Package 'phenmod'

February 20, 2015

Type Package

Title Auxiliary functions for phenological data processing, modelling and result handling

Version 1.2-3

Date 2013-08-20

Author Maximilian Lange

Maintainer Maximilian Lange <maximilian.lange@ufz.de>

Description Provides functions to preprocess phenological data, for modelling and result handling.

Depends R (>= 2.14), gstat, RColorBrewer, lattice, pheno

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-08-20 14:56:42

R topics documented:

---

bbObs *Phenological observation: budburst*

Description

This dataset gives an example of budburst observations from the German Weather Service (DWD) obtainable by the plant phenological online database PPODB. The dataset was extracted out of the downloadable tsv-File by use of `data.read.phase`.

Usage

```
bbObs
```

Format

A dataframe containing three observations.

Source

Plant Phenological Online Database (PPODB), `www.phenology.de`

References

Dierenbach, J., Badeck, F.W., Schaber, J., 2013. The plant phenological online database (PPODB): an online database for long-term phenological data. International Journal of Biometeorology , 1-8.

See Also

`data.read.phase`,`lcObs`

`data.addTemperatures` *Add temperature data to a dataset.*

Description

Adds a temperature vector to a dataset containing spatial and seasonal phenological information.

Usage

```
data.addTemperatures(dataset, grid.related.to.Temperatures,
temperature.filenames,
temperature.matrix, temperature.scale.factor,
out2File=FALSE, silent=FALSE)
```

Arguments

`dataset`	A dataset containing spatial and seasonal phenological information. Can be created by using function `data.extract`.
`grid.related.to.Temperatures`	A grid containing spatial informations for the temperature data.
`temperature.filenames`	The full name of temperature files. The filenames have to contain the year (YYYY) of the temperature observation. The files should be stored as RData-Files and have to contain a matrix called 'edk.one.year' with 366 columns (one per day) and the number of rows equal to the number of rows in 'grid.related.to.Temperatures'. Should have the value NULL if a temperature matrix should be used instead.
`temperature.matrix`	An array containing temperature data. The year of the observation should given as rowname, the columns should equal the julian day of the observation and the third dimension of the array should equal the location given in 'grid.related.to.Temperatures'. The matrix will be used instead of temperature files if 'temperature.filenames' is NULL.
`temperature.scale.factor`	The down-scaling factor for the temperature data (needed if the data is scaled).
`out2File`	A boolean value determining wether the output will be stored in log-files.
`silent`	A boolean value determining wether the function should generate output messages or not.

Details

Adds a temperature vector to a dataset containing spatial (The coordinates of the station as Gauss-Krueger coordinates) and seasonal (The year for which the budburst day should be modelled and the leafcolouring day of the previous year) phenological information.

Value

A dataset containing spatial and seasonal phenological data and the added temperatures.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

data.loadTemperature,data.extract,relatedGrid,temperatures

Examples

```
  ## load extracted observations as created by 'data.extract'
  data(extractedObs)
  ## load temperature data
  data(temperatures)
  ## load grid with spatial informations for temperature files
  data(relatedGrid)

  ## add temperatures from files in working directory
  dataset <- data.addTemperatures(dataset=extractedObs,
out2File=FALSE,
grid.related.to.Temperatures=relatedGrid,
temperature.filenames=NULL,
temperature.matrix=temperatures,
temperature.scale.factor=0.1)

  ## resulting dataset should equal 'dataFinal'
  data(dataFinal)
  dataFinal
  dataset
```

---

data.combine *Main function to combine timeseries*

Description

This function creates a station net and builds clusters of stations out of it. These clusters are used to create combined timeseries.

Usage

```
data.combine(dataset, range=5000, alt.range=50, shuffle=TRUE,
tries=100, silent=FALSE, out2File=FALSE,
clusters.tmp.file="tmpcluster.RData")
```

Arguments

dataset	A dataset created by data.extract containing the information that should be used to generate the combined timeseries.
range	The maximum distance between two stations that should be connected in the station net.

alt.range	The maximum altitude difference between two stations that should be connected in the station net.
shuffle	A boolean value determining wether the stations should be processed in their order (value: FALSE) or if they should be shuffled befor processing (value: TRUE). Shuffled stations produce different results each run.
tries	If value of 'shuffle' is true, the integer value 'tries' determines how much cluster-lists should be created. The cluster-list with the lowest number of entries will be returned (this will increase the size of the clusters).
silent	A boolean value determining wether the function should generate output messages or not.
out2File	A boolean value determining wether the output will be stored in log-files.
clusters.tmp.file	A file where the clusters are saved for evaluation. If the value is NULL, no file will be created.

Details

This function joins the functions data.combine.stationNet, data.combine.clusters and data.combine.timeseries.

Value

A dataset containing the combined timeseries as a data.frame with same columns like a data.frame created by data.extract.

Author(s)

Daniel Doktor, Maximilian Lange

References

Schaber J., Badeck F. (2002). Evaluation of methods for the combination of phenological time series and outlier detection. Tree Physiology, 22:973-982

See Also

data.combine.clusters,data.combine.stationNet,data.combine.timeseries

Examples

```
  ## load extracted observations as created by 'data.extract'
  data(extractedObs)

  ## combine timeseries
  data.combined <- data.combine(dataset=extractedObs, range=5000,
alt.range=50, shuffle=TRUE, tries=3,
silent=FALSE, out2File=FALSE,
clusters.tmp.file=NULL)
```

`data.combine.clusters` *Station cluster creation.*

Description

Creates a list of station clusters.

Usage

```
data.combine.clusters(dataset, stations.net,
shuffle=TRUE, tries=100,
silent=FALSE, out2File=FALSE)
```

Arguments

`dataset`	The dataset with the stations to cluster and their coordinates.
`stations.net`	A list containing the information which stations are neighbours. Created by function `data.combine.stationNet`.
`shuffle`	A boolean value determining wether the stations should be processed in their order (value: FALSE) or if they should be shuffled befor processing (value: TRUE). Shuffled stations produce different results each run.
`tries`	If value of 'shuffle' is true, the integer value 'tries' determines how much cluster-lists should be created. The cluster-list with the lowest number of entries will be returned (this will increase the size of the clusters).
`silent`	A boolean value determining wether the function should generate output messages or not.
`out2File`	A boolean value determining wether the output will be stored in log-files.

Details

Creates a list of station clusters by randomly creating clusters (if value of 'shuffle' is TRUE) and chosing the list with the lowest number of clusters.

Value

Returns a list of station clusters (a station cluster is a vector containing related stations).

Author(s)

Daniel Doktor, Maximilian Lange

See Also

`data.combine.clusters.search`,`data.combine.stationNet`

Examples

```
  ## load extracted observations as created by 'data.extract'
  data(extractedObs)

  ## create station net
  stations.net <- data.combine.stationNet(extractedObs,
range=5000, alt.range=50, silent=FALSE,
out2File=FALSE)

  ## search clusters in station net
  clusters <- data.combine.clusters(extractedObs, stations.net,
shuffle=TRUE, tries=3, silent=FALSE,
out2File=FALSE)
```

---

data.combine.clusters.search

Cluster search

Description

Searches clusters in a station net.

Usage

```
data.combine.clusters.search(stations, stations.net,
shuffle=TRUE)
```

Arguments

stations	A list of all stations.
stations.net	A list of neighbourstations (neighbours of the station at same list position in list 'stations').
shuffle	A boolean value determining wether the stations should be processed in their order (value: FALSE) or if they should be shuffled befor processing (value: TRUE).

Details

Searches related stations in a station net. If value of 'shuffle' is TRUE, each run of this method produces different results.

Value

Returns a list containing related stations.

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
  ## load extracted observations as created by 'data.extract'
  data(extractedObs)

  ## create station net
  stations.net <- data.combine.stationNet(extractedObs, range=5000,
alt.range=50, silent=FALSE, out2File=FALSE)

  ## search clusters in station net
  stations <- as.list(unique(extractedObs$STAT_ID))
  clusters <- data.combine.clusters.search(stations,
stations.net, shuffle=TRUE)
```

`data.combine.stationNet` *Station net creator*

Description

Creates a net of the stations of dataset by checking the distances between the stations.

Usage

```
data.combine.stationNet(dataset, range,
alt.range, silent=FALSE,
out2File=FALSE)
```

Arguments

`dataset`	A dataset created by `data.extract` containing the information that should be used to generate the station net.
`range`	The maximum distance between two stations that should be connected in the net.
`alt.range`	The maximum altitude difference between two stations that should be connected in the net.
`silent`	A boolean value determining wether the function should generate output messages or not.
`out2File`	A boolean value determining wether the output will be stored in log-files.

Details

Only stations with a maximal distance of 'range' and maximal altitude difference of 'alt.range' are connected.

Value

Returns a list with entries for all stations of dataset. Each list entry stands for a station and contains all neighbours of that station.

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
  ## load extracted observations as created by 'data.extract'
  data(extractedObs)

  ## create station net
  stations.net <- data.combine.stationNet(extractedObs,
range=5000, alt.range=50, silent=FALSE,
out2File=FALSE)
```

---

data.combine.timeseries

Create combined timeseries

Description

Creates combined timeseries out of the dataset by using clusters of stations.

Usage

```
data.combine.timeseries(dataset, clusters,
silent=FALSE, out2File=FALSE,
minimalClusterSize=5)
```

Arguments

dataset	A dataset created by data.extract containing the information that should be used to generate the combined timeseries.
clusters	A list of station clusters generated by data.combine.clusters.
silent	A boolean value determining wether the function should generate output messages or not.
out2File	A boolean value determining wether the output will be stored in log-files.
minimalClusterSize	An integer value determining the minimal number of stations in a cluster to be included in combining process.

Details

The combined timeseries are created by using the function pheno.lad.fit of package 'pheno'. This process eliminates outliers and smooths the data.

Value

A dataset containing the combined timeseries as a data.frame with same columns like a data.frame created by data.extract.

Author(s)

Daniel Doktor, Maximilian Lange

References

Schaber J., Badeck F. (2002). Evaluation of methods for the combination of phenological time series and outlier detection. Tree Physiology, 22:973-982

See Also

pheno.lad.fit, data.extract, data.combine.stationNet, data.combine.clusters

Examples

```
  ## load extracted observations as created by 'data.extract'
  data(extractedObs)

  ## create station net
  stations.net <- data.combine.stationNet(extractedObs,
range=5000, alt.range=50, silent=FALSE,
out2File=FALSE)

  ## search clusters in station net
  clusters <- data.combine.clusters(extractedObs,
stations.net, shuffle=TRUE, tries=3,
silent=FALSE, out2File=FALSE)

  ## combine timeseries
  data.combined <- data.combine.timeseries(extractedObs,
clusters, out2File=FALSE)
```

---

data.coordinates2gridcellnumber

Search number of gridcell for given coordinates.

Description

Searches the number of the cell of a given spatial grid by given coordinates (Gauss-Krueger-Coordinates).

Usage

```
data.coordinates2gridcellnumber(grid, x,y)
```

Arguments

grid	The grid in which the cell should be searched.
x	The 'Rechtswert' of the Gauss-Krueger-Coordinates.
y	The 'Hochwert' of the Gauss-Krueger-Coordinates.

Details

Searches the number of the cell of a given spatial grid by given coordinates (Gauss-Krueger-Coordinates).

Value

Returns the cell-number of the grid-cell which is next to the given coordinates.

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
  ## load grid containing spatial information
  data(relatedGrid)

  ## search position in grid
  xy <- util.geoco2gk(54.12,10.17,4)
  gridposition <- data.coordinates2gridcellnumber(grid=relatedGrid,
x=xy[,1],y=xy[,2])

  gridposition
```

---

data.extract *Essential data extraction.*

Description

Extracts the essential data out of budburst and leafcolouring data and combines them.

Usage

```
data.extract(data.budburst, data.leafcolouring,
valid.years=1952:2009, out2File=FALSE,
silent=FALSE)
```

Arguments

data.budburst	A dataset extracted by function data.read.phase containing budburst information of currently processed plant.
data.leafcolouring	A dataset extracted by function data.read.phase containing leafcolouring information of currently processed plant.
valid.years	The years that should be included in extraction process (may be needed if some data in datasets are insufficient).
out2File	A boolean value determining wether the output will be stored in log-files.
silent	A boolean value determining wether the function should generate output messages or not.

Details

This function extracts the essential data out of budburst and leafcolouring data and combines them. Therefor it extracts the stations contained in both datasets and searches the leafcolouring dates and the related budburst date (at the following year).

Value

A dataset containing the station ID, the station geographic and the Gauss-Krueger-coordinates of the station, the altitude of the station, the year and the julian day of the budburst and the leafcolouring and outlier information for budburst and leafcolouring.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

data.read.phase,lcObs,bbObs,extractedObs

Examples

```
  ## load budburst observations (extracted via 'data.read.phase')
  data(bbObs)
  ## load leafcolouring observations (extracted via 'data.read.phase')
  data(lcObs)

  ## extract essential data
  data.extracted <- data.extract(data.budburst=bbObs,
data.leafcolouring=lcObs,
out2File=FALSE)
```

`data.loadTemperature` *Loads a temperature vector*

Description

Loads a temperature vector of given length from a given day of a given year.

Usage

```
data.loadTemperature(year, temperature.year, temperature.previous.year,
from.previous.year.doy, length, position,
scale.factor=0.1)
```

Arguments

`year`	The year of the budburst doy which should be calculated.
`temperature.year`	A vector of temperature data (with length 365 or 366) of year given by 'year'.
`temperature.previous.year`	A vector of temperature data (with length 365 or 366) of previous year.
`from.previous.year.doy`	The day the modelling starts (leafcolouring day of previous year).
`length`	The length of the resulting temperature vector.
`position`	The number of the grid-cell for which the temperatures should be loaded.
`scale.factor`	The down-scaling factor for the temperature data (needed if the data is scaled).

Details

Loads a temperature vector of given length from temperature data. The day of leafcolouring of the previous year should be given as starting day. The vector contains one temperature datapoint per day and should have a length of 300 or more to ensure a stable modelling process. Internal function used by method `data.addTemperatures`.

Value

A vector with temperature data for a given location and a given period.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

`data.addTemperatures`,`data.coordinates2gridcellnumber`

Examples

```
  t.year <- as.matrix(rnorm(365, mean=10, sd=5))
  t.p.year <- as.matrix(rnorm(365, mean=9, sd=6))
  temperatures <- data.loadTemperature(year=2006,temperature.year=t.year,
temperature.previous.year=t.p.year,
from.previous.year.doy=210, length=365,
position=1, scale.factor=0.1)
```

`data.main` *Main function for data conversion.*

Description

This function extracts data from tsv-Files, creates combined timeseries out of them if needed and adds temperature data.

Usage

```
data.main(extraction.done=TRUE, out2File=FALSE,
grid.related.to.temperaturefiles,
valid.years=1952:2009,
combine.time.series=TRUE, range=10000,
alt.range=50, temperature.scale.factor=1,
dataPath=getwd(),temperature.filenames,
temperature.matrix, pathForTmpFiles=getwd(),
pathToSave=getwd(), plant="beech")
```

Arguments

`extraction.done`	If data extraction was already done, turn this value to TRUE and the extracted values will be loaded from 'pathForTmpFiles'.
`out2File`	A boolean value determining wether the output will be stored in log-files.
`grid.related.to.temperaturefiles`	A grid containing spatial informations for the temperature files.
`valid.years`	The years that should be included in extraction process (may be needed if some data in datasets are insufficient).
`combine.time.series`	A boolean value determining wether the extracted timeseries should be combined or not.
`range`	The maximum distance between two stations that should be connected in the station station-net needed for the timeserie-combination.
`alt.range`	The maximum altitude difference between two stations that should be connected in the station-net needed for the timeserie-combination.
`temperature.scale.factor`	The down-scaling factor for the temperature data (needed if the data is scaled).

`dataPath` The path where the tsv-tables are stored. These tsv-tables should have the following name: '<PLANT>_budburst.tsv' for budburst-data or '<PLANT>_leafcolouring.tsv' for leafcolouring-data, where <PLANT> stands for the name of the plant, for example 'beech'.

`temperature.filenames` The full name of temperature files. The filenames have to contain the year (YYYY) of the temperature observation. The files should be stored as RData-Files and have to contain a matrix called 'edk.one.year' with 366 columns (one per day) and the number of rows equal to the number of rows in 'grid.related.to.Temperatures'. Should have the value NULL if a temperature matrix should be used instead.

`temperature.matrix` An array containing temperature data. The year of the observation should given as rowname, the columns should equal the julian day of the observation and the third dimension of the array should equal the location given in 'grid.related.to.Temperatures'. The matrix will be used instead of temperature files if 'temperature.filenames' is NULL.

`pathForTmpFiles` The path where the extracted data should be temporarily stored as RData-files.

`pathToSave` The path where the resulting dataframe should be stored as RData-file with the name '<PLANT>-dataset.RData' for non-combined data or '<PLANT>-dataset-cts.RData' for combined data.

`plant` The name of the plant which should be processed.

Details

This function is the main method for data extraction. It exctracts budburst and leafcolouring data from tsv-Files, creates combined timeseries out of them if needed and adds temperature data from RData files containing daily mean temperatures.

Value

Returns nothing, but saves the resulting dataset in 'pathToSave'.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

`data.extract`,`data.combine`,`data.addTemperatures`

Examples

```
## load temperature data
data(temperatures)
## load grid with spatial informations for temperature files
data(relatedGrid)

## extract dataset from datafiles in working directory
```

```
  ## Not run: dataset <- data.main(extraction.done=FALSE, out2File=FALSE,
grid.related.to.temperaturefiles=relatedGrid,
combine.timeseries=TRUE, range=5000, alt.range=50,
temperature.scale.factor=0.1, dataPath=getwd(),
temperature.filenames=NULL,
temperature.matrix=temperatures,
pathForTmpFiles=getwd(), pathToSave=getwd(),
plant="beech")
## End(Not run)
```

`data.read.phase` *Read data from TSV-table*

Description

This function reads phenological data out of a tsv-table.

Usage

```
data.read.phase(path, filename)
```

Arguments

`path`	The path of the tsv-file to read.
`filename`	The filename of the tsv-table.

Details

Reads phenological data out of a tsv-table containing the columns 'DWD_STAT_ID' (the ID of the phenological station), 'STAT_NAME' (the name of the phenological station), 'STAT_LON' (the longitude of the station), 'STAT_LAT' (the latitude of the station), 'STAT_ALT' (the altitude of the station), 'BEGIN_OBS' (the year the station started the observation), 'END_OBS' (the year the station stopped the observation), 'NATURRAUM_ID' (the id of the region), 'PHASE_ID' (the ID of the observed phase), 'OBS_DAY' (the julian day the observed phase starts), 'OBS_YEAR' (the year at which 'OBS_DAY' was observed), 'CHECKED' (a value determining wether the result was checked or not), 'outlier' (if result was checked, this value determines wether it was an outlier or not). Such a dataset can be obtained at www.phenology.de.

Value

Returns the table as R-Dataframe.

Author(s)

Daniel Doktor, Maximilian Lange

References

Dierenbach, J., Badeck, F.W., Schaber, J., 2013. The plant phenological online database (PPODB): an online database for long-term phenological data. International Journal of Biometeorology , 1-8.

See Also

lcObs,bbObs

Examples

```
  ##read file 'beech_budburst.tsv' out of working directory
  ## Not run: data.budburst <- data.read.phase(path=getwd(),
filename="beech_budburst.tsv")
## End(Not run)

  ## resulting data looks like the following:
  data(lcObs)
  data(bbObs)

  lcObs
  bbObs
```

---

dataFinal *Preprocessed data*

Description

This dataset contains phenological data (connected leafcolouring and budburst observations) as well as spatially related temperature data. The dataset was created out of extractedObs, relatedGrid and temperatures by use of data.addTemperatures.

Usage

```
dataFinal
```

Format

A dataframe containing three rows of connected phenological information combined with temperature data.

See Also

data.addTemperatures,bbObs,lcObs,extractedObs,relatedGrid,temperatures

extractedObs — *Phenological observation: leafcolouring and budburst*

Description

This dataset gives an example of leafcolouring observations connected with subsequent budburst observations. The observations are extracted out of phenological data from the German Weather Service (DWD) obtainable by the plant phenological online database PPODB. The dataset was created out of the datasets `bbObs` and `lcObs` by use of method `data.extract`.

Usage

```
extractedObs
```

Format

A dataframe containing three rows of connected observations.

Source

Plant Phenological Online Database (PPODB), `www.phenology.de`

References

Dierenbach, J., Badeck, F.W., Schaber, J., 2013. The plant phenological online database (PPODB): an online database for long-term phenological data. International Journal of Biometeorology , 1-8.

See Also

`data.extract`,`bbObs`,`lcObs`

lcObs — *Phenological observation of leafcolouring*

Description

This dataset gives an example of leafcolouring observations from the German Weather Service (DWD) obtainable by the plant phenological online database PPODB. The dataset was extracted out of the downloadable tsv-File by use of `data.read.phase`.

Usage

```
lcObs
```

Format

A dataframe containing three observations.

Source

Plant Phenological Online Database (PPODB), www.phenology.de

References

Dierenbach, J., Badeck, F.W., Schaber, J., 2013. The plant phenological online database (PPODB): an online database for long-term phenological data. International Journal of Biometeorology , 1-8.

See Also

data.read.phase,bbObs

`pim.solve` *Promoter-Inhibitor-Model*

Description

Applies a promoter-inhibitor-model to a given dataset.

Usage

```
pim.solve(params, data, model.no=1,
silent=FALSE, out2File=FALSE)
```

Arguments

`params`	The parameters for the promoter-inhibitor-model as list or vector with following order: a1, a2, a3, a4, T.min.i, T.opt.i, T.max.i, T.min.p, T.opt.p, T.max.p
`data`	A dataset containing the station ID, the station geographic and the Gauss-Krueger-coordinates of the station, the altitude of the station, the year and the julian day of the budburst and the leafcolouring, outlier information for budburst and leafcolouring and temperature data for modelling. Can be created by using the function `data.main`.
`model.no`	The promoter-inhibitor-model to use. See references for more details.
`silent`	A boolean value determining wether the function should generate output messages or not.
`out2File`	A boolean value determining wether the output will be stored in log-files.

Details

Applies a promoter-inhibitor-model with given parameters to a given dataset.

Value

A dataset containing the values of the origin dataset and additionally the modelled budburst days.

Author(s)

Daniel Doktor, Maximilian Lange

References

Schaber, J. and Badeck, F.-W. (2003). Physiology-based phenology models for forest tree species in Germany . International Journal of Biometeorology 47:193-201

See Also

data.main

Examples

```
  ## load preprocessed data
  data(dataFinal)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)
```

---

relatedGrid *Grid containing spatial information related to temperature data*

Description

This dataset, based on a digital elevation model contains spatial information related to the temperature data.

Usage

```
relatedGrid
```

Format

A dataframe containing spatial information.

Source

digital elevation model

See Also

temperatures

`result.extract.interpolate` *Result interpotion*

Description

Interpolates result values with given spatial information.

Usage

```
result.extract.interpolate(mask.grid, values, alt, x, y)
```

Arguments

`mask.grid`	The grid with spatial information the values are ordered by.
`values`	The values which should be interpolated.
`alt`	The related altitude for the gridcells of 'mask.grid'.
`x`	The related Rechtswert (Gauss-Krueger-coordinates) for the gridcells of 'mask.grid'.
`y`	The related Hochwert (Gauss-Krueger-coordinates) for the gridcells of 'mask.grid'.

Details

Interpolates result values with given spatial information by external drift kriging.

Value

A vector with the interpolated values.

Author(s)

Daniel Doktor, Maximilian Lange

References

Krige, D.G., 1951. A statistical approach to some basic mine valuation problems on the witwatersrand. Journal of the Chemical, Metallurgical and Mining Society of South Africa 52, 119-139. Pebesma, E.J., 2004. Multivariable geostatistics in S: the gstat package. Computers & Geosciences 30, 683-691.

See Also

`result.extract.main`,`data.main`

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)

  ## resolve outlier information
  outliers <- result$outlier.bb + result$outlier.lc
  outliers.na <- which(is.na(outliers)==TRUE)
  outliers[outliers.na] <- rep(0, length(outliers.na))

  mask.grid <- relatedGrid

  ## extract valid modelled values
  values.model <- result.extract.sub(mask.grid=mask.grid,
result$doy.bb.pim, result$gk4.x,
result$gk4.y, outliers=outliers,
silent=FALSE, withOutliers=FALSE)$values

  ## interpolate result values with spatial informations of mask.grid
  values.model <- result.extract.interpolate(mask.grid=mask.grid,
values=values.model, alt=mask.grid$alt,
x=mask.grid$x, y=mask.grid$y)
```

`result.extract.main` *Essential result extraction*

Description

Extracts essential results from model.

Usage

```
result.extract.main(mask.grid, result.grid, model="pim",
interpolate=TRUE, silent=FALSE, withOutliers=FALSE)
```

Arguments

`mask.grid`	A grid containing spatial information related to values (which should be extracted) and their Gauss-Krueger-Coordinates. The values in the resulting data.frame will be ordered related to values in mask.grid.

result.grid	The grid created by pim.solve containing observated and modelled values.
model	A character value determining which model was used to create the results (either 'pim' or 'tsm').
interpolate	A boolean value determining wether the results should be interpolated (with spatial information of mask.grid) or not.
silent	A boolean value determining wether the function should generate output messages or not.
withOutliers	A boolean value determining wether outliers should be included in extraction and transformation or not.

Details

Extracts essential results (observed and modelled results and their difference and coordinates) from used model.

Value

A data.frame containing the modelled values ('doy.model'), the observed values ('doy.observed'), their difference ('doy.dif') and related coordinates ('x','y')

Author(s)

Daniel Doktor, Maximilian Lange

See Also

result.extract.sub,result.extract.mask,result.extract.interpolate

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)

  ## extract essential results
  result.values <- result.extract.main(
mask.grid=relatedGrid,
result.grid=result, model="pim",
interpolate=FALSE, silent=FALSE)
```

`result.extract.mask` *Result masking*

Description

Masks values with the spatial informations of a given grid.

Usage

```
result.extract.mask(mask.grid, values)
```

Arguments

`mask.grid`	The grid with the spatial informations which should be used for masking.
`values`	The values to mask (in same order as 'mask.grid').

Details

Masks values with the spatial informations of a given grid. Gridcells with NA value are masked in the resulting data.frame by setting them to '-9999'.

Value

A vector of masked values.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

`result.extract.main`

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)
```

```
  ## resolve outlier information
  outliers <- result$outlier.bb + result$outlier.lc
  outliers.na <- which(is.na(outliers)==TRUE)
  outliers[outliers.na] <- rep(0, length(outliers.na))

  mask.grid <- relatedGrid

  ## extract valid modelled values
  values.model <- result.extract.sub(mask.grid=mask.grid,
result$doy.bb.pim, result$gk4.x,
result$gk4.y, outliers=outliers,
silent=FALSE, withOutliers=FALSE)$values

  ## mask result values with spatial informations of mask.grid
  values.model <- result.extract.mask(relatedGrid,
values.model)
```

`result.extract.sub` *Extract values*

Description

Extracts and transformates given values to a data.frame with same order as a given masking grid.

Usage

```
result.extract.sub(mask.grid, values, gk4.x, gk4.y,
outliers, silent=FALSE, withOutliers=FALSE)
```

Arguments

`mask.grid`	A grid containing spatial information related to the values and their Gauss-Krueger-Coordinates. The values in the resulting data.frame will be ordered related to values in 'mask.grid'.
`values`	The values that should be extracted and transformed.
`gk4.x`	Gauss-Krueger-Rechtswert related to values.
`gk4.y`	Gauss-Krueger-Hochwert related to values.
`outliers`	Outlier information related to values.
`silent`	A boolean value determining wether the function should generate output messages or not.
`withOutliers`	A boolean value determining wether outliers should be included in extraction and transformation or not.

Details

Extracts and transformates given values (with related Gauss-Krueger-Coordinates) to a data.frame with same order as a given masking grid.

Value

A grid containing the values and their coordinates (as given in 'mask.grid').

Author(s)

Daniel Doktor, Maximilian Lange

See Also

result.extract.main

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)

  ## resolve outlier information
  outliers <- result$outlier.bb + result$outlier.lc
  outliers.na <- which(is.na(outliers)==TRUE)
  outliers[outliers.na] <- rep(0, length(outliers.na))

  mask.grid <- relatedGrid

  ## extract valid modelled values
  values.model <- result.extract.sub(mask.grid=mask.grid,
result$doy.bb.pim, result$gk4.x,
result$gk4.y, outliers=outliers,
silent=FALSE, withOutliers=FALSE)$values
```

---

result.main *Result evaluation*

Description

Main function for result evaluation.

Usage

```
result.main(mask.grid, result.grid, plant="beech", model="pim",
year=1954, picPath=getwd(), picName="beech-budburst",
createFiles=TRUE, rsquarePath=getwd(),
rsquareFile="rsquare.RData", rsquare.type="cod",
silent=FALSE, withOutliers=FALSE)
```

Arguments

mask.grid	A grid with spatial information related to the resulting grid of modelling.
result.grid	The resulting grid of modelling. Can be obtained by using pim.solve or tsm.solve.
plant	The plant name for which the values of 'result.grid' are modelled.
model	A character value determining which model was used to create the results (either 'pim' or 'tsm').
year	The processed year.
picPath	The path where the created png-files should be stored.
picName	The filename of the created png-files.
createFiles	A boolean flag determining wether the results should be stored in files or not.
rsquarePath	The path where the RData-file with the rsquare-dataset should be stored.
rsquareFile	The filename of the RData-file with the rsquare-dataset.
rsquare.type	The value of type (either 'cod' or 'pearson') determines whether the coefficient of determination or the squared pearson correlation coefficient is used as rsquare.
silent	A boolean value determining wether the function should generate output messages or not.
withOutliers	A boolean value determining wether outliers should be included in extraction and transformation or not.

Details

Main function for result evaluation. Extracts essential values from resulting grid, interpolates and masks values, creates histogramms, scatterplot and maps and calculates r-square.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

result.extract.main,result.extract.interpolate,result.pic.histogramm,result.pic.maps,result.pic.scatt

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)
  ## evaluate resuts
  result.main(mask.grid=relatedGrid,
result.grid=result, plant="beech", model="pim",
year="1952-2009", picPath=getwd(), picName="beech-budburst",
createFiles=FALSE, rsquarePath=getwd(),
rsquareFile="rsquare.RData", rsquare.type="cod",
silent=FALSE, withOutliers=FALSE)
```

`result.pic.histogramm` *Histogramm Creation*

Description

Creates histogramms of results.

Usage

```
result.pic.histogramm(values, picPath=getwd(),
picName="budburst-beech", silent=FALSE,
createFile=TRUE)
```

Arguments

`values`	The data.frame with result values (can be created with `result.extract.main`).
`picPath`	The path where the png-files should be stored.
`picName`	The name of the created files with the histogramms.
`silent`	A boolean value determining wether the function should generate output messages or not.
`createFile`	A boolean value determining wether a png-File will be created or not.

Details

Creates histogramms of results (observed value, modelled value and difference of them).

Value

Returns nothing but creates histogramms as png-files with given path and filename.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

result.extract.main,result.main

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)

  ## extract essential results
  result.values <- result.extract.main(
mask.grid=relatedGrid,
result.grid=result, model="pim",
interpolate=FALSE, silent=FALSE)

  ## create histogramm
  result.pic.histogramm(values=result.values,
picPath=getwd(), picName="beech_budburst",
silent=FALSE, createFile=FALSE)
```

---

result.pic.maps *Map creation*

Description

Creates maps out of results.

Usage

```
result.pic.maps(values, picPath=getwd(),
picName="beech-budburst", silent=FALSE,
createFile=TRUE)
```

Arguments

values	The data.frame with result values (can be created with result.extract.main).
picPath	The path where the png-files should be stored.
picName	The name of the created files with the maps.
silent	A boolean value determining wether the function should generate output messages or not.
createFile	A boolean value determining wether a png-File will be created or not.

Details

Creates maps out of results.

Value

Returns nothing but stores maps as png-files with given path and filename.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

result.extract.main,result.main

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)

  ## extract essential results
  result.values <- result.extract.main(
mask.grid=relatedGrid,
result.grid=result, model="pim",
interpolate=TRUE, silent=FALSE)

  ## not enough successfully calculated budbursts
  ## replace with examples
  result.values$doy.model <- round(rnorm(n=9,mean=100,sd=20))
  result.values$doy.observed <- round(rnorm(n=9,mean=100,sd=20))
```

```
  ## create maps
  result.pic.maps(values=result.values,
picPath=getwd(), picName="beech_budburst",
silent=FALSE, createFile=FALSE)
```

`result.pic.scatterplot` *Scatterplot creation*

Description

Creates a scatterplot out of result values.

Usage

```
result.pic.scatterplot(values, picPath=getwd(),
picName="beech-budburst", createFile=TRUE)
```

Arguments

`values`	The values for which the scatterplot should be created.
`picPath`	The path where the png-files should be stored.
`picName`	The name of the created file with the scatterplot.
`createFile`	A boolean value determining wether a png-File will be created or not.

Details

Creates a scatterplot out of result values. The observed values are shown at the abscissa and the modelled values at the ordinate.

Value

Returns nothing but stores the scatterplot as png-file with given path and filename.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

`result.main`

Examples

```
  ## load preprocessed data
  data(dataFinal)
  ## load spatial information
  data(relatedGrid)

  ## set or load params
  params <- c(0, 0.058326, 0.109494, 0.039178,
-10.34, -0.89, 18.11,-10.03,
28.61, 44.49)

  ## apply model
  result <- pim.solve(params, dataFinal, model.no=11,
silent=FALSE, out2File=FALSE)

  ## extract essential results
  result.values <- result.extract.main(
mask.grid=relatedGrid,
result.grid=result, model="pim",
interpolate=FALSE, silent=FALSE)

  ## create scatterplot
  result.pic.scatterplot(values=result.values,
picPath=getwd(), picName="beech_budburst",
createFile=FALSE)
```

`result.rsquare` *R-square calculation.*

Description

Calculates the r-square value of a model.

Usage

```
result.rsquare(values, type="cod")
```

Arguments

`values`	A list containing the modelled (doy.model) and the observed (doy.observed) values.
`type`	The value of type (either 'cod' or 'pearson') determines whether the coefficient of determination or the squared pearson correlation coefficient is calculated.

Details

Calculates the r-square value of modelled values with given observed values.

Value

The r-square as numeric value.

Author(s)

Daniel Doktor, Maximilian Lange

See Also

result.main

Examples

```
  modelled <- c(100,102,98,97,96)
observed <- rep(100,5)
values <- list(doy.model=modelled, doy.observed=observed)
result.rsquare(values, type="cod")
```

---

temperatures *Temperature data*

Description

This dataset contains interpolated temperature data based on data obtained from WebWerdis, the Web-based weather request and distribution system of the German Weather Service (DWD). The interpolation was done via external drift kriging provided by the 'gstat' package.

Usage

```
extractedObs
```

Format

A matrix containing 4 years (366 days) of temperature data at 9 different locations given by relatedGrid.

Source

WebWerdis (DWD)

References

Krige, D.G., 1951. A statistical approach to some basic mine valuation problems on the witwatersrand. Journal of the Chemical, Metallurgical and Mining Society of South Africa 52, 119-139. Pebesma, E.J., 2004. Multivariable geostatistics in S: the gstat package. Computers & Geosciences 30, 683-691.

See Also

relatedGrid

`tsm.solve` *Temperature-Sum-Model*

Description

Applies a temperature-sum-model to a given dataset.

Usage

```
tsm.solve(params, data, silent=FALSE, out2File=FALSE)
```

Arguments

`params`	The parameters for the temperature-sum-model as list or vector with following order: T_b, F^*. T_b is the threshold temperature. Above this value, temperatures are summed for calculating the development rate. F^* is the threshold development rate. If the development rate reached this threshold, the phase occurs.
`data`	A dataset containing the station ID, the station geographic and the Gauss-Krueger-coordinates of the station, the altitude of the station, the year and the julian day of the budburst and the leafcolouring, outlier information for budburst and leafcolouring and temperature data for modelling. Can be created by using the function `data.main`.
`silent`	A boolean value determining wether the function should generate output messages or not.
`out2File`	A boolean value determining wether the output will be stored in log-files.

Details

Applies a temperature-sum-model with given parameters to a given dataset.

Value

A dataset containing the values of the origin dataset and additionally the modelled budburst days.

Author(s)

Daniel Doktor, Maximilian Lange

References

Menzel, A. (1997). Phaenologie von Waldbaeumen unter sich aendernden Klimabedingungen - Auswertung der Beobachtungen in den Internationalen Phaenologischen Gaerten und Moeglichkeiten der Modellierung von Phaenodaten. Thesis. Forstwissenschaftliche Fakultaet der Uni Muenchen. Muenchen, Universitaet Muenchen.

See Also

`data.main`

Examples

```
  ## load preprocessed data
  data(dataFinal)

  ## set or load params
  params <- c(0, 100)

  ## apply model
  result <- tsm.solve(params, dataFinal,
silent=FALSE, out2File=FALSE)
```

`util.geoco2gk` *Geographic coordinates to Gauss-Krueger-Coordinates*

Description

Converts geographic coordinates to Gauss-Krueger-Coordinates.

Usage

```
util.geoco2gk(x,y, meridian=4)
```

Arguments

`x`	A vector of longitudes to transform.
`y`	A vector of latitudes to transform.
`meridian`	The referenced meridian for Gauss-Krueger-Coordinates.

Details

Converts geographic coordinates (longitude, latitude) to Gauss-Krueger-Coordinates ('Rechtswert', 'Hochwert') referenced by a given meridian.

Value

A matrix containing the the 'Rechtswert' of the Gauss-Krueger-Coordinates in its first column and the 'Hochwert' in its second column.

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
x <- c(51.3, 54.7)
y <- c(12.3, 13.5)
gk.xy <- util.geoco2gk(x,y,4)
```

`util.isLeapYear` *Leap year check*

Description

Checks wether a given year is a leap year or not.

Usage

```
util.isLeapYear(year)
```

Arguments

`year`	A year or vector of years to check.

Details

This function checks wether a year (yyyy) is a leap year or not.

Value

Returns a boolean vector determining wether the given years are leap years (TRUE) or not (FALSE).

Author(s)

Daniel Doktor, Maximilian Lange

Examples

```
util.isLeapYear(2000:2012)
```

Index