Package 'LPJmLmdi'

August 16, 2019

Title Model-Data Integration for the LPJmL Dynamic Global Vegetation Model

Version 1.31

Date 2019-01-22

Author
Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Drueke <drueke@pik-potsdam.de> [aut

Maintainer
Matthias Forkel <matthias.forkel@geo.tuwien.ac.at>, Markus Drueke <drueke@pik-potsdam.de>

Description Model-data integration framework for the LPJmL dynamic global vegetation model. Specifically, the package provides functions 1) to optimize LPJmL model parameters using the GENOUD genetic optimization algorithm, 2) to read and write LPJmL input data, and 3) to read LPJmL output files.

Depends R (>= 2.15.3)

Imports

License GPL-2

URL

LazyLoad yes

NeedsCompilation no

R topics documented:

`LPJmLmdi-package` *Model-Data Integration for the LPJmL Dynamic Global Vegetation Model*

Description

Model-data integration framework for the LPJmL dynamic global vegetation model. Specifically, the package provides functions 1) to optimize LPJmL model parameters using the GENOUD genetic optimization algorithm, 2) to read and write LPJmL input data, and 3) to read LPJmL output files.

Details

The DESCRIPTION file:

Package:	LPJmLmdi
Title:	Model-Data Integration for the LPJmL Dynamic Global Vegetation Model
Version:	1.31
Date:	2019-01-22
Author:	Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Drueke <drueke@pik-potsdam.de>
Maintainer:	Matthias Forkel <matthias.forkel@geo.tuwien.ac.at>, Markus Drueke <drueke@pik-potsdam.de>
Description:	Model-data integration framework for the LPJmL dynamic global vegetation model. Specifically, the packa
Depends:	R (>= 2.15.3)
Imports:	
License:	GPL-2
URL:	
LazyLoad:	yes

Index of help topics:

AggregateNCDF	Temporal aggregations and statistics on NetCDF files
AllEqual	Check if all values in a vector are the same
BarplotCost	plot a barplot of the cost change from prior to best parameter set
BreakColors	Colours from class breaks
BreakColours	Colours from class breaks
Breaks	Class breaks for plotting
Cbalance	Calculate global C balance, C fluxes and stocks
ChangeParamFile	Change parameters in a parameter file
ChangeSoilCodeFile	Change soil code in LPJ soil code file
CheckLPJpar	Checks LPJ parameters 'LPJpar'
CheckMemoryUsage	Check usage of memory by R objects
CombineLPJpar	Combines several 'LPJpar' objects into one
CombineRescueFiles	Combine single rescue files into one rescue file
CorW	Weighted correlation
CorrelationMatrixS	plot a correlation matrix
CostMDS.KGE	Cost function for multiple data streams based on Kling-Gupta efficiency
CostMDS.KGEw	Cost function for multiple data streams based on a weighted Kling-Gupta efficiency
CostMDS.SSE	Cost function for multiple data streams based on SSE
CreateRestartFromRescue	Create a *.pro file from binary rescue files to restart optimization
DefaultParL	default 'par' settings for plots
Df2optim	Convert a data.frame to a 'optim' list
EstOptimUse	Estimate optimal number of jobs given a number of cluster nodes
FileExistsWait	Iterative checking and waiting for a file.
GridProperties	Derive grid properties from an object of class 'LPJfiles'
InfoCLM	Returns information about a CLM file
InfoLPJ	Information about a LPJmL binary file
InfoNCDF	Get information about variables in a NetCDF
IntegrationData	Create an object of class 'IntegrationData'
IntegrationData2Df	Converts IntegrationData to a data.frame
IntegrationDataset	Create an object of class 'IntegrationDataset'
LE2ET	Compute evapotranspiration (ET) from latent heat (LE).
LPJ2NCDF	Convert binary LPJmL model output files to NetCDF
LPJfiles	Create an object of class 'LPJfiles'
LPJmLmdi-package	Model-Data Integration for the LPJmL Dynamic Global Vegetation Model
LPJpar	Create an object of class 'LPJpar'
LPJparList	Create a list of 'LPJpar' objects
LPJpp	Post-process LPJmL model output
LPJppNBP	Post-process LPJmL model output: calculate NEE
LWin2LWnet	Compute long-wave net radiation from long-wave

	incoming radiation and temperature.
LegendBar	Add a colour legend bar to a plot
MeanW	Weighted mean
OptimizeLPJgenoud	Optimize LPJ using the GENOUD optimizer (genetic optimization using derivatives)
PlotPar	Plot parameter vs. cost
PlotParPCA	plot a PCA of optimized parameters
PlotParUnc	Plot the psoterior parameter uncertainty
PlotWorld110	Plot a world map based on 1:110Mio data
PrepareRestartFiles	Prepare restart files to restart OptimizeLPJgenoud
ReadBIN	Read simple binary files without header Read a CLM file to a SpatialPixelsDataFrame
ReadCLM	Read a CLM file to a SpatialPixelsDataFrame
ReadGrid	Reads a binary input data grid file.
ReadLPJ	Read a LPJ binary file
ReadLPJ2IntegrationData	Read LPJ model results into an of class IntegrationData
ReadLPJ2ts	Read a LPJ binary file and returns a spatial averaged time series
ReadLPJinput	Read and subset CLM files to LPJinput objects
ReadLPJsim	Read a LPJ simulation results
ReadOutputvars	Read 'outputvars.par' to get information about LPJmL output
ReadPRO	Read *.pro files as produced from GENOUD
RegridLPJinput	Regrid or subset LPJmL input
Rescue2Df	Convert a 'rescue' list to a data.frame
Rescue2LPJpar	Add information from a 'rescue' list to an 'LPJpar' object
RunLPJ	Run LPJmL from R and get results
SSE	Sum-of-squared residuals error
SdW	Weighted standard deviation
StandardError	Compute standard errors from a variance-covariance matrix
StartingValues	Get starting values for genoud from *.pro file
Texture2Soilcode	Convert soil texture to a LPJ soilcode
Turnover	Calculate turnover time from stock and flux
VarCovMatrix	Compute variance-covariance matrix
VarW	Weighted variance
WriteBIN	Write a BIN file from SpatialPointsDataFrame
WriteCLM	Write a CLM file from SpatialPointsDataFrame
WriteGrid	Write a *.grid file from a matrix of coordiantes or a SpatialPointsDataFrame
WriteLPJinput	Write an object of class 'LPJinput' to CLM files
WriteLPJpar	Writes an object of class 'LPJpar' as parameter file or table.
WriteNCDF4	Write NetCDF files
plot.Cbalance	Plots a C balance
plot.IntegrationData	Plot an object of class IntegrationData
plot.LPJpar	Plot parameters in 'LPJpar' object.

plot.LPJparList	Plots to compare LPJpar objects
plot.LPJsim	Plots a LPJsim object
plot.rescue	plot an object of class "rescue" / monitor OptimizeLPJgenoud

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

AggFPCBoBS *Temporal aggregation for BoBS PFT: get BoBS from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCBoBS(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCBoNE *Temporal aggregation for BoNE PFT: get BoNE from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCBoNE(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCBoNS *Temporal aggregation for BoNS PFT: get BoNS from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCBoNS(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCPoH *Temporal aggregation for PoH PFT: get PoH from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCPoH(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCTeBE *Temporal aggregation for PFTs: get TeBE from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCTeBE(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCTeBS *Temporal aggregation for PFTs: get TeBS from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCTeBS(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCTeH *Temporal aggregation for PFTs: get TeH from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCTeH(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCTeNE *Temporal aggregation for PFTs: get TeNE from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCTeNE(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCTrBE *Temporal aggregation for PFTs: get TrBE from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCTrBE(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggFPCTrBR *Temporal aggregation for PFTs: get TrBR from vector of all PFTs, average over years*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggFPCTrBR(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggMaxNULL *Temporal aggregation: aggregate by using annual maximum*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggMaxNULL(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggMeanMean *Temporal aggregation: first mean, then mean = mean over all values*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggMeanMean(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggMeanNULL *Temporal aggregation: first mean, then nothing = mean per group*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggMeanNULL(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggMSC *Temporal aggregation: mean seasonal cycle*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggMSC(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns the mean seasonal cycle

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggNULLMean *Temporal aggregation: mean over all values*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggNULLMean(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggQ09NULL *Temporal aggregation: aggregate by using quantile with prob=0.9*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggQ09NULL(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggregateNCDF *Temporal aggregations and statistics on NetCDF files*

Description

Compute temporal aggregations and statistics of data in NetCDF files.

Usage

```
AggregateNCDF(files, fun.agg = sum, var.name = NULL, tstep = NULL,
    agg.monthly = TRUE, agg.annual = TRUE, agg.ndaily = TRUE,
    ndays = 7, stat.annual = TRUE, stat.monthly = FALSE, stat.ndaily = FALSE,
    stat.daily = FALSE, msc.monthly = TRUE, path.out = NULL,
    path.out.prefix = "img", nodes = 1, stats = NULL, ...)
```

Arguments

files	(character) file name or vector file names. In case of multiple file names, it is assumed that each file corresponds to a different time period (i.e. all files are a time series)
fun.agg	function to be used for temporal aggregations
var.name	(character) variable name in NetCDF files for which computations should be done. If NULL, all variables will be processed.
tstep	(character) time step of input data: "daily", "ndaily" (period of n days), "monthly", "annual". If NULL tstep will be estimated from the files.

agg.monthly	(boolean) aggregate to monthly data? This will be only done if 'tstep' is daily or ndaily.
agg.annual	(boolean) aggregate to annual data? This will be only done if 'tstep' is < annual.
agg.ndaily	(boolean) aggregate to N-daily periods? This will be only done if 'tstep' is < ndays.
ndays	(integer) length of period [in days] for N-daily aggregations. For example, a aggregation to 7-daily periods will be done if ndays=7. A period starts always at the 1st January. Please note that a 7-daily aggregation does not necessarily correspond to calendar weeks (see GroupDates for details). This aggregation will be only done if 'tstep' is < ndays.
stat.annual	(boolean) compute statistics based on annual data?
stat.monthly	(boolean) compute statistics based on monthly data?
stat.ndaily	(boolean) compute statistics based on N-daily data?
stat.daily	(boolean) compute statistics based on daily data?
msc.monthly	(boolean) compute (mean/median) seasonal cycles on monthly data and monthly anomalies? This computation is based on SeasonalCycleNCDF and uses CDO modules.
path.out	directory for output files. If NULL, directories will be created within the location of the files, otherwise directories will be created under the specified directory.
path.out.prefix	prefix for output directory names, directory names are created according to the following pattern 'prefix'_'resolution'_'timestep' (e.g. img_0d25_monthly)
nodes	How many nodes should be used for parallel processing of files? Parallel computing can be only used if length(files) > 1.
stats	statistical metrics to compute
...	further agruments (unused)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

`AggSumMean` *Temporal aggregation: first sum, then mean*

Description

This function can be provided to `IntegrationDataset` to aggregate model results to the temporal resolution of the observations.

Usage

```
AggSumMean(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

AggSumNULL *Temporal aggregation: first sum, then nothing = sum per group*

Description

This function can be provided to IntegrationDataset to aggregate model results to the temporal resolution of the observations.

Usage

```
AggSumNULL(x, agg)
```

Arguments

x	full time series
agg	vector of grouping elements (years)

Details

No details.

Value

The function returns a the aggregated result.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationDataset

`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], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

Examples

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

`BarplotCost` *plot a barplot of the cost change from prior to best parameter set*

Description

The function plots two barplots that are showing the change in the cost per dataset.

Usage

```
BarplotCost(rescue.l, type = 1:2, ylim = NULL, set.par = TRUE,
    ...)
```

Arguments

rescue.l	a list of class "rescue", see CombineRescueFiles
type	plot type: 1 barplot of total cost from prior and best, 2: change of cost per data set
ylim	limits of y-axis (works for type = 2)
set.par	set par() settings from DefaultParL
...	further arguments for plot

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles

Examples

```
# files <- c(list.files(pattern="rescue.RData", recursive=TRUE), list.files(pattern="rescue0.RData", recursi
# rescue.l <- CombineRescueFiles(files, remove=FALSE)
# BarplotCost(rescue.l)
```

BreakColors *Colours from class breaks*

Description

Creates colour palettes from a vector of class breaks

Usage

```
BreakColors(x, pal = NULL, rev = FALSE, cols = NULL, ...)
```

Arguments

x	numeric vector of class breaks
pal	name of a colour palette from brewer.pal
rev	should the colour palette be reversed?
cols	alternatively, a colour vector to be interpolated
...	Further arguments (unused)

Details

No details.

Value

The function returns a vector of colours.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

BreakColours

Examples

```
brks1 <- seq(0, 10, 2)
cols1 <- BreakColours(brks1)

brks2 <- seq(-100, 100, 25)
cols2 <- BreakColours(brks2)

brks3 <- seq(-100, 100, 25)
cols3 <- BreakColours(brks3, pal="BrBG")

brks4 <- seq(0, 10, 1)
cols4 <- BreakColours(brks4, cols=c("red", "green", "blue"), rev=TRUE)

MapRb()
LegendBarRb(brks=brks1, cols=cols1)
LegendBarRb(brks=brks2, cols=cols2, pos="top", lon = c(-180, 180), lat = c(-20, -15))
LegendBarRb(brks=brks3, cols=cols3, pos="inside", lon = c(-180, 180), lat = c(15, 20))
LegendBarRb(brks=brks4, cols=cols4, pos="inside", lon = c(-180, 180), lat = c(30, 35))
```

BreakColours *Colours from class breaks*

Description

Creates colour palettes from a vector of class breaks

Usage

```
BreakColours(x, pal = NULL, rev = FALSE, cols = NULL, ...)
```

Arguments

x	numeric vector of class breaks
pal	name of a colour palette from brewer.pal
rev	should the colour palette be reversed?
cols	alternatively, a colour vector to be interpolated
...	Further arguments (unused)

Details

No details.

Value

The function returns a vector of colours.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

BreakColours

Examples

```
brks1 <- seq(0, 10, 2)
cols1 <- BreakColours(brks1)

brks2 <- seq(-100, 100, 25)
cols2 <- BreakColours(brks2)

brks3 <- seq(-100, 100, 25)
cols3 <- BreakColours(brks3, pal="BrBG")

brks4 <- seq(0, 10, 1)
cols4 <- BreakColours(brks4, cols=c("red", "green", "blue"), rev=TRUE)

MapRb()
LegendBarRb(brks=brks1, cols=cols1)
LegendBarRb(brks=brks2, cols=cols2, pos="top", lon = c(-180, 180), lat = c(-20, -15))
LegendBarRb(brks=brks3, cols=cols3, pos="inside", lon = c(-180, 180), lat = c(15, 20))
LegendBarRb(brks=brks4, cols=cols4, pos="inside", lon = c(-180, 180), lat = c(30, 35))
```

`Breaks` *Class breaks for plotting*

Description

Calculates class breakpoints based on quantiles.

Usage

```
Breaks(x, n = 12, quantile = c(0.01, 0.99), zero.min = FALSE,
    ...)
```

Arguments

`x`	numeric vector
`n`	number of breaks
`quantile`	lower and upper quantiles that should be used to exclude outliers
`zero.min`	should the minimum break be at 0?
`...`	Further arguments (unused)

Details

No details.

Value

The function returns a vector of values.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`BreakColours`

Examples

```
Breaks(rnorm(100, 50, 30))
Breaks(runif(100, 10, 30))
Breaks(rlnorm(100))
```

Cbalance *Calculate global C balance, C fluxes and stocks*

Description

The function takes numeric vectors or NetCDF files with values of C fluxes and stocks, and calculates C balances and turnover times [years] (see details). In case of NetCDF files, global total C fluxes and stocks [PgC year-1] are computed from NetCDF files. Thereby the input data unit needs to be [gC m-2] for stocks and [gC m-2 year-1] for fluxes. However, the argument scale is a multiplier that can be used to convert to the original unit to [gC m-2]. The results are returned as a data.frame (table). If some values or files are not provided (i.e. NA), the function will first try to compute these from other metrics (see details).

Usage

```
Cbalance(gpp = NA, npp = NA, ra = NA, rh = NA, reco = NA, firec = NA,
    estab = NA, harvest = NA, vegc = NA, soilc = NA, litc = NA,
    scale = 1, ti = NA, mask = NA, ...)
```

Arguments

gpp	numeric vector or NetCDF file of gross primary production
npp	numeric vector or NetCDF file of net primary production
ra	numeric vector or NetCDF file of autotrophic respiration
rh	numeric vector or NetCDF file of heterotrophic respiration
reco	numeric vector or NetCDF file of ecosystem respiration
firec	numeric vector or NetCDF file of fire C emissions
estab	numeric vector or NetCDF file of establishment C flux (specific to LPJ)
harvest	numeric vector or NetCDF file of C removal from vegetation through harvest
vegc	numeric vector or NetCDF file of vegetation C stocks (or biomass)
soilc	numeric vector or NetCDF file of soil C stocks
litc	numeric vector or NetCDF file of litter C stocks
scale	multiplier to convert original units to gC m-2 (stocks) or gC m-2 year-1 (fluxes)
ti	time axis of the data. In case of NetCDF files, time will be extracted from the files.
mask	A mask in a NetCDF file in order to compute the C fluxes, stocks, balances and turnover times only for specific regions.
...	further arguments (currently not used)

Details

The function computes (global) terrestrial C balances based on given input data. The used terminology is based on Schulze (2006) and Chapin et al. (2006). The following equations are used:

- Net primary production NPP = GPP - Ra
- Ecosystem respiration Reco = Rh + Ra

- Net ecosystem exchange NEE = Reco - GPP = Rh - NPP
- Net biome productivity NBP = (GPP + Estab) - (Reco + FireC + Harvest)

Vegetation and total ecosystem turnover times are computed based on the formulas in Carvalhais et al. (2014) and Thurner et al. (2016): \item Vegetation turnover time: TauVeg_NPP = VegC / NPP \item Vegetation turnover time based on GPP is a approximiation to the real vegetation turnover time (TauVeg_NPP) assuming that NPP is around 50% of GPP: TauVeg_GPP = VegC / ((GPP+Estab) * 0.5) \item Total ecosystem turnover time as in Carvalhais et al. (2014): TauEco_GPP = (VegC + SoilC + LitC) / (GPP+Estab) \item Total ecosystem turnover time based on Reco: TauEco_Reco = (VegC + SoilC + LitC) / Reco \item Total ecosystem turnover time based on Reco and disturbances: TauEco_Dist = (VegC + SoilC + LitC) / (Reco + FireC + Harvest)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

Carvalhais et al. (2014), Global covariation of carbon turnover times with climate in terrestrial ecosystems, Nature, 514(7521), 213<e2><80><93>217, doi:10.1038/nature13731. Chapin et al. (2006), Reconciling Carbon-cycle Concepts, Terminology, and Methods, Ecosystems, 9(7), 1041<e2><80><93>1050, doi:10.1007/s10021-005-0105-7. Schulze (2006), Biological control of the terrestrial carbon sink, Biogeosciences, 3(2), 147<e2><80><93>166, doi:10.5194/bg-3-147-2006. Thurner, M., C. Beer, N. Carvalhais, M. Forkel, M. Santoro, M. Tum, and C. Schmullius (2016), Large<e2><80><90>scale variation in boreal and temperate forest carbon turnover rate is related to climate, Geophysical Research Letters, doi:10.1002/2016GL068794.

See Also

Turnover

Examples

```
# with some typical numbers for the global C budget:
cbal <- Cbalance(gpp=123, npp=61, rh=57, firec=2, vegc=400, soilc=2400)
cbal
plot(cbal)

## using time series::
#cbal <- Cbalance(gpp=118:128, npp=(118:128)*rnorm(11, 0.5, 0.1), rh=57, firec=runif(11, 0, 4), harvest=2, ve
#cbal
#plot(cbal)
```

ChangeParamFile *Change parameters in a parameter file*

Description

The function writes values to a parameter file. It requires a 'file.template' in which the positions of the new parameter values are marked with a flag. For example, instead of a parameter of 0.5 for alphaa in a parameter file the flag ALPHAA is written. The function substitutes this flag with the new parameter value in a new file 'file.new'.

Usage

```
ChangeParamFile(newpar, file.template, file.new, wait = FALSE,
    ...)
```

Arguments

newpar	a named vector with new parameter values
file.template	file name of the template parameter with flagged parameters
file.new	file name of the new parameter file
wait	If TRUE wait 1 second to check if file.template exists in order to relax slow file writting.
...	further arguments (currently not used)

Details

The function works only on Unix systems because it is based on 'sed'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

Examples

```
# newpar <- c(ALPHAA_BoNE=0.8)
# LPJChangeParamFile(newpar, file.template="pft_template.par", file.new="pft.par")
```

`ChangeSoilCodeFile` *Change soil code in LPJ soil code file*

Description

The function changes the soil code for the specified grid cells and writes a new LPJ soil code file.

Usage

```
ChangeSoilCodeFile(file.soilcode, file.soilcode.new, xy, newcode,
    file.grid = "cru.grid", ...)
```

Arguments

`file.soilcode`	original soil code file
`file.soilcode.new`	new soil code file
`xy`	matrix of grid cells (lon, lat) where the soil code should be changed
`newcode`	new soil code at each grid cell (vector with length = nrow(xy))
`file.grid`	grid file for the original soil code file
`...`	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`ReadBIN`

CheckLPJpar *Checks LPJ parameters 'LPJpar'*

Description

The function checks if LPJ parameters are within the lower and upper boundaries or are 0.

Usage

```
CheckLPJpar(lpjpar, correct = FALSE)
```

Arguments

lpjpar	object of class 'LPJpar'
correct	correct parameter values (TRUE) or return error message?

Details

No details.

Value

the function return an object of class 'LPJpar'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

LPJpar

CheckMemoryUsage *Check usage of memory by R objects*

Description

Prints a message about the used memory and writes a file with the used memory per each R object.

Usage

```
CheckMemoryUsage(...)
```

Arguments

...	The function has no arguments.

Value

a data.frame

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

`CombineLPJpar` *Combines several 'LPJpar' objects into one*

Description

The function takes several lpjpar objects and combines them into one LPJpar object

Usage

```
CombineLPJpar(lpjpar.l)
```

Arguments

`lpjpar.l`	a list of `LPJpar` objects

Details

No details.

Value

The function returns a list of class 'LPJparList'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`LPJpar`

`CombineRescueFiles` *Combine single rescue files into one rescue file*

Description

Within OptimizeLPJgenoud, RunLPJ creates rescue file ("_.rescue0.RData") that save the parameter vectors and cost of each individual during optimization. These files allow to create restart files to restart OptimizeLPJgenoud (`CreateRestartFromRescue`). During OptimizeLPJgenoud many rescue files can be created. The function CombineRescueFiles reads the individual files, combines the rescue objects, saves it in one "rescue.RData" file, and deletes the single files.

Usage

```
CombineRescueFiles(files.rescue, remove = TRUE)
```

Arguments

`files.rescue`	file names
`remove`	save new rescue file and delete single rescue files?

Details

No details.

Value

The function returns a list of class "rescue", whereby each element corresponds to one individual of the genetic optimization with two entries: 'cost' (cost of the individual) and 'dpar' (parameter scaled relative to the prior parameter).

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`CreateRestartFromRescue`, `plot.rescue`

CorrelationMatrixS *plot a correlation matrix*

Usage

```
CorrelationMatrixS(data, method = "spearman", iscor = NULL, main = "",
    ...)
```

Arguments

data	a correlation matrix or a data.frame (
method	method to compute the correlation
iscor	Is 'data' a correlation matrix?
main	main title for the plot
...	further arguments for plot

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles

CorW *Weighted correlation*

Description

Compute the correlation.

Usage

```
CorW(x, y, w = rep(1, length(x)))
```

Arguments

x	a vector of x values
y	a vector of y values
w	vector of weights

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

ObjFct

Examples

```
x <- 1:5
y <- x * -1 + rnorm(5)
cor(x, y)
CorW(x, y, w=c(1, 1, 1, 2, 2))
```

`CostMDS.KGE` *Cost function for multiple data streams based on Kling-Gupta efficiency*

Description

The function computes for each grid cell and data stream in 'integrationdata' the cost besed on the Kling-Gupta efficiency (KGE, Gupta et al. 2009, J. Hydrology). See Forkel et al. (in prep.) for the specific use of KGE for multiple data streams.

Usage

```
CostMDS.KGE(integrationdata)
```

Arguments

`integrationdata`	object of class 'integrationdata', see IntegrationData

Details

No details.

Value

The function returns a list with the total cost (total), the cost per KGE component (per.cell), per data streams (per.ds), per KGE component and data stream (per.cell.ds), and the fractional contribution of a data stream and KGE component to the total cost (fractional).

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

`CostMDS.KGEw` *Cost function for multiple data streams based on a weighted Kling-Gupta efficiency*

Description

The function computes for each grid cell and data stream in 'integrationdata' the cost besed on the Kling-Gupta efficiency (KGE, Gupta et al. 2009, J. Hydrology). Thereby each component of KGE is weighted by the uncertainty of the observations (i.e. weighted mean, variance and correlation). See Forkel et al. (in prep.) for the specific use of KGE for multiple data streams.

Usage

```
CostMDS.KGEw(integrationdata)
```

Arguments

`integrationdata`
: object of class 'integrationdata', see `IntegrationData`

Details

No details.

Value

The function returns a list with the total cost (total), the cost per KGE component (per.cell), per data streams (per.ds), per KGE component and data stream (per.cell.ds), and the fractional contribution of a data stream and KGE component to the total cost (fractional).

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

Examples

```
# load(paste0(path.me, "/lpj/LPJmL_131016/out_optim/opt_fpc/OFPC_BO-GI-BM_v1_all_0_59_posterior-best.RData")
# x <- result.post.lpj$integrationdata
# plot(x, 2)

# cost.see <- CostMDS.SSE(x)
# cost.kge <- CostMDS.KGE(x)
# cost.kgew <- CostMDS.KGEw(x)

# DefaultParL(mfrow=c(1,3))
# barplot(cost.see$per.ds)
# barplot(t(cost.kge$per.cell.ds))
# barplot(t(cost.kgew$per.cell.ds))
```

`CostMDS.SSE` *Cost function for multiple data streams based on SSE*

Description

The function computes the cost for each grid cell and data stream in 'integrationdata'. Firstly, the cost per data stream and grid cell is computed using the defined 'CostFunction' for each `IntegrationDataset`. Secondly, the cost is weighted by (1) the dataset-specific weight, (2) the number of observations per grid cell and data streams, and (3) by the grid cell area.

Usage

```
CostMDS.SSE(integrationdata)
```

Arguments

`integrationdata`
: object of class 'integrationdata', see `IntegrationData`

Details

No details.

Value

The function returns a list with the total cost, the cost per grid cell, per data streams, per grid cell and data stream, the error as computed with the defined CostFunction, the number of observations per grid cell and data stream, the weighting factors and grid cell area.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

`CreateRestartFromRescue` *Create a *.pro file from binary rescue files to restart optimization*

Description

The function creates a *.pro file friom binary 'rescue' files. The *.pro file can be used to restart OptimizeLPJgenoud.

Usage

```
CreateRestartFromRescue(path.rescue, pop.size)
```

Arguments

`path.rescue`	directory where the resuce files from each iteration of the optimization are saved.
`pop.size`	(estimated) population size of the genetic optimization

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`genoudLPJrescue`

`DefaultParL` *default 'par' settings for plots*

Description

The function calls 'par' with some default settings to improve plots. See `par` for details.

Usage

```
DefaultParL(mfrow = c(1, 1), mar = c(3.7, 3.5, 2.5, 0.5), oma = c(0.8,
    0.1, 0.1, 0.2), mgp = c(2.4, 1, 0), cex = 1.3, cex.lab = cex *
    1.1, cex.axis = cex * 1.1, cex.main = cex * 1.1, ...)
```

Arguments

mfrow	number of rows/columns
mar	margins
oma	outer margins
mgp	margin line for axis title, label and lines
cex	text and symbol size
cex.lab	label size
cex.axis	axis anootation size
cex.main	title size
...	further arguments to par

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

par

Examples

```
DefaultParL()
plot(1:10)
```

Df2optim *Convert a data.frame to a optim list*

Description

The function takes a 'data.frame' as created by Rescue2DF and converts it to a list with the same structure like the results of the optim and genoud functions.

Usage

```
Df2optim(optim.df, pop.size = NA, ...)
```

Arguments

optim.df	a 'data.frame' as created by Rescue2DF
pop.size	used population size. If NA, ngen (number of generations) and peak generation cannot be returned correctly. In this case both estimates will be 1.
...	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles

Examples

```
# files <- c(list.files(pattern="rescue.RData", recursive=TRUE), list.files(pattern="rescue0.RData", recursi
# rescue.l <- CombineRescueFiles(files, remove=FALSE)
# optim.df <- Rescue2Df(rescue.l)
# opt <- Rescue2optim(rescue.l)
# opt
```

EstOptimUse *Estimate optimal number of jobs given a number of cluster nodes*

Usage

```
EstOptimUse(nodes = 16, wish = 1000)
```

Arguments

nodes	number of cluster nodes that you want to use
wish	approx. number of elements

Details

No details.

Value

an integer value

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

WriteCLM

`FileExistsWait` *Iterative checking and waiting for a file.*

Description

The function repeately checks if a file exists and returns TRUE if the file is existing.

Usage

```
FileExistsWait(file, waitmin = 0, waitinterval = 0.5, waitmax = 2,
    ...)
```

Arguments

`file`	file for which checking and waiting should be applied
`waitmin`	minimum waiting time (seconds)
`waitinterval`	interval after which the existence of the file should be checked again (seconds)
`waitmax`	maximum waiting time (seconds)
`...`	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

Examples

```
FileExistsWait(system.file("external/rlogo.grd", package="raster"))
FileExistsWait("nofile.txt")
```

`GridProperties` *Derive grid properties from an object of class 'LPJfiles'*

Description

The function reads the grid of the input files in 'LPJfiles' and computes the area per grid cell.

Usage

```
GridProperties(lpjfiles, res = 0.5, ...)
```

Arguments

`lpjfiles`	list of class 'LPJinput'
`res`	resolution of LPJmL
`...`	further arguments (currently not used)

Details

No details.

Value

the function returns a list with 'grid' (raster of grid cells), 'area' (vector of grid cell area) and 'ncell' (number of grid cells)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`LPJfiles`

InfoCLM *Returns information about a CLM file*

Usage

```
InfoCLM(file.clm, endian = "little", ...)
```

Arguments

file.clm	CLM file name with extension *.clm
endian	endianess of the file
...	Further arguments (currently not used).

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

WriteLPJinput

InfoLPJ *Information about a LPJmL binary file*

Description

The function reads information about a LPJ binary output file.

Usage

```
InfoLPJ(file.bin = "fpc.bin", file.grid = "grid.bin", file.annual = c("vegc.bin",
    "litc.bin", "soilc.bin"), size = 4, data.type = numeric(),
    ...)
```

Arguments

file.bin	binary LPJ output file
file.grid	binary LPJ grid file
file.annual	one of the binary LPJ output files with annual data
size	the number of bytes per element in the byte stream.
data.type	data type of the file (default=numeric())
...	further arguments (currently not used)

Details

No details.

Value

The function returns a list with information about the LPJ binary file (number of grid cells, number of years, number of bands, spatial extent).

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

ReadLPJ

Examples

```
# InfoLPJ("vegc.bin")
```

InfoNCDF *Get information about variables in a NetCDF*

Usage

```
InfoNCDF(file)
```

Arguments

file	file name

Value

The function returns a list with information about the dimensions and variables in the NetCDF file.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

See Also

WriteNCDF4

`IntegrationData` *Create an object of class 'IntegrationData'*

Description

The function takes several objects of class `IntegrationDataset` and converts them to an object 'IntegrationData' that is used in `RunLPJ` and `OptimizeLPJgenoud`.

Usage

```
IntegrationData(...)
```

Arguments

`...`	one or several objects of class 'IntegrationDataset'

Details

No details.

Value

The function returns a list of class 'IntegrationData'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`IntegrationDataset`

Examples

```
# # grid cells for which LPJmL should be run and for which the integration data should be extracted
# xy <- cbind(c(136.75, 137.25, 160.75,168.75), c(45.25, 65.25, 68.75, 63.75))

# # use monthly FAPAR in model-data integration
# fapar <- IntegrationDataset(name="FAPAR", unit="",
# data.val.file="GIMMS.FAPAR.1982.2011.nc",
# data.unc.file=0.12,
# data.time=seq(as.Date("1982-01-01"), as.Date("2011-12-31"), by="month"),
# model.val.file="mfapar.bin",
# model.agg=NULL,
# xy=xy,
# data.factor=NULL,
# cost=TRUE,
```

```
# CostFunction=SSE,
# weight=1)

# # use mean annual GPP in model-data integration
# gpp <- IntegrationDataset(name="GPP", unit="gC m-2 yr-1",
# data.val.file="MTE.GPP.1982.2011.meanannual.nc",
# data.unc.file="MTE.GPPunc.1982.2011.meanannual.nc",
# data.time=seq(as.Date("1982-01-01"), as.Date("2011-12-31"), by="month"),
# model.val.file="mgpp.bin",
# model.agg=AggSumMean, # sum of each year, mean over all years -> mean annual GPP
# xy=xy,
# data.factor=NULL,
# cost=TRUE,
# CostFunction=SSE,
# weight=1)

# integrationdata <- IntegrationData(fapar, gpp)
```

`IntegrationData2Df` *Converts IntegrationData to a data.frame*

Description

The function takes an object of class `IntegrationData` and converts it into a data.frame in long format. The data.frame has the columns 'lon', 'lat', 'time' and 'id' (a combination of lon_lat_time), and columns for each variable in IntegrationData.

Usage

```
IntegrationData2Df(x, sim.name = "sim", ...)
```

Arguments

`x`	object of class `IntegrationData`
`sim.name`	name that should be added to the variables for the simulation (e.g. use 'sim', or something like 'prior' or 'posterior' to create column names like 'FAPAR.sim')
`...`	further arguments (not used)

Details

No details.

Value

a data.frame

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationData

`IntegrationDataset` *Create an object of class 'IntegrationDataset'*

Description

The function sets up an object of class 'IntegrationDataset' to define a dataset that should be used in model optimization, including dataset properties and the corresponding model output files. The function also reads the data input files as defined in 'data.val.file' and subsets the input data for the grid cells in 'xy'. One or several 'IntegrationDataset's need to be collected in an object of class IntegrationData which is used in the RunLPJ and OptimizeLPJgenoud functions.

Usage

```
IntegrationDataset(name, unit = "", data.val.file, data.unc.file,
    data.time, model.time = data.time, model.val.file, xy, AggFun = NULL,
    data.factor = 1, model.factor = 1, cost = TRUE, CostFunction = SSE,
    weight = 1)
```

Arguments

`name`	name of the dataset or variable
`unit`	unit of the variable (same unit as in LPJmL model output file 'model.val.file')
`data.val.file`	name of file with the observation values, should be a file that can be read with `brick`
`data.unc.file`	name of file with the data uncertainties or a numeric value if the same uncertainty value should be used for all observations
`data.time`	a vector of class 'Date' with the time steps of the observations.
`model.time`	a vector of class 'Date' with the time steps for which model results should be read. For example, if data.val.file represents just one value (e.g. long-term mean), the full time period for which the model results should be averaged needs to be defined here.
`model.val.file`	file name of the corresponding model result [e.g. model.val.file="mnpp.bin"] or function without arguments [e.g. model.val.file=function() ReadLPJ("mnpp.bin", start=1901, end=2009, ...)]. The option to pass a function allows to perform any calculations on LPJ model results or to combine several LPJ model outputs in order to be comparable with observations.
`xy`	a matrix of grid cells that is used in RunLPJ and OptimizeLPJgenoud. The data in 'data.val.file' and 'data.unc.file' is extracted for these grid cells.
`AggFun`	aggregation function to aggregate model results to the temporal resolution of the observations, for example AggSumMean for annual sums and mean over annual sums. If NULL no temporal aggregation is done.

data.factor	scaling factor to be applied to the observation data, e.g. for unit conversions
model.factor	scaling factor to be applied to model outputs, e.g. for unit conversions or scaling
cost	Should the data stream be included in the computation of the total cost (TRUE) or not (FALSE). In case of FALSE, evaluation plots are produced for this dataset but the dataset is not considered in the compuation of the total cost and therfore not in optimization.
CostFunction	cost function that should be used for this dataset, default SSE
weight	weighting factor for the dataset in the cost function, cost = CostFunction / number of observations * weight

Details

No details.

Value

The function returns a list of class 'IntegrationDataset'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

IntegrationData

Examples

```
# # grid cells for which LPJmL should be run and for which the integration data should be extracted
# xy <- cbind(c(136.75, 137.25, 160.75,168.75), c(45.25, 65.25, 68.75, 63.75))

# # use monthly FAPAR in model-data integration
# fapar <- IntegrationDataset(name="FAPAR", unit="",
# data.val.file="GIMMS.FAPAR.1982.2011.nc",
# data.unc.file=0.12,
# data.time=seq(as.Date("1982-01-01"), as.Date("2011-12-31"), by="month"),
# model.val.file="mfapar.bin",
# model.agg=NULL,
# xy=xy,
# data.factor=NULL,
# cost=TRUE,
# CostFunction=SSE,
# weight=1)

# # use mean annual GPP in model-data integration
# gpp <- IntegrationDataset(name="GPP", unit="gC m-2 yr-1",
# data.val.file="MTE.GPP.1982.2011.meanannual.nc",
# data.unc.file="MTE.GPPunc.1982.2011.meanannual.nc",
```

```
# data.time=seq(as.Date("1982-01-01"), as.Date("2011-12-31"), by="month"),
# model.val.file="mgpp.bin",
# model.agg=AggSumMean, # sum of each year, mean over all years -> mean annual GPP
# xy=xy,
# data.factor=NULL,
# cost=TRUE,
# CostFunction=SSE,
# weight=1)
```

LE2ET *Compute evapotranspiration (ET) from latent heat (LE).*

Usage

```
LE2ET(le, temp = 20, rho_w = 1000)
```

Arguments

le	latent heat (W m-2)
temp	temperature (degC, default 20 degC)
rho_w	

Value

The function returns evapotranspiration (mm day-1)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

FAO (1998): Crop evapotranspiration - Guidelines for computing crop water requirements - FAO Irrigation and drainage paper 56, http://www.fao.org/docrep/x0490e/x0490e04.htm

See Also

WriteLPJinput

Examples

```
# Example from FAO (1998)
le <- 12 # latent heat that is used to vapourize water (MJ m-2 day-1)
le <- le / 86400 # MJ m-2 day-1  ->  MJ m-2 sec-1
le <- le * 1E6 # MJ m-2 sec-1 -> W m-2
LE2ET(le=le)

temp <- -30:40
et <- LE2ET(le=le, temp=temp)
plot(temp, et)
```

LegendBar *Add a colour legend bar to a plot*

Description

Adds a colour legend bar to a plot

Usage

```
LegendBar(x, y, brks = seq(0, 1, by = 0.2), cols = NULL, brks.txt = NULL,
    title = "", srt = NULL, col.txt = "black", cex.txt = 1, ...)
```

Arguments

x	x coordinates for the legend bar
y	y coordinates for the legend bar
brks	class breaks for the legend bar
cols	colours for each class. If NULL grey scales are used.
brks.txt	text labels for the class breaks. If NULL, 'brks' are used
title	title for the legend bar
srt	rotation of breaks text labels
col.txt	colour for text labels
cex.txt	size of the text labels
...	arguments (unused)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CRSll

Examples

```
plot.new()
LegendBar(x=c(0.1, 0.9), y=c(0.4, 0.6))
LegendBar(x=c(0.1, 0.5), y=c(0.7, 0.8))

brks <- seq(-1, 1, 0.2)
cols <- BreakColors(brks)
LegendBar(x=c(0.6, 1), y=c(0.7, 0.8), brks=brks, cols=cols, title="My title")

LegendBar(x=c(0.2, 0.8), y=c(0.1, 0.2), brks=brks, cols=cols, col.txt="purple", title="purple", srt=90)
```

LPJ2NCDF *Convert binary LPJmL model output files to NetCDF*

Description

The function converts a binary LPJmL output file to NetCDF

Usage

```
LPJ2NCDF(file, var.name, var.unit, start = 1982, end = 2011,
    sim.start.year = 1901, var.longname = var.name, run.name = "LPJmL",
   run.description = "LPJmL run", provider = "M. Forkel, matthias.forkel@geo.tuwien.ac.at",
   creator = provider, reference = "Sitch et al. 2003 GCB, Gerten et al. 2004 J. Hydrol., Thonicke
    ...)
```

Arguments

file	file name of LPJmL model output, e.g. "mgpp.bin"
var.name	variable name, e.g. "GPP"
var.unit	variable unit, e.g. "gC m-2 mon-1"
start	first year for which the data should be converted to NetCDF
end	last year for which the data should be converted to NetCDF
sim.start.year	first year of the simulation
var.longname	long variable name, e.g. "gross primary production"
run.name	name of the LPJmL run (will be part of the file names)
run.description	description of the LPJmL run
provider	name of the provider
creator	name of the creator
reference	
...	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

`LPJfiles` *Create an object of class 'LPJfiles'*

Description

The function creates a list of class 'LPJfiles' that defines all paths, input files, and configurations files for a LPJ run.

Usage

```
LPJfiles(path.lpj, path.tmp, path.out, sim.start.year, sim.end.year = NA,
    lpj.conf, param.conf, pft.par, param.par, input.conf, input,
    ...)
```

Arguments

`path.lpj`	path where LPJ is installed
`path.tmp`	path for temporary outputs
`path.out`	path for results
`sim.start.year`	start year of the LPJ simulation as defined in lpjml.conf
`sim.end.year`	last year of the LPJ simulation as defined in lpjml.conf
`lpj.conf`	template for LPJ configuration file (create a template from lpjml.conf)
`param.conf`	template for parameter configuration file (create a template from param.conf)
`pft.par`	template file for PFT-specific parameters (create a template from pft.par)
`param.par`	template file for global parameters (create a template from param.par)
`input.conf`	template file for input data (create a template from input.conf)
`input`	a data.frame of LPJ input files with 2 columns. The first coumn defines the flag as in written in the input.conf template file and the second column the file name, e.g. data.frame(name=c("GRID_FILE", "TMP_FILE"), file=c("cru.grid", "temp.bin"))
`...`	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

LPJpar *Create an object of class 'LPJpar'*

Description

The function creates a data.frame of class 'LPJpar' that defines the parameters for LPJ runs.

Usage

```
LPJpar(par.prior, par.lower, par.upper, par.pftspecif, par.names,
    is.int = rep(FALSE, length(par.prior)), ...)
```

Arguments

par.prior	parameter vector (prior)
par.lower	lower boundaries for parameters
par.upper	upper boundaries for parameters
par.pftspecif	Which parameter is PFT specific (TRUE) or global (FALSE)?
par.names	parameter name
is.int	is parameter a integer?
...	further arguments for CheckLPJpar

Details

No details.

Value

The function returns a list of class 'LPJpar'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CheckLPJpar

LPJparList *Create a list of 'LPJpar' objects*

Description

The function creates a list of LPJpar objects that can be used to compare parameters from different optimization experiments

Usage

```
LPJparList(...)
```

Arguments

...	objects of class LPJpar

Details

No details.

Value

The function returns a list of class 'LPJparList'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

LPJpar

LPJpp *Post-process LPJmL model output*

Description

The function converts binary LPJmL output files to NetCDF and calculates summary statistics. Please note, climate data operators (CDO) is required.

Usage

```
LPJpp(path, start = 1982, end = 2011, sim.start.year = 1901,
   run.name = "LPJ", run.description = "LPJ run", provider = "M. Forkel, matthias.forkel@geo.tuwie
   creator = provider, reference = "Sitch et al. 2003 GCB, Gerten et al. 2004 J. Hydrol., Thonicke
    lpj.df = NULL, convert = TRUE, calc.nbp = FALSE, calc.cbalance = FALSE,
    calc.tau = FALSE, calc.et = FALSE, calc.tree = FALSE, pft.istree = 2:9,
    mask = NA, ...)
```

Arguments

path	directory with LPJmL outputs in *.bin format
start	first year for which the data should be converted to NetCDF
end	last year for which the data should be converted to NetCDF
sim.start.year	first year of the simulation
run.name	name of the LPJmL run (will be part of the file names)
run.description	description of the LPJmL run
provider	name of the provider
creator	name of the creator
reference	
lpj.df	A data.frame with information about LPJmL outputs that should be post-processed. If NULL, a set of default outputs will post-processed. See details for the required structure of this data.frame.
convert	Convert files in lpj.df to NetCDF?
calc.nbp	Calculate net biome productivity? NBP = Rh + FireC + HarvestC - (NPP + Estab)
calc.cbalance	Calculate global total C stocks, fluxes, balances, and turnover times?
calc.tau	Calculate spatial fields of turnover times?
calc.et	Calculate evapotranspiration? ET = transp + evap + interc
calc.tree	Calculate total tree cover? See also the argument pft.istree
pft.istree	Which bands in fpc.bin represents tree?
mask	A mask in a NetCDF file in order to compute the C fluxes, stocks, balances and turnover times only for specific regions.
...	further arguments (currently not used)

Details

The data.frame 'lpj.df' should have the following columns

- file name of binary LPJmL output file (e.g. mgpp.bin)
- var.name short name of the variable (e.g. GPP)
- var.unit units of the variables in the input file (e.g. "gC m-2")
- var.longname (optional) long name of the variable (e.g. "Gross primary production"). If this column is not provided 'var.name' will be used instead.
- var.agg.fun (optional) name of a function to aggregate the varibale to annual values (e.g "sum", "mean", "min", "max", or NA (no aggregation)). If this column is not provided, aggregations will be not computed.

- `subset.start` (optional) This can be used to additionally subset the NetCDF files to a shorter time period. Set to a year (e.g. 2000) or NA.
- `subset.end` (optional) This can be used to additionally subset the NetCDF files to a shorter time period. Set to a year (e.g. 2001) or NA.
- `stat.annual` (optional) Set to TRUE to compute statistical values based on annual aggregated data. If this is not provided, statistical values will be not computed
- `stat.monthly` (optional) Set to TRUE to compute statistical values based on monthly data. If this is not provided, statistical values will be not computed

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

`LPJppNBP` *Post-process LPJmL model output: calculate NEE*

Description

The function calculates NEE from LPJmL model output

Usage

```
LPJppNBP(path, start = 1982, end = 2011, sim.start.year = 1901,
    ...)
```

Arguments

`path`	directory with LPJmL outputs in *.bin format
`start`	first year for which the data should be converted to NetCDF
`end`	last year for which the data should be converted to NetCDF
`sim.start.year`	first year of the simulation
`...`	further arguments (currently not used)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

`LWin2LWnet` *Compute long-wave net radiation from long-wave incoming radiation and temperature.*

Usage

```
LWin2LWnet(lwin, temp, emissivity = 0.97)
```

Arguments

`lwin`	long-wave incoming radiation (Wm-2)
`temp`	temperature (degC, conversion to K is done within the function)
`emissivity`	emissivity of the surface. Values around 0.97 are valid for various natural surface types (leaves 0.94-0.99, soil 0.93-0.96, water 0.96) (Campbell and Norman 1998, p. 162-163, 177).

Details

Long-wave net radiation is computed as the difference between long-wave incoming and long-wave outgoing radiation. Long-wave outgoing radiation is computed based on the Stefan-Boltzmann law and an emissivity factor (LWout = emissivity * sigma * temp^4), whereas sigma is the Stefan-Boltzmann constant (5.67037 * 10^(-8) Wm-2 K-4) (Campbell and Norman 1998, p. 162-163, 177).

Value

The function returns long-wave net radiation (Wm-2)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

Campbell GS, Norman JM (1998) An Introduction to Environmental Biophysics. Springer New York, New York, NY.

See Also

`WriteLPJinput`

Examples

```
lwin <- 200:380 # long-wave incoming radiation (Wm-2)
temp <- 0.14 * lwin - 32 + rnorm(length(lwin), 0, 5) # temperature (degC)
plot(lwin, temp)
lwnet <- LWin2LWnet(lwin, temp)
plot(temp, lwnet)
plot(lwin, lwnet)
```

MeanW *Weighted mean*

Description

Compute the weighted mean.

Usage

```
MeanW(x, w = rep(1, length(x)))
```

Arguments

x	a vector
w	vector of weights

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

ObjFct

Examples

```
x <- 1:5
mean(x)
MeanW(x, w=c(1, 1, 1, 2, 2))
```

`OptimizeLPJgenoud` *Optimize LPJ using the GENOUD optimizer (genetic optimization using derivatives)*

Description

This function performs an optimization of LPJmL model parameters for the specified grid cells unsing the GENOUD genetic optimization algorithm.

Usage

```
OptimizeLPJgenoud(xy, name, lpjpar, par.optim, lpjfiles, lpjcmd = "srun ./bin/lpjml",
    copy.input = TRUE, integrationdata, plot = TRUE, pop.size = 1000,
    max.generations = 20, wait.generations = 19, BFGSburnin = 18,
    calc.jacob = FALSE, restart = 0, path.rescue = NULL, restart.jacob = FALSE,
    nodes = 1, maxAutoRestart = 5, runonly = FALSE, warnings = TRUE,
    new.spinup4post = TRUE, CostMDS = CostMDS.SSE)
```

Arguments

`xy`	matrix of grid cell coordinates to run LPJ
`name`	name of the experiment (basic file name for all outputs)
`lpjpar`	data.frame of class `LPJpar` that define all LPJ parameter values, ranges, and names
`par.optim`	names of the parameters that should be optimized
`lpjfiles`	list of class `LPJfiles` that define all LPJ directories, input files, configuration template files
`lpjcmd`	How you usually run the LPJ model at the console: 'srun ./bin/lpjml' or './bin/lpjml'
`copy.input`	Should LPJ input data be copied to the directory for temporary output? This might speed up computations if the directory is on the same machine where the program runs.
`integrationdata`	list of integration data and information
`plot`	plot diagnostic graphics of optimization results?
`pop.size`	population size, see `genoud`
`max.generations`	max number of generations, see `genoud`
`wait.generations`	How many generations should genoud wait before returning an optimum, see `genoud`
`BFGSburnin`	The number of generations before the L-BFGS-B algorithm is first used, see `genoud`
`calc.jacob`	Should the Hessian and Jacobian matrix be computed (yes = TRUE, no = FALSE)?
`restart`	Where to re-start the optimization? 0 = start at the beginning, 1 = continue with existing genoud optimization, 2 = start after genoud and post-process results.
`path.rescue`	directory where the resuce files from each iteration of a previous optimization are saved. This is needed if restart > 0.

restart.jacob	Should the Hessian and Jacobian matrix be recomputed if restart > 0 (yes = TRUE, no = FALSE)? Works only if calc.jacob is TRUE.
nodes	use parallel computing? How many nodes to use?
maxAutoRestart	maximum number of automatic restarts of the optimization if an error occurs within genoud()
runonly	run only the model with prior parameters set but don't perform optimization. Produces only results of prior model run.
warnings	print all LPJmL warning messages during optimization?
new.spinup4post	What spinup conditions should be used for the posterior ('posterior-best' and 'posterior-median') model runs? If TRUE, a new spinup is computed based on the optimized parameters. If FALSE, the posterior model runs are started from the spinup conditions of the prior model run (like the runs during optimization).
CostMDS	cost function for multiple data streams to calculate total cost, cost per data stream, and eventually cost per grid cell. See CostMDS.SSE (default) or CostMDS.KGE

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

---

plot.Cbalance *Plots a C balance*

Description

The function takes an object of class Cbalance and creates time series plots or barplots.

Usage

```
## S3 method for class 'Cbalance'
plot(x, what = NULL, trend = TRUE, baseunit = "PgC",
    ylab = NULL, ...)
```

Arguments

x	object of class Cbalance
what	Which variables of C balance to plot? If NULL, sole plots are generated automatically.
trend	Compute trends?
baseunit	unit of C stocks
ylab	labels for y-axis
...	further arguments (currently not used)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

Examples

```
# with some typical numbers for the global C budget:
cbal <- Cbalance(gpp=123, npp=61, rh=57, firec=2, vegc=400, soilc=2400)
cbal
plot(cbal)

## using time series::
#cbal <- Cbalance(gpp=118:128, npp=(118:128)*rnorm(11, 0.5, 0.1), rh=57, firec=runif(11, 0, 4), harvest=2, ve
#cbal
#plot(cbal)
```

`plot.IntegrationData` *Plot an object of class IntegrationData*

Description

The function plots an object of class `IntegrationData`, i.e. it produces a time series plots, scatterplots and a boxplot for the observations and LPJmL model outputs in `IntegrationData`.

Usage

```
## S3 method for class 'IntegrationData'
plot(x, ds = 1:length(x), CostMDS = CostMDS.SSE,
    fits = "poly3", ...)
```

Arguments

`x`	object of class `IntegrationData`
`ds`	Which data sets in x should be plotted (integer)
`CostMDS`	cost function for multiple data streams
`fits`	Fitting methods that should be used for scatter plots, see `MultiFit`
`...`	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`IntegrationData`

`plot.LPJpar` *Plot parameters in 'LPJpar' object.*

Usage

```
## S3 method for class 'LPJpar'
plot(x, par.name = NULL, uncertainty = "uncertainty.005",
    unc.change = FALSE, col = NULL, ylim = NULL, xlim = NULL,
    which.pft = NULL, if.opt = FALSE, names = FALSE, opt.val = TRUE,
    xaxt = "s", add = FALSE, xoff = 0)
```

Arguments

`x`	object of class 'LPJpar'
`par.name`	name(s) of the parameters that should be plotted
`uncertainty`	name of the uncertainty estimate in LPJpar that should be used to plot posterior uncertainties
`unc.change`	plot the change in uncertainty? If TRUE the function plots the fraction of the posterior uncertainty relative to the prior, i.e. uncertainty / abs(upper - lower)
`col`	vector of colours for PFT-specific parameters
`ylim`	limits of the y-axis
`xlim`	limits of the x-axis
`which.pft`	character vector of PFT names that should be plotted. If NULL all
`if.opt`	plot parameters only if optimized (i.e. best) parameters are in LPJpar
`names`	plot PFT names within the plot?
`opt.val`	plot value of optimized parameter?
`xaxt`	x axis type. "n" suppresses the x axis.
`add`	add to existing plot?
`xoff`	offset for adjusting in x-direction

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`LPJpar`, `CheckLPJpar`

Examples

```
# plot(lpjpar, par.name="ALBEDO_LEAF_TeBS", uncertainty="uncertainty.iqr95")
# plot(lpjpar, par.name="ALBEDO_LEAF", uncertainty="uncertainty.iqr")
# plot(lpjpar, par.name="LIGHTEXTCOEFF", uncertainty="uncertainty.iqr")
# par(mfrow=c(2,2))
# plot(lpjpar, par.name=c("ALPHAA", "LIGHTEXTCOEFF", "ALBEDO_LEAF", "ALBEDO_STEM"))
```

`plot.LPJparList` *Plots to compare LPJpar objects*

Description

The function takes a `LPJparList` object and creates a plot to compare optimized parameters

Usage

```
## S3 method for class 'LPJparList'
plot(x, par.name = NULL, ...)
```

Arguments

`x`	object of class 'LPJparList'
`par.name`	name(s) of the parameters that should be plotted
`...`	further arguments to `plot.LPJpar`

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`LPJpar`, `plot.LPJpar`

`plot.LPJsim` *Plots a LPJsim object*

Description

The function plots a LPJsim object: monthly, annual time series or map of grid cells

Usage

```
## S3 method for class 'LPJsim'
plot(x, what = "annual", start = NA, end = NA, omit0 = TRUE,
    AggFun = AggMeanNULL, ...)
```

Arguments

`x`	an object of class 'LPJsim'
`what`	What type of plot should be created? 'annual' for yearly time series, 'monthly' for monthly time series, 'daily' for daily time series, and 'grid' for a map of grid cells
`start`	first year for time series plot
`end`	last year for time series plot
`omit0`	omit variables from plotting that are only 0?
`AggFun`	aggregation function to aggregate results to the temporal resolution as selected in 'what', for example `AggMeanNULL` for monthly or annual means, `AggSumNULL` for monthly or annual sums.
`...`	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`ReadLPJ2ts`, `ReadLPJsim`

Examples

```
# setwd(path.mylpjresult)
# sim <- ReadLPJ2ts(start=1982, end=2011)
# plot(sim, what="annual")
```

`plot.rescue` *plot an object of class "rescue" / monitor OptimizeLPJgenoud*

Description

The function plots the cost per data set for all individuals of the genetic optimization from an object of class "rescue". This function can be used to monitor the development of the optimization within OptimizeLPJgenoud. Therefor read the rescue files from your optimization with "rescue.l <- CombineRescueFiles(list.files(pattern=".RData"), remove=FALSE)" and call "plot(rescue.l)".

Usage

```
## S3 method for class 'rescue'
plot(x, ylim = NULL, xlim = NULL, ylab = "Cost", xlab = "Individuals of genetic optimization",
    only.cost = FALSE, ...)
```

Arguments

`x`	a list of class "rescue", see `CombineRescueFiles`
`ylim`	limits of the y-axis of the plot
`xlim`	limits of the x-axis of the plot
`ylab`	label for the y axis
`xlab`	label for the x axis
`only.cost`	plot all integration datasets (TRUE) or only these ones with cost=TRUE
`...`	further arguments for `plot`

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`CombineRescueFiles`

PlotPar *Plot parameter vs. cost*

Description

The function takes an object of class 'rescue' (see CombineRescueFiles) (alternatively a 'data.frame' as created with Rescue2Df) and a 'LPJpar' object and plots different plots of cost vs. parameter value and parameter uncertainties.

Usage

```
PlotPar(rescue.l, lpjpar, par.name = NULL, ...)
```

Arguments

rescue.l	a list of class "rescue" (CombineRescueFiles) or alternatively a data.frame as created with Rescue2Df.
lpjpar	a list of class "LPJpar" (see LPJpar)
par.name	name(s) of the parameters that should be plotted
...	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles

Examples

```
# files <- c(list.files(pattern="rescue.RData", recursive=TRUE), list.files(pattern="rescue0.RData", recursi
# rescue.l <- CombineRescueFiles(files, remove=FALSE)
# PlotPar(rescue.l, lpjpar)
```

PlotParPCA *plot a PCA of optimized parameters*

Description

The function takes an object of class 'rescue' (see CombineRescueFiles), computes a PCA (principle component analysis) based on the model parameter sets and cost function values of the optimization, and plots PCA results as biplots.

Usage

```
PlotParPCA(rescue.l, ...)
```

Arguments

rescue.l	a list of class "rescue", see CombineRescueFiles
...	further arguments for plot

Details

No details.

Value

The function returns an object of class 'princomp'.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles, princomp

PlotParUnc *Plot the psoterior parameter uncertainty*

Description

The function takes an object of class LPJpar and plots the relative uncertainty of the optimized parameters, i.e. uncertainty_best / uncertainty_prior

Usage

```
PlotParUnc(lpjpar, uncertainty = "uncertainty.005", ylab = "Relative parameter uncertainty",
    main = NULL, par.name = NULL, use.par = TRUE, legend = TRUE,
    srt = 40, cex = 1, ...)
```

Arguments

lpjpar	a list of class "LPJpar" (see LPJpar)
uncertainty	name of the uncertainty estimate in LPJpar that should be used to compute posterior uncertainties
ylab	label of y-axis
main	title of plot
par.name	name(s) of the parameters that should be plotted
use.par	use default settings for the graphic window? If FALSE, the internal settings for par() are not used.
legend	plot a legend for PFTs?
srt	string rotation of parameter names at the x-axis
cex	size of point symbols
...	further arguments for plot

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles

Examples

```
# PlotParUnc(lpjpar)
```

PlotWorld110 *Plot a world map based on 1:110Mio data*

Usage

```
PlotWorld110(admin = FALSE, lakes = TRUE, rivers = TRUE, col = c("black",
    "blue", "red"), bg = NA, ...)
```

Arguments

admin	Plot administrative boundaries?
lakes	Plot lakes?
rivers	Plot rivers?
col	Colors for (1) coastlines, (2) rivers and (3) administrative boundaries
bg	background color, default: NA (no background)
...	additional arguments to plot

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

Examples

```
PlotWorld110()
```

PrepareRestartFiles *Prepare restart files to restart OptimizeLPJgenoud*

Description

The function prepares all files that are needed to restart OptimizeLPJgenoud

Usage

```
PrepareRestartFiles(file.optsetup, ...)
```

Arguments

file.optsetup	OptimizeLPJgenoud setup file, ends with "_optsetup.RData"
...	further arguments (currently not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

LPJfiles

ReadBIN — *Read simple binary files without header Read a CLM file to a SpatialPixelsDataFrame*

Description

The function is used to read for example the soil*.bin and drainclass.bin files

Usage

```
ReadBIN(file.bin, nbands = 1, size = 1, file.grid = NA, endian.data = "little",
    endian.grid = "little", data.type = integer(), ...)
```

Arguments

file.bin	binary file name with extension *.bin
nbands	number of bands per year
size	The number of bytes per element in the byte stream.
file.grid	file name of the corresponding grid file
endian.data	endinaess of the data file
endian.grid	endianess of the grid file
data.type	type of the data
...	Further arguments (currently not used).

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

WriteCLM

Examples

```
# ReadBIN("soil_new_67420.bin")
```

ReadCLM *Read a CLM file to a SpatialPixelsDataFrame*

Usage

```
ReadCLM(file.clm, start = NA, end = NA, start.year = NA, grid = NULL,
    nbands = NA, size = NA, file.grid = NA, endian.data = NA,
    endian.grid = "big", data.type = integer(), ...)
```

Arguments

file.clm	CLM file name with extension *.clm
start	first year to be read
end	last year to be read, reads until last year in case of NA
start.year	first year in the dataset, read from header information in case NA
grid	a matrix of coordinates (lon, lat) if data should be read only for specific cells, if NULL the data for all grid cells is read
nbands	number of bands per year, read from header information in case NA
size	The number of bytes per element in the byte stream.
file.grid	file name of the corresponding grid file
endian.data	endinaess of the data file
endian.grid	endianess of the grid file
data.type	type of the data
...	Further arguments (currently not used).

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

WriteCLM

ReadGrid *Reads a binary input data grid file.*

Usage

```
ReadGrid(file.grid = "cru.grid", endian = "little", ...)
```

Arguments

file.grid	CLM file name with extension *.clm
endian	endianess of the file
...	Further arguments (currently not used).

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

ReadCLM

ReadLPJ *Read a LPJ binary file*

Description

The function reads a binary LPJ output file and returns a SpatialPointsDataFrame

Usage

```
ReadLPJ(file.bin, file.grid = "grid.bin", sim.start.year = 1901,
    start = sim.start.year, end = NA, file.annual = c("vegc.bin",
        "litc.bin", "soilc.bin"), size = 4, data.type = numeric(),
    endian = "little", ...)
```

Arguments

file.bin	binary LPJ output file
file.grid	binary LPJ grid file
sim.start.year	first year of the simulation
start	first year to read
end	last year to read, reads until last year if NA
file.annual	one of the binary LPJ output files with annual data
size	the number of bytes per element in the byte stream.
data.type	data type of the file (default=numeric())
endian	endianess of the binary file
...	further arguments (currently not used)

Details

No details.

Value

The function returns a SpatialPointsDataFrame.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

ReadLPJsim

Examples

```
# ReadLPJ("mgpp.bin", start=1982, end=2011)
```

`ReadLPJ2IntegrationData` *Read LPJ model results into an of class IntegrationData*

Description

The function reads for each dataset in `IntegrationData` the corresponding model output and performs temporal aggregation.

Usage

```
ReadLPJ2IntegrationData(integrationdata, xy, lpjfiles, ...)
```

Arguments

`integrationdata`	object of class `IntegrationData`
`xy`	matrix of grid cell coordinates to run LPJ
`lpjfiles`	list of class `LPJfiles` that define all LPJ directories, input files, configuration template files
`...`	further arguments (currently not used)

Details

No details.

Value

The function returns the same list oc class 'IntegrationData' but with added model outputs.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`IntegrationData`

ReadLPJ2ts *Read a LPJ binary file and returns a spatial averaged time series*

Description

The function reads LPJ binary output files *.bin, aggregates (mean) the time series over all grid cells and returns the regional-averaged time series

Usage

```
ReadLPJ2ts(file.bin, sim.start.year = 1901, start = sim.start.year,
    end = NA, ...)
```

Arguments

file.bin	binary LPJ output file
sim.start.year	first year of the simulation
start	first year to read
end	last year to read, reads until last year if NA
...	further arguments (currently not used)

Details

No details.

Value

The function returns a time series of class 'ts'.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

ReadLPJsim

Examples

```
# gpp <- ReadLPJ2ts("mgpp.bin")
```

`ReadLPJinput` *Read and subset CLM files to LPJinput objects*

Description

The functions reads a CLM file, selects the data according to the provided grid and returns an object of class LPJinput.

Usage

```
ReadLPJinput(files, grid = NULL, start = NA, ...)
```

Arguments

`files`	character vector of CLM or binary file names
`grid`	Matrix of grid cells with 2 columns: longitude and latitude (optional). If NULL the data is returned for the grid of the first CLM file. If a grid is provided the data is subesetted for the specified grid cells.
`start`	first year to read
`...`	Further arguments to ReadCLM or ReadBIN

Details

No details.

Value

The function returns a list of class "LPJinput".

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`WriteLPJinput`

Examples

```
# lpjinput <- ReadLPJinput("cru_ts_3.20.1901.2011.tmp.clm", grid=cbind(c(136.75, 137.25, 160.75,168.75), c(4
# str(lpjinput)
```

`ReadLPJsim` *Read a LPJ simulation results*

Description

The function reads all binary output files from a LPJ simulation and returns regional aggregated time series.

Usage

```
ReadLPJsim(sim.start.year = 1901, start = sim.start.year, end = NA,
    files = NA, outputvars.par = NULL, ...)
```

Arguments

`sim.start.year`	first year of the simulation
`start`	first year to read
`end`	last year to read. If NA, reads until last year
`files`	Which LPJ binary output files should be read? If NA, all *.bin files in the current directory are read.
`outputvars.par`	path and file name to the LPJmL 'outputvars.par' file. If NULL the file is searched 1 level above or below the current working directory.
`...`	further arguments (currently not used)

Details

No details.

Value

The function returns a list of class 'LPJsim'

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`ReadLPJ2ts`

Examples

```
# setwd(path.mylpjresult)
# sim <- ReadLPJsim(start=1982, end=2011)
```

`ReadOutputvars` *Read 'outputvars.par' to get information about LPJmL output*

Description

LPJmL output is defined in par/outputvars.par. This file contains for each variable the id, name, variable name, description, unit, and scale. This file can be used to correctly read LPJmL output. The function is for example used within `ReadLPJsim`.

Usage

```
ReadOutputvars(outputvars.par = NULL, ...)
```

Arguments

`outputvars.par`	path and file name to the LPJmL 'outputvars.par' file. If NULL the file is searched 1 level above or below the current working directory.
`...`	further arguments (currently not used)

Details

No details.

Value

The function returns a time series of class 'ts'.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`ReadLPJsim`

Examples

```
# ReadOutputvars()
```

ReadPRO — *Read *.pro files as produced from GENOUD*

Description

The function is used within OptimizeLPJgenoud

Usage

```
ReadPRO(files.pro)
```

Arguments

files.pro	file names (*.pro) of genoud optimization results.

Details

No details.

Value

The function returns a data.frame with number of individual, cost and parameer values

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

RegridLPJinput — *Regrid or subset LPJmL input*

Description

Subsets grid cells or regrids LPJmL input files.

Usage

```
RegridLPJinput(files, grid.clm, grid, path.out, overwrite = TRUE,
    ...)
```

Arguments

files	character vector of CLM or binary file names
grid.clm	old grid *.clm file
grid	Matrix of new grid cells with 2 columns: longitude and latitude (optional). If NULL the data is returned for the grid of the first CLM file. If a grid is provided the data is subesetted for the specified grid cells.
path.out	directory where the new files should be saved
overwrite	overwrite existing files?
...	further arguments (currently not used)

Details

No details.

Value

The function returns TRUE if the CLM file was created.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

WriteLPJinput

Examples

```
# no example
```

Rescue2Df *Convert a 'rescue' list to a data.frame*

Description

The function takes an object of class 'rescue' (see CombineRescueFiles) and converts it to a data.frame including the total cost value (1st column), the parameter values (next columns), and the log-likelihood, Akaike's Information Criterion (AIC) and AIC differences (last columns). If 'lpjpar' it not specified the function returns just the scaled parameters (e.g. dpar = par / prior) otherwise it returns the parameters in the original units (e.g. par = dpar * prior).

Usage

```
Rescue2Df(rescue.l, lpjpar = NULL, ...)
```

Arguments

rescue.l	a list of class "rescue", see CombineRescueFiles
lpjpar	a list of class "LPJpar" (see LPJpar) to convert the scaled parameters in rescue.l back to the original units (optional)
...	further arguments (currently not used)

Details

No details.

Value

The function returns a data.frame.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles

Examples

```
# files <- c(list.files(pattern="rescue.RData", recursive=TRUE), list.files(pattern="rescue0.RData", recursi
# rescue.l <- CombineRescueFiles(files, remove=FALSE)
# optim.df <- Rescue2Df(rescue.l)
# summary(optim.df)
```

Rescue2LPJpar *Add information from a 'rescue' list to an 'LPJpar' object*

Description

The function takes an object of class 'rescue' (see CombineRescueFiles) (alternatively a 'data.frame' as created with Rescue2Df) and a 'LPJpar' (see LPJpar) object. Then it extracts the best parameter set, the median of the best parameter sets (defined based on dAIC <= 2), and various uncertainty measures and adds them to the 'LPJpar' object.

Usage

```
Rescue2LPJpar(rescue.l, lpjpar, ...)
```

Arguments

rescue.l	a list of class "rescue" (CombineRescueFiles) or alternatively a data.frame as created with Rescue2Df.
lpjpar	a list of class "LPJpar" (LPJpar)
...	further arguments (currently not used)

Details

No details.

Value

The function returns the provided 'LPJpar' object with the following additional slots:

- best Best parameter set
- best.median median of best parameter sets (based on all parameter sets with dAIC <= 2)
- uncertainty.iqr uncertainty of parameters as the inter-quartile range of the best parameter sets
- uncertainty.iqr95 uncertainty of parameters as the central 95% inter-quantile range (0.975-0.025) of the best parameter sets
- uncertainty.005.min lower parameter uncertainty estimate based on the minimum parameter value from all parameter sets for which the cost <= quantile(cost, 0.05)
- uncertainty.005.max upper parameter uncertainty estimate based on the maximum parameter value from all parameter sets for which the cost <= quantile(cost, 0.05)

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

CombineRescueFiles

Examples

```
# files <- c(list.files(pattern="rescue.RData", recursive=TRUE), list.files(pattern="rescue0.RData", recursi
# rescue.l <- CombineRescueFiles(files, remove=FALSE)
# lpjpar2 <- Rescue2LPJpar(rescue.l, lpjpar)
# str(lpjpar2)
# plot(lpjpar2, "ALPHAA", "uncertainty.iqr95")
# plot(lpjpar2, "TMIN_BASE", "uncertainty.iqr95")
```

RunLPJ *Run LPJmL from R and get results*

Description

This function calls LPJmL, reads the results of the model run, computes the cost based on the data sets in IntegrationData and the defined cost function (in CostMDS), and returns the simulation results.

Usage

```
RunLPJ(dpar, lpjpar, which.par.opt, lpjfiles, path = NULL, integrationdata,
    xy, newcell = FALSE, name = "LPJmL", lpjcmd = "srun ./bin/lpjml",
    plot = FALSE, getresult = FALSE, clean = 1, clean.path = FALSE,
    CostMDS = CostMDS.SSE, nkeep = 400, warnings = TRUE)
```

Arguments

dpar	vector of scaling factors for each parameter in 'which.par.opt': parameter = dpar * prior (e.g. if dpar is 1, prior parameters are used in the model run). Optimization is performed on these scaling factors
lpjpar	data.frame of class LPJpar that define LPJ parameter values, ranges, and names
which.par.opt	integer vector that indicates which parameters in lpjpar should be optimized
lpjfiles	list of class LPJfiles that define all LPJ directories, input files, configuration template files
path	path for output files of actual model run
integrationdata	list of of class IntegrationData
xy	matrix of grid cell coordinates to run LPJ
newcell	calculate new cell and new spinup?
name	name of the LPJ run, basic name for all outputs
lpjcmd	How you usually run the LPJ model at the console: 'srun ./bin/lpjml' or './bin/lpjml'
plot	plot results? see plot.IntegrationData
getresult	If TRUE, all model results are returned in a LPJsim object and model results are saved. If FALSE, only the cost function value is returned.
clean	clean results and temporay configuration and parameter files? 0 = keep everything; 1 = delete parameter files, conf files and outputs; 2 = clean additionally input files, soil code files and restart
clean.path	Delete output directory 'path' in case it already exists before the model run?
CostMDS	cost function for multiple data streams
nkeep	number of result files to keep. If more are existing, the ones with highest costs will be deleted
warnings	print all LPJmL warning messages during optimization?

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

SdW *Weighted standard deviation*

Description

Compute the standard deviation.

Usage

```
SdW(x, w = rep(1, length(x)))
```

Arguments

x	a vector
w	vector of weights

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

ObjFct

Examples

```
x <- 1:5
sd(x)
SdW(x, w=c(1, 1, 1, 2, 2))
```

SSE *Sum-of-squared residuals error*

Description

The function implements the sum-of-squared residuals error as cost function

Usage

```
SSE(sim, obs, unc)
```

Arguments

sim	vector of simulations
obs	vector of observations
unc	vector of observation uncertainties

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

Examples

```
obs <- rnorm(10, 0, 2)
sim <- obs + rnorm(10, 0.05, 0.01)
unc <- 0.01
SSE(sim, obs, unc)
```

StandardError *Compute standard errors from a variance-covariance matrix*

Description

SE = sqrt(diag(vc) * cost^2 / (nobs - npar))

Usage

```
StandardError(vc, nobs, cost)
```

Arguments

vc	variance-covariance matrix
nobs	number of observations
cost	cost function value

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

StartingValues *Get starting values for genoud from *.pro file*

Description

The function extracts the best individuals trhat occured during a genoud optimization from a *.pro file. These best individuals can be used as starting values if a optimization is restarted. This function is called within OptimizeLPJgenoud is a restart is performed.

Usage

```
StartingValues(file.optresult, pop.size = NULL, ...)
```

Arguments

file.optresult	genoud *.pro file with optimization results
pop.size	population size
...	further arguments (not used)

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

OptimizeLPJgenoud

`Texture2Soilcode` *Convert soil texture to a LPJ soilcode*

Description

The function takes fractions/percentages of sand, silt and clay and returns the correspondign LPJ soil code. The USDA soil classification is used. The function requires the package "soiltexture".

Usage

```
Texture2Soilcode(sand, silt, clay, lpj.soilcodes = c("Cl", "SiCl",
    "SaCl", "ClLo", "SiClLo", "SaClLo", "Lo", "SiLo", "SaLo",
    "Si", "LoSa", "Sa"), plot = TRUE, ...)
```

Arguments

`sand`	percentage of sand
`silt`	percentage of silt
`clay`	percentage of clay
`lpj.soilcodes`	LPJ soil codes
`plot`	plot soil triangle?
`...`	Further arguments (currently not used).

Details

No details.

Value

The function returns the LPJ soilcode

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`ReadBIN`

Examples

```
# data.sp <- SpatialPointsDataFrame(lpjinput$grid, as.data.frame(data.m))
# WriteBIN(data.sp, file="data.bin")
```

Turnover *Calculate turnover time from stock and flux*

Description

Calculates turnover times.

Usage

```
Turnover(stock, flux, ...)
```

Arguments

stock	stock, e.g. biomass
flux	flux, e.g. NPP
...	further arguments (currently not used)

Details

turnover = stock / flux

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

VarCovMatrix *Compute variance-covariance matrix*

Description

The function computes the variance-covariance matrix from the hessian matrix. Parameters that have a hessian = 0 (in sensitive parameters) area removed from the matrix before calculating the variance-covariance matrix.

Usage

```
VarCovMatrix(hessian, nms = paste("P", 1:n, sep = ""))
```

Arguments

hessian	Hessian matrix
nms	names of the parameters (rows and columns in the matrix

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

`VarW` *Weighted variance*

Description

Compute the weighted variance.

Usage

```
VarW(x, w = rep(1, length(x)))
```

Arguments

`x`	a vector
`w`	vector of weights

Details

No details.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`ObjFct`

Examples

```
x <- 1:5
var(x)
VarW(x, w=c(1, 1, 1, 2, 2))
```

`WriteBIN` *Write a BIN file from SpatialPointsDataFrame*

Description

The function writes BIN files from a SpatialPointsDataFrame or SpatialPixelsDataFrame.

Usage

```
WriteBIN(data.sp, file.bin, size = 1, ...)
```

Arguments

`data.sp`	SpatialPointsDataFrame or SpatialPixelsDataFrame with data
`file.bin`	binary file name with extension *.bin
`size`	The number of bytes per element in the byte stream.
`...`	Further arguments (currently not used).

Details

No details.

Value

The function returns TRUE if the CLM file was created.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`ReadBIN`

Examples

```
# data.sp <- SpatialPointsDataFrame(lpjinput$grid, as.data.frame(data.m))
# WriteBIN(data.sp, file="data.bin")
```

`WriteCLM` *Write a CLM file from SpatialPointsDataFrame*

Description

The function writes CLM files from a SpatialPointsDataFrame or SpatialPixelsDataFrame. The LPJmL program cru2clm needs to be installed.

Usage

```
WriteCLM(data.sp, file.clm, start, nbands, size = 2, scale = 1,
    na.replace = -9999, path.lpj = NULL, res = 0.5, ...)
```

Arguments

`data.sp`	SpatialPointsDataFrame or SpatialPixelsDataFrame with data
`file.clm`	CLM file name with extension *.clm
`start`	integer. First year in data.
`nbands`	Number of bands per year.
`size`	The number of bytes per element in the byte stream.
`scale`	Scaling factor to be written to the header of the CLM file. The factor will be not applied to the data.
`na.replace`	integer to replace NA values.
`path.lpj`	path to LPJ installation
`res`	spatial resolution of the grid cells
`...`	Further arguments (currently not used).

Details

No details.

Value

The function returns TRUE if the CLM file was created.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`WriteLPJinput`

Examples

```
# data.sp <- SpatialPointsDataFrame(lpjinput$grid, as.data.frame(data.m))
# WriteCLM(data.sp, file="data.clm", start=1901, nbands=12, size=2)
```

`WriteGrid` *Write a *.grid file from a matrix of coordiantes or a SpatialPoints-DataFrame*

Description

Writes a grid file for LPJ input data. The functions needs the LPJmL module txt2grid to be installed.

Usage

```
WriteGrid(grid, file.grid, ...)
```

Arguments

`grid`	SpatialPointsDataFrame; SpatialPixelsDataFrame, matrix or data.frame with coordinates
`file.grid`	Grid file name
`...`	Further arguments (currently not used).

Details

No details.

Value

The function returns TRUE if the grid file was created.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`WriteLPJinput`

Examples

```
lon <- c(59.75, 68.25)
lat <- c(61.25, 65.75)
WriteGrid(cbind(lon, lat), "test.grid")
```

`WriteLPJinput` *Write an object of class 'LPJinput' to CLM files*

Description

The function writes CLM input files for LPJ.

Usage

```
WriteLPJinput(lpjinput, files = NULL, path.lpj = NULL, ...)
```

Arguments

`lpjinput`	Object of class 'LPJinput' to be written.
`files`	names of the output CLM or binary files.
`path.lpj`	path to LPJ installation
`...`	further arguments (currently not used)

Details

No details.

Value

The function returns TRUE if the CLM file was created.

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`WriteLPJinput`

Examples

```
# lpjinput <- ReadLPJinput("cru_ts_3.20.1901.2011.tmp.clm", grid=cbind(c(136.75, 137.25, 160.75,168.75), c(4
# str(lpjinput)
# WriteLPJinput(lpjinput)
```

`WriteLPJpar` *Writes an object of class 'LPJpar' as parameter file or table.*

Description

The function takes a 'LPJpar' object and writes 1) LPJ parameter files and 2) write *.txt files with parameter values in a table format.

Usage

```
WriteLPJpar(x, file = "LPJpar", pft.par = NULL, param.par = NULL,
    param.only = TRUE, ...)
```

Arguments

`x`	object of class 'LPJpar'
`file`	basic file name for all output files, e.g. name of the optimization experiment
`pft.par`	template file for PFT-specific parameters (create a template from pft.par). If NULL, parameter files will be not written but only parameter tables.
`param.par`	template file for global parameters (create a template from param.par). If NULL, parameter files will be not written but only parameter tables.
`param.only`	write only parameters to table (TRUE) or also parameter prior ranges (FALSE)?
`...`	further arguments for CheckLPJpar

Details

No details.

Value

The function returns a data.frame with an overview of the written files

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

References

No reference.

See Also

`LPJpar`, `CheckLPJpar`

`WriteNCDF4` *Write NetCDF files*

Description

Writes NetCDF files from rasters and makes sure that meta-information is properly defined.

Usage

```
WriteNCDF4(data.l, var.name, var.unit, time = as.Date("2000-01-01"),
    var.description = var.name, file = NULL, data.name = NA,
    region.name = NA, file.title = var.name, file.description = var.name,
    reference = "", provider = "", creator = "", missval = -9999,
    scale = 1, offset = 0, compression = 9, overwrite = FALSE)
```

Arguments

`data.l`	a single Raster* object or a list of Raster* objects
`var.name`	vector of variable names
`var.unit`	vector of variable units
`time`	vector of time steps for each layer.
`var.description`	vector of variable descriptions
`file`	file name. If NULL the file name will be created from the variable name and the dimensions of the data.
`data.name`	name of the dataset
`region.name`	name of the region
`file.title`	title of the file
`file.description`	description of the file
`reference`	reference for the dataset
`provider`	dataset provider
`creator`	dataset creator
`missval`	flag for missing/NA values
`scale`	scaling values for the data
`offset`	offset value
`compression`	If set to an integer between 1 (least compression) and 9 (most compression), this enables compression for the variable as it is written to the file. Turning compression on forces the created file to be in netcdf version 4 format, which will not be compatible with older software that only reads netcdf version 3 files.
`overwrite`	overwrite existing file?

Author(s)

Matthias Forkel <matthias.forkel@geo.tuwien.ac.at> [aut, cre], Markus Dr<c3><bc>ke <drueke@pik-potsdam.de> [aut]

Index