arguments

x	An object of class 'NDVI' containing modelled NDVI values.
phase	Determines which phase will be extracted: "**max**": Day of the year with highest NDVI value is returned. "**maxval**": The highest modelled NDVI value and its standard deviation is returned. "**min**": Day of the year (before day with maximum NDVI value) with lowest NDVI value is returned. "**minval**": The lowest modelled NDVI value and its standard deviation is returned. "**greenup**": Day of the year at which greenup takes place and its standard deviation is returned. This day refers to the point where the function of modelled NDVI values exceeds a certain threshold. "**senescence**": Day of the year at which senescence takes place and its standard deviation is returned. This day refers to the point where the function of modelled NDVI values exceeds a certain threshold.
method	Determines whether a global or local threshold is used for greenup and senescence extraction. "**global**" threshold: The day of the year is returned, where NDVI values are first equal or higher as the value of 'threshold'. If the threshold is higher than the values of the timeseries, '-1' will be returned. "**local**" threshold: The day of the year is returned, for which NDVI values first reach the value of 'threshold' (interpreted as percentage) between lowest and highest NDVI value of timeserie. The lowest NDVI value is chosen depending on phase selected. For "greenup", the lowest value before day of maximum NDVI value is used. For "senescence", the lowest value after day of maximum NDVI value is used.
threshold	Threshold for local or global greenup/senescence-extraction method. Should have numerical value between '0' and '1'. Global thresholds refer to a fix NDVI value independent of actual NDVI magnitudes. Local thresholds are based on (max - min) * e.g. 0.55. Hence, the respective NDVI value will vary dependent on e.g. land-cover.

n — The number 'n' of normal distributed values around the threshold for estimation of 'sd'. The normal distribution uses the threshold as mean and a standaed deviation consisting of satellite error and standard deviation of fit residuals. The satellite error is considered as 0.02+0.02*value.

Value

A list containing the julian day of the year at which the phenologial phase occurs as list entry 'mean' and its standard deviation as list entry 'sd'. The list contains vectors in 'mean' and 'sd' if multiple seasons are available in 'NDVI' object.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

NDVI, detectSeasons, seasons

Examples

```
# load data
data(avhrr)

# create NDVI object, correct and model NDVI values
ndvi <- modelNDVI(ndvi.values=avhrr.ndvi/10000, year.int=1995,
   correction="bise", method="LinIP", MARGIN=2,
   doParallel=FALSE, slidingperiod=40)[[1]]

# extract greenup DOY
greenup <- phenoPhase(ndvi, phase="greenup", method="local", threshold=0.55, n=1000)
```

---

rsquare *Squared Pearson correlation coefficient*

Description

Calculates the squared Pearson correlation coefficient of corrected and modelled timeserie.

Usage

```
rsquare(x)
```

Arguments

x — An object of class 'NDVI' containing corrected and modelled values. If corrected timeserie is not available, the raw one out of 'values' will be used

Value

The squared Pearson correlation coefficient as numeric value.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

NDVI

Examples

```
# load data
data(avhrr)

# create NDVI object, correct and model values
ndvi.list <- modelNDVI(ndvi.values=avhrr.ndvi/10000,
  year.int=1995, correction="bise", method="Growth", MARGIN=2,
  doParallel=FALSE, slidingperiod=40)

#plot
plot(ndvi.list[[1]])

# squared Pearson correlation coefficient
rsquare(ndvi.list[[1]])
```

---

runningAvg *Running Average*

Description

Reduces noise in NDVI time-series through running averaging.

Usage

```
runningAvg(x, window)
```

Arguments

x	An object of class 'NDVI' containing raw NDVI values.
window	Window size of the running averaging algorithm. Default is 7.

Value

An object of class 'NDVI' containing raw and corrected NDVI values.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

modelNDVI,NDVI

Examples

```
# load data
data(avhrr)

# create NDVI object
ndvi <- new("NDVI", values=avhrr.ndvi/10000, year=as.integer(1995))

# correct values (bise)
ndvi.ravg <- runningAvg(ndvi, window=7)

#plot
plot(ndvi.ravg)
```

---

seasons *Seasons Accessor*

Description

Access to detected seasons.

Usage

```
seasons(x)
```

Arguments

x	An object of class 'NDVI' containing raw NDVI values and detected seasons.

Value

Returns a vector containing position of seasons.

Author(s)

Lange, Maximilian and Doktor, Daniel

See Also

NDVI, detectSeasons

Examples

```
# load data
data(avhrr)

# create NDVI object, correct and model values
ndvi.list <- modelNDVI(ndvi.values=c(avhrr.ndvi/10000,avhrr.ndvi/10000),
  year.int=1995, detectSeasons=TRUE,
  correction="bise", method="LinIP", MARGIN=2,
  doParallel=FALSE, slidingperiod=40)
ndvi <- ndvi.list[[1]]

#get seasons
seasondates <- seasons(ndvi)
```

`values` *Raw Value Accessor*

Description

Access to raw values of NDVI object.

Usage

```
values(x)
```

Arguments

`x`	An object of class 'NDVI' containing raw NDVI values.

Value

Returns a vector containing raw NDVI values.

Author(s)

Lange, Maximilian and Doktor, Daniel

See Also

NDVI

Examples

```
# load data
data(avhrr)

# create NDVI object
ndvi <- new("NDVI", values=avhrr.ndvi/10000, year=as.integer(1995))

#extract raw values
rawValues <- values(ndvi)
```

`yearlength` *Number of Days*

Description

Returns the number of days of the given year.

Usage

```
yearlength(year)
```

Arguments

`year`	A vector of years as integer values.

Value

The number of days of the year in the date

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
year <- c(1995, 2000, 2005, 2010)
days <- yearlength(year)
days
```

Index