Package 'SOfireA'

November 15, 2016

Title Satellite Observations for Fire Activity

Version 1.0

Date 2016-11-15

Author Matthias Forkel <`matthias.forkel@geo.tuwien.ac.at`> [aut, cre]

Maintainer Matthias Forkel <`matthias.forkel@geo.tuwien.ac.at`>

Description SOFIA (Satellite Observations for Fire Activity) is an empirical modelling concept to predict burned area based on satellite and climate data. The package implements the basic SOFIA model structure, and functions to optimize and plot SOFIA models.

Depends R (>= 3.2.3), rgenoud, ModelDataComp

Suggests

License GPL-2

URL

LazyLoad yes

R topics documented:

`SOfireA-package` *Satellite Observations for Fire Activity*

Description

SOFIA (Satellite Observations for Fire Activity) is an empirical modelling concept to predict burned area based on satellite and climate data. The package implements the basic SOFIA model structure, and functions to optimize and plot SOFIA models.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

`AllEqual` *Check if all values in a vector are the same*

Description

This function is used to check if all values in a vector are equal. It can be used for example to check if a time series contains only 0 or NA values.

Usage

```
AllEqual(x)
```

Arguments

`x`	numeric, character vector, or time series of type ts

Value

The function returns TRUE if all values are equal and FALSE if it contains different values.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Examples

```
# check if all values are equal in the following vectors:
AllEqual(1:10)
AllEqual(rep(0, 10))
AllEqual(letters)
AllEqual(rep(NA, 10))
```

`FitSofia` *Fit a Sofia model to a data set*

Description

The function fits a SOFIA model to a dataset.

Usage

```
FitSofia(x, y, unc = NULL, per.group = rep(FALSE, ncol(x)), nodes = 4,
    sofiapar, restart = 0, cost = NULL, ...)
```

Arguments

`x`	data.frame with independent variables
`y`	dependent variable (observation)
`unc`	uncertainty of dependent variable
`per.group`	a boolean vector that indicates if a column in x acts per group (e.g. PFTs)
`nodes`	number of nodes for parallel compuation during genetic optimization
`sofiapar`	SofiaPar object with prior parameters
`restart`	restart previous Sofia optimization? 0 = start new, 1 = continue with previous, 2 = do post-processing
`cost`	cost function to be used
`...`	further arguments

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`FitDataModel`

`Logistic` *Logistic function*

Description

Compute values of a logistic function.

Usage

```
Logistic(par, x, ...)
```

Arguments

`par`	parameters of logistic function, a vector of length 3 (asymptote, slope, turning point)
`x`	independent variable
`...`	further arguments (not used)

Details

No details.

Value

a vector

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`FitDataModel`

Examples

```
x <- -20:20
par <- c(1, 0.5, 0)
plot(x, Logistic(par, x), type="l")

par <- c(1, 0.2, 0)
plot(x, Logistic(par, x), type="l")

par <- c(10, -1, 0)
```

```
plot(x, Logistic(par, x), type="l")

par <- c(-2, -1, 0)
plot(x, Logistic(par, x), type="l")
```

MakeFig *Calculate figure positions for graphics that consist of multiple figure*

Usage

```
MakeFig(nfig, border = c(0, 1, 0, 1), nrow = NULL, ncol = NULL)
```

Arguments

`nfig`	number of figures
`border`	relative graphic borders in which the figures should be placed
`nrow`	number of rows to arrange the figures
`ncol`	number of cols to arrange the figures

Value

A list with positions for each figure and number of rows and cloumns

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

Pd2Logistic *Fit a logistic function to a partial dependence*

Description

Fits parameters of logistic function to an object of class "PartialDependence"

Usage

```
Pd2Logistic(pd, normalize = TRUE, direction = c(0, -1, 1), ...)
```

Arguments

`pd`	object of class "PartialDependence"
`normalize`	normalize y variable to [0,1]
`direction`	slope of the fit: 0 test positive and negative, -1 test only negative, 1 test onle positive
`...`	further arguments

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`PartialDependence`, `Logistic`

`plot.Sofia` *plot a Sofia object*

Description

Plots a `Sofia` object.

Usage

```
## S3 method for class 'Sofia'
plot(x, ylab = "y", mfrow = NULL, names = NULL, main = NULL,
    plot.order = NULL, labels = paste0("(", letters, ")"), ...)
```

Arguments

`x`	a 'Sofia' object
`ylab`	label for response variable
`mfrow`	number of rows and columns for the plot
`names`	names of the variables in the response functions
`main`	title of the plot
`plot.order`	Order for plotting of factors
`labels`	Labels for subplots. Set to NULL to avoid labels.
`...`	further arguments (not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`Sofia`

Examples

```
# explanatory variables
sm <- 1:100
temp <- rnorm(100, 12, 10)
x <- cbind(sm, temp)

# fractional coverage of groups, e.g. plant functional types
tree <- runif(100, 0, 0.8)
grass <- 1 - tree
area <- cbind(tree, grass)

# with some more realisitc parameters:
par <- SofiaPar(colnames(x), per.group=c(TRUE, FALSE), group.names=c("tree", "grass"))
par$par <- c(1, 1, 20, 2, 1, -0.2, -0.1, 13, 10)
sf <- Sofia(x, area, per.group=c(TRUE, FALSE), sofiapar=par)
plot(sf)
```

`plot.SofiaOpt` *plot a SofiaOpt object*

Description

The optimization within `SofiaFit` produces files that can be used to restart or monitor an optimization experiment. These files can be read with `ReadSofiaFit` and plotted with this function..

Usage

```
## S3 method for class 'SofiaOpt'
plot(x, plot.objfct = c("Cor", "MEF", "Pbias"), ...)
```

Arguments

`x`	an object of class `SofiaFit` as returned by `ReadSofiaFit`
`plot.objfct`	which objective function should be plotted (maximum 3)?
`...`	further arguments (not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`SofiaFit`

`predict.Sofia` *Predict values based on a 'Sofia' object*

Description

Make a predicition based on a `Sofia` object and newdata

Usage

```
## S3 method for class 'Sofia'
predict(object, newdata, return.all = FALSE, ...)
```

Arguments

`object`	an object of class 'Sofia', see `Sofia`
`newdata`	a data frame with columns names as in object$group.names for area fractions of groups and as in object$x.names for explantory variables
`return.all`	return all Sofia results? If FALSE, returns only total burned area
`...`	further arguments (not used)

Details

No details.

Value

A vector with predicted values.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`Sofia`, `SofiaOpt`

`ReadSofiaOpt` *Read results from an SOFIA optimization experiment*

Description

The optimization within `SofiaOpt` produces files that can be used to restart or monitor an optimization experiment. This function reads these files.

Usage

```
ReadSofiaOpt(files, combine = TRUE, ...)
```

Arguments

`files`	vector of file names
`combine`	combine several files in a single file?
`...`	further arguments (not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`SofiaOpt`

`Sofia` *Satellite Observations for Fire Activity*

Description

SOFIA (Satellite Observations for Fire Activity) is an empirical modelling concept to predict burned area based on satellite and climate data. Thereby several logistic functions are multiplicatively combined.

Usage

```
Sofia(x, area = rep(1, nrow(x)), per.group = rep(FALSE, ncol(x)),
    sofiapar = NULL, par = NULL, return.all = TRUE, ...)
```

Arguments

`x`	data.frame with independent variables
`area`	a vector or data.frame/matrix with fractional coverage of grid cell area. If 'area' is a vector, it represents the maximal fractional burned area of a grid cell (e.g. the maximum vegetated area). If 'area' is a data.frame or matrix, it represents fractional coverage of groups (e.g. PFTs). Columns should represent groups and rows should be observations (grid cells and time steps).
`per.group`	a boolean vector that indicates if a column in x acts per group (e.g. PFTs)
`sofiapar`	object of class `SofiaPar` which is used for the fit. If NULL, the argument 'par' is used to create sofiapar using the function `SofiaPar`
`par`	vector of parameters of logistic functions. If NULL, default parameters are used (that are usually physically not plausible)
`return.all`	return all input and results? The function returns an object of class 'Sofia'. If TRUE, this object includes in the 'data' slot the fitted values, the fits per group, the response functions, the inputs 'x' and 'area'. If FALSE, only the fitted values are included.
`...`	further arguments

Details

No details.

Value

an object of class 'Sofia' which is actually a list.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

SofiaFit, Logistic

Examples

```
# explanatory variables
sm <- 1:100
temp <- rnorm(100, 12, 10)
x <- cbind(sm, temp)

# fractional coverage of groups, e.g. plant functional types
tree <- runif(100, 0, 0.8)
grass <- 1 - tree
area <- cbind(tree, grass)

# calculate Sofia with some dummy parameters:
sf <- Sofia(x, area, per.group=c(TRUE, FALSE))
sf$eq
summary(sf$data)
plot(sf)

# with some more realisitc parameters:
par <- SofiaPar(colnames(x), per.group=c(TRUE, FALSE), group.names=c("tree", "grass"))
par
par$par <- c(1, 1, 20, 2, 1, -0.2, -0.1, 13, 10)
sf <- Sofia(x, area, per.group=c(TRUE, FALSE), sofiapar=par)
plot(sf)

sm <- 1:100
sm.2 <- sm
temp <- rnorm(100, 12, 10)
x <- cbind(sm, sm.2, temp)
par <- SofiaPar(colnames(x), per.group=c(TRUE, TRUE, FALSE), group.names=c("tree", "grass
par
par$par <- c(2, 1, 20, 2, 2, 0.3, 0.2, 20, 40, 1, 1, -0.2, -0.1, 20, 10)
sf <- Sofia(x, area, per.group=c(TRUE, TRUE, FALSE), sofiapar=par)
plot(sf)
```

SofiaOpt *Optimize a SOFIA model using genetic optimization*

Description

The function fits a SOFIA model to observations by estimating model parameters using `Sofia` genetic optimization.

Usage

```
SofiaOpt(x, area = rep(1, nrow(x)), per.group = rep(FALSE, ncol(x)),
    sofiapar = NULL, par.init = NULL, obs, unc = NULL, cost = NULL,
    pop.size = 500, max.generations = 30, path = NULL, restart = 0,
    nodes = 5, BFGSburnin = max.generations - 2, ...)
```

Arguments

`x`	data.frame with independent variables
`area`	a vector or data.frame/matrix with fractional coverage of grid cell area. If 'area' is a vector, it represents the maximal fractional burned area of a grid cell (e.g. the maximum vegetated area). If 'area' is a data.frame or matrix, it represents fractional coverage of groups (e.g. PFTs). Columns should represent groups and rows should be observations (grid cells and time steps).
`per.group`	a boolean vector that indicates if a column in x acts per group (e.g. PFTs)
`sofiapar`	object of class `SofiaPar` which is used for the fit. If NULL, the argument 'par.init' is used to create sofiapar using the function `SofiaPar`
`par.init`	matrix of inital parameters for optimization
`obs`	a vector of observed values
`unc`	vector of observation uncertainties, if NULL an uncertainty of 1 is is used for all observations
`cost`	a function to compute the cost, if NULL SSE (sum of squared error) is used
`pop.size`	population size, see `genoud`
`max.generations`	maximum number of generations, see `genoud`
`path`	directory for optimization results
`restart`	restart: 0 = start with new optimization, 1 = start with best individuals from previous optimization in 'path', 2 = return results
`nodes`	how many nodes to use for parallel executaion of genoud?
`BFGSburnin`	The number of generations before the L-BFGS-B algorithm is first used, see `genoud`
`...`	further arguments to `genoud`

Details

No details.

Value

an object of class 'Sofia' which is actually a list.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`Sofia`

Examples

```
# some example data
n <- 500
sm <- runif(n, 0, 100) # soil moisture
temp <- rnorm(n, 12, 10) # temperature
tree <- runif(n, 0, 1) # fractional tree cover
grass <- 1 - tree # fractional grass cover
area <- cbind(tree, grass)
x <- cbind(sm, temp)

# create 'observations'
sofiapar <- SofiaPar(colnames(x), colnames(area), per.group=c(TRUE, FALSE))
sofiapar$par <- c(1, 1, 20, 2, 1, -0.2, -0.1, 13, 10) # actual parameters
sf <- Sofia(x, area, per.group=c(TRUE, FALSE), sofiapar=sofiapar)
plot(sf) # fitted values vs. temperature
obs <- sf$data$y # 'observations'

# re-estimate parameters: for a real optimization pop.size and max.generations should be
setwd("~/tmp/")
par.init <- sofiapar$par * 1.5 # some inital parameters for optimization
sfbest <- SofiaOpt(x, area, per.group=c(TRUE, FALSE), obs=obs, sofiapar=sofiapar, par.ini
sfbest
plot(sfbest)

# plot iterations of optimization: set directory where optimization results are saved
files <- list.files(pattern="SofiaOpt")
fit <- ReadSofiaOpt(files)
plot(fit)
plot(fit, plot.objfct = c("IoA", "FV", "MEF"))

# compare retrieved with original parameters
sfbest$par$par / par.init

# compare retrieved vs. real
sim <- sfbest$data$y
lim <- range(c(sim, obs))
plot(obs, sim, ylim=lim, xlim=lim)
abline(0,1)
ObjFct(sim, obs)
```

```
# compare real and retrieved response functions
plot(sf$data$x.temp, sf$data$f.temp)
points(sfbest$data$x.temp, sfbest$data$f.temp, col="red")

plot(sf$data$x.sm, sf$data$f.sm.tree)
points(sfbest$data$x.sm, sfbest$data$f.sm.tree, col="red")

plot(sf$data$x.sm, sf$data$f.sm.grass)
points(sfbest$data$x.sm, sfbest$data$f.sm.grass, col="red")
```

SofiaPar *Parameters for SOFIA models*

Description

The function creates an object of class 'SofiaPar' (which is actually a list) which contains information about Sofia model parameters.

Usage

```
SofiaPar(x.names, per.group = rep(FALSE, length(x.names)), group.names = NULL,
    par.act = NULL, par.prior = NULL, par.lower = NULL, par.upper = NULL,
    ...)
```

Arguments

`x.names`	names of independent variables
`per.group`	a boolean vector that indicates if a column in x acts per group (e.g. PFTs)
`group.names`	names of groups
`par.act`	
`par.prior`	prior parameters
`par.lower`	lower parameter limits
`par.upper`	upper parameter limits
`...`	further arguments

Details

No details.

Value

An object of class 'SofiaPar', which is actually a list.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre]

References

No reference.

See Also

`Sofia`, `Logistic`

Examples

```
# explanatory variables
sm <- 1:100
temp <- rnorm(100, 12, 10)
x <- cbind(sm, temp)

# fractional coverage of groups, e.g. plant functional types
tree <- runif(100, 0, 0.8)
grass <- 1 - tree
area <- cbind(tree, grass)

# parameters for SOFIA models
par <- SofiaPar(colnames(x), per.group=c(TRUE, FALSE), group.names=c("tree", "grass"))
par
```