airGR: Suite of GR Hydrological Models for Precipitation-Runoff Modelling. Manual of the R package version 1.0.10.11

Laurent Coron, Charles Perrin, Olivier Delaigue, Guillaume Thirel, Claude Michel

Package 'airGR'

June 29, 2018

Type Package

Title Suite of GR Hydrological Models for Precipitation-Runoff Modelling

Version 1.0.10.11

Date 2018-06-29

Depends R (>= 3.0.1)

Suggests knitr, rmarkdown, coda, DEoptim, dplyr, FME, ggmcmc, hydroPSO, Rmalschains

Description Hydrological modelling tools developed at Irstea-Antony (HYCAR Research Unit, France). The package includes several conceptual rainfall-runoff models (GR4H, GR4J, GR5J, GR6J, GR2M, GR1A), a snow accumulation and melt model (CemaNeige) and the associated functions for their calibration and evaluation. Use help(airGR) for package description.

License GPL-2

URL https://webgr.irstea.fr/en/airGR/

NeedsCompilation yes

Encoding UTF-8

VignetteBuilder knitr

Author Laurent Coron [aut, trl],
Charles Perrin [aut, ths],
Olivier Delaigue [aut, cre],
Guillaume Thirel [aut],
Claude Michel [aut, ths],
Vazken Andréassian [ctb, ths],
François Bourgin [ctb] ('Parameter estimation' vignettes),
Pierre Brigode [ctb],
Nicolas Le Moine [ctb],
Thibaut Mathevet [ctb],
Safouane Mouelhi [ctb],
Ludovic Oudin [ctb],
Raji Pushpalatha [ctb],
Audrey Valéry [ctb]

Maintainer Olivier Delaigue <airGR@irstea.fr>

Repository CRAN

Date/Publication 2018-06-29 16:38:09 UTC

R topics documented:

airGR *Suite of GR Hydrological Models for Precipitation-Runoff Modelling*

Description

This package brings into R the hydrological modelling tools used at IRSTEA-Antony (HYCAR Research Unit, France), including rainfall-runoff models (**GR4H**, **GR4J**, **GR5J**, **GR6J**, **GR2M**, **GR1A**) and a snow accumulation and melt model (**CemaNeige**). Each model core is coded in FORTRAN to ensure low computational time. The other package functions (i.e. mainly the calibration algorithm and the computation of the efficiency criteria) are coded in R.

Functions and objects

The airGR package has been designed to fulfil two major requirements: facilitate the use by non-expert users and allow flexibility regarding the addition of external criteria, models or calibration algorithms. The names of the functions and their arguments were chosen to this end.

The package is mostly based on three families of functions:
- the functions belonging to the `RunModel` family require three arguments: *InputsModel*, *RunOptions* and *Param*; please refer to help pages `CreateInputsModel` and `CreateRunOptions` for further details and examples;
- the functions belonging to the `ErrorCrit` family require two arguments: *InputsCrit* and *OutputsModel*; please refer to help pages `CreateInputsCrit` and `RunModel` for further details and examples;
- the functions belonging to the `Calibration` family require four arguments: *InputsModel*, *RunOptions*, *InputsCrit* and *CalibOptions*; please refer to help pages `CreateInputsModel`, `CreateRunOptions`, `CreateInputsCrit` and `CreateCalibOptions` for further details and examples.

In order to limit the risk of mis-use and increase the flexibility of these main functions, we imposed the structure of their arguments and defined their class. Most users will not need to worry about these imposed structures since functions are provided to prepare these arguments for them: `CreateInputsModel`, `CreateRunOptions`, `CreateInputsCrit`, `CreateCalibOptions`. However, advanced users wishing to supplement the package with their own models will need to comply with these imposed structures and refer to the package source codes to get all the specification requirements.

Models

Six hydrological models and one snow melt and accumulation model are implemented in airGR. The snow model can also be used alone or with the daily hydrological models, and each hydrological model can either be used alone or together with the snow model.
These models can be called within airGR using the following functions:
- `RunModel_GR4H`: four-parameter hourly lumped hydrological model (Mathevet, 2005)
- `RunModel_GR4J`: four-parameter daily lumped hydrological model (Perrin *et al.*, 2003)
- `RunModel_GR5J`: five-parameter daily lumped hydrological model (Le Moine, 2008)
- `RunModel_GR6J`: six-parameter daily lumped hydrological model (Pushpalatha *et al.*, 2011)
- `RunModel_GR2M`: two-parameter monthly lumped hydrological model (Mouelhi, 2003 ; Mouelhi *et al.*, 2006a)
- `RunModel_GR1A`: one-parameter yearly lumped hydrological model (Mouelhi, 2003 ; Mouelhi *et*

al., 2006b)
- RunModel_CemaNeige: two-parameter degree-day snow melt and accumulation daily model (Valéry *et al.*, 2014)
- RunModel_CemaNeigeGR4J: combined use of GR4J and CemaNeige
- RunModel_CemaNeigeGR5J: combined use of GR5J and CemaNeige
- RunModel_CemaNeigeGR6J: combined use of GR6J and CemaNeige

How to get started

To learn how to use the functions from the airGR package, it is recommended to follow the five steps described below:
1. refer to the help for RunModel_GR4J then run the provided example to assess how to make a simulation;
2. refer to the help for CreateInputsModel to understand how the inputs of a model are prepared/organised;
3. refer to the help for CreateRunOptions to understand how the run options of a model are parametrised/organised;
4. refer to the help for ErrorCrit_NSE and CreateInputsCrit to understand how the computation of an error criterion is prepared/made;
5. refer to the help for Calibration_Michel, run the provided example and then refer to the help for CreateCalibOptions to understand how a model calibration is prepared/made.

For more information and to get started with the package, you can refer to the vignette (`vignette("airGR")`) and go on the airGR website.

References

- Le Moine, N. (2008). Le bassin versant de surface vu par le souterrain : une voie d'amélioration des performances et du réalisme des modèles pluie-débit ?, PhD thesis (in French), UPMC - Cemagref Antony, Paris, France, 324 pp.
- Mathevet, T. (2005). Quels modèles pluie-débit globaux pour le pas de temps horaire ? Développement empirique et comparaison de modèles sur un large échantillon de bassins versants, PhD thesis (in French), ENGREF - Cemagref Antony, Paris, France, 463 pp.
- Mouelhi S. (2003). Vers une chaîne cohérente de modèles pluie-débit conceptuels globaux aux pas de temps pluriannuel, annuel, mensuel et journalier, PhD thesis (in French), ENGREF - Cemagref Antony, Paris, France, 323 pp.
- Mouelhi, S., C. Michel, C. Perrin and V. Andréassian (2006a). Stepwise development of a two-parameter monthly water balance model, Journal of Hydrology, 318(1-4), 200-214, doi:10.1016/j.jhydrol.2005.06.014.
- Mouelhi, S., C. Michel, C. Perrin. & V. Andreassian (2006b). Linking stream flow to rainfall at the annual time step: the Manabe bucket model revisited, Journal of Hydrology, 328, 283-296, doi:10.1016/j.jhydrol.2005.12.022.
- Perrin, C., C. Michel and V. Andréassian (2003). Improvement of a parsimonious model for streamflow simulation, Journal of Hydrology, 279(1-4), 275-289, doi:10.1016/S0022-1694(03)00225-7.
- Pushpalatha, R., C. Perrin, N. Le Moine, T. Mathevet and V. Andréassian (2011). A downward structural sensitivity analysis of hydrological models to improve low-flow simulation, Journal of Hydrology, 411(1-2), 66-76, doi:10.1016/j.jhydrol.2011.09.034.
- Valéry, A., V. Andréassian and C. Perrin (2014). "As simple as possible but not simpler": What is useful in a temperature-based snow-accounting routine? Part 2 - Sensitivity analysis of the Cemaneige snow accounting routine on 380 catchments, Journal of Hydrology, 517(0): 1176-1187,

doi: 1176-1187, doi:10.1016/j.jhydrol.2014.04.058.

`BasinInfo` *Data sample: characteristics of a fictional catchment (L0123001, L0123002 or L0123003)*

Description

R-object containing the code, station's name, area and hypsometric curve of the catchment.

Format

List named 'BasinInfo' containing

- two strings: catchment's code and station's name
- one float: catchment's area in km2
- one numeric vector: catchment's hypsometric curve (min, quantiles 01 to 99 and max) in metres

See Also

`BasinObs`.

Examples

```
library(airGR)
data(L0123001)
str(BasinInfo)
```

`BasinObs` *Data sample: time series of observations of a fictional catchment (L0123001, L0123002 or L0123003)*

Description

R-object containing the times series of precipitation, temperature, potential evapotranspiration and discharges.
Times series for L0123001 or L0123002 are at the daily time step for use with daily models such as GR4J, GR5J, GR6J, CemaNeigeGR4J, CemaNeigeGR5J and CemaNeigeGR6J. Times series for L0123003 are at the hourly time step for use with hourly models such as GR4H.

Format

Data frame named 'BasinObs' containing

- one POSIXct vector: time series dates in the POSIXct format
- five numeric vectors: time series of catchment average precipitation [mm/time step], catchment average air temperature [°C], catchment average potential evapotranspiration [mm/time step], outlet discharge [l/s], outlet discharge [mm/time step]

See Also

BasinInfo.

Examples

```
library(airGR)
data(L0123001)
str(BasinObs)
```

Calibration — *Calibration algorithm that optimises the error criterion selected as objective function using the provided functions*

Description

Calibration algorithm that optimises the error criterion selected as objective function using the provided functions.

Usage

```
Calibration(InputsModel, RunOptions, InputsCrit, CalibOptions, FUN_MOD,
  FUN_CRIT, FUN_CALIB = Calibration_Michel, FUN_TRANSFO = NULL,
  verbose = TRUE)
```

Arguments

InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
InputsCrit	[object of class *InputsCrit*] see CreateInputsCrit for details
CalibOptions	[object of class *CalibOptions*] see CreateCalibOptions for details
FUN_MOD	[function] hydrological model function (e.g. RunModel_GR4J, RunModel_CemaNeigeGR4J)
FUN_CRIT	[function] error criterion function (e.g. ErrorCrit_RMSE, ErrorCrit_NSE)
FUN_CALIB	(optional) [function] calibration algorithm function (e.g. Calibration_Michel), default = Calibration_Michel
FUN_TRANSFO	(optional) [function] model parameters transformation function, if the FUN_MOD used is native in the package FUN_TRANSFO is automatically defined
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Value

list see Calibration_Michel

Author(s)

Laurent Coron

See Also

Calibration_Michel, ErrorCrit, TransfoParam, CreateInputsModel, CreateRunOptions, CreateInputsCrit, CreateCalibOptions.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## calibration period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## calibration criterion: preparation of the InputsCrit object
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])

## preparation of CalibOptions object
CalibOptions <- CreateCalibOptions(FUN_MOD = RunModel_GR4J, FUN_CALIB = Calibration_Michel)

## calibration
OutputsCalib <- Calibration(InputsModel = InputsModel, RunOptions = RunOptions,
                            InputsCrit = InputsCrit,  CalibOptions = CalibOptions,
                            FUN_MOD = RunModel_GR4J, FUN_CRIT = ErrorCrit_NSE,
                            FUN_CALIB = Calibration_Michel)

## simulation
Param <- OutputsCalib$ParamFinalR
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions,
                         Param = Param, FUN = RunModel_GR4J)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])
```

```
## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`Calibration_Michel` *Calibration algorithm that optimises the error criterion selected as objective function using the Irstea-HBAN procedure described by C. Michel*

Description

Calibration algorithm that optimises the error criterion selected as objective function.

The algorithm combines a global and a local approach. First, a screening is performed using either a rough predefined grid or a list of parameter sets. Then a steepest descent local search algorithm is performed, starting from the result of the screening procedure.

Usage

```
Calibration_Michel(InputsModel, RunOptions, InputsCrit, CalibOptions,
  FUN_MOD, FUN_CRIT, FUN_TRANSFO = NULL, verbose = TRUE)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`InputsCrit`	[object of class *InputsCrit*] see `CreateInputsCrit` for details
`CalibOptions`	[object of class *CalibOptions*] see `CreateCalibOptions` for details
`FUN_MOD`	[function] hydrological model function (e.g. `RunModel_GR4J`, `RunModel_CemaNeigeGR4J`)
`FUN_CRIT`	[function] error criterion function (e.g. `ErrorCrit_RMSE`, `ErrorCrit_NSE`)
`FUN_TRANSFO`	(optional) [function] model parameters transformation function, if the `FUN_MOD` used is native in the package `FUN_TRANSFO` is automatically defined
`verbose`	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = `TRUE`

Details

A screening is first performed either based on a rough predefined grid (considering various initial values for each parameter) or from a list of initial parameter sets.
The best set identified in this screening is then used as a starting point for the steepest descent local search algorithm.
For this search, since the ranges of parameter values can be quite different, simple mathematical transformations are applied to parameters to make them vary in a similar range and get a similar sensitivity to a predefined search step. This is done using the TransfoParam functions.
During the steepest descent method, at each iteration, we start from a parameter set of NParam values (NParam being the number of free parameters of the chosen hydrological model) and we determine the 2*NParam-1 new candidates by changing one by one the different parameters (+/- search step).
All these candidates are tested and the best one kept to be the starting point for the next iteration. At the end of each iteration, the the search step is either increased or decreased to adapt the progression speed. A composite step can occasionally be done.
The calibration algorithm stops when the search step becomes smaller than a predefined threshold.

Value

list list containing the function outputs organised as follows:

$ParamFinalR	[numeric] parameter set obtained at the end of the calibration
$CritFinal	[numeric] error criterion selected as objective function obtained at the end of the calibration
$NIter	[numeric] number of iterations during the calibration
$NRuns	[numeric] number of model runs done during the calibration
$HistParamR	[numeric] table showing the progression steps in the search for optimal set: parameter values
$HistCrit	[numeric] table showing the progression steps in the search for optimal set: criterion values
$MatBoolCrit	[boolean] table giving the requested and actual time steps over which the model is calibrated
$CritName	[character] name of the calibration criterion used as objective function
$CritBestValue	[numeric] theoretical best criterion value

Author(s)

Laurent Coron, Claude Michel, Olivier Delaigue, Guillaume Thirel

References

Michel, C. (1991), Hydrologie appliquée aux petits bassins ruraux, Hydrology handbook (in French), Cemagref, Antony, France.

See Also

`Calibration`, `RunModel_GR4J`, `TransfoParam_GR4J`, `ErrorCrit_RMSE`, `CreateInputsModel`, `CreateRunOptions`, `CreateInputsCrit`, `CreateCalibOptions`.

Examples

```
library(airGR)
```

```
## loading catchment data
data(L0123001)

## preparation of InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## calibration period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J, InputsModel = InputsModel,
                               IndPeriod_Run = Ind_Run)

## calibration criterion: preparation of the InputsCrit object
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])

## preparation of CalibOptions object
CalibOptions <- CreateCalibOptions(FUN_MOD = RunModel_GR4J, FUN_CALIB = Calibration_Michel)

## calibration
OutputsCalib <- Calibration_Michel(InputsModel = InputsModel, RunOptions = RunOptions,
                                   InputsCrit = InputsCrit, CalibOptions = CalibOptions,
                                   FUN_MOD = RunModel_GR4J, FUN_CRIT = ErrorCrit_NSE)

## simulation
Param <- OutputsCalib$ParamFinalR
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel,
                              RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`CreateCalibOptions` *Creation of the CalibOptions object required but the Calibration functions*

Description

Creation of the *CalibOptions* object required by the `Calibration*` functions.

Usage

```
CreateCalibOptions(FUN_MOD, FUN_CALIB = Calibration_Michel,
  FUN_TRANSFO = NULL, FixedParam = NULL,
  SearchRanges = NULL, StartParamList = NULL,
  StartParamDistrib = NULL)
```

Arguments

`FUN_MOD` [function] hydrological model function (e.g. `RunModel_GR4J`, `RunModel_CemaNeigeGR4J`)

`FUN_CALIB` (optional) [function] calibration algorithm function (e.g. Calibration_Michel), default = `Calibration_Michel`

`FUN_TRANSFO` (optional) [function] model parameters transformation function, if the `FUN_MOD` used is native in the package, `FUN_TRANSFO` is automatically defined

`FixedParam` (optional) [numeric] vector giving the values set for the non-optimised parameter values (NParam columns, 1 line)
Example:

NA	NA	3.34	...	NA

`SearchRanges` (optional) [numeric] matrix giving the ranges of real parameters (NParam columns, 2 lines)
Example:

	[X1]	[X2]	[X3]	[...]	[Xi]
[1,]	0	-1	0	...	0.0
[2,]	3000	+1	100	...	3.0

`StartParamList` (optional) [numeric] matrix of parameter sets used for grid-screening calibration procedure (values in columns, sets in line)
Example:

	[X1]	[X2]	[X3]	[...]	[Xi]
[set1]	800	-0.7	25	...	1.0
[set2]	1000	-0.5	22	...	1.1
[...]	...	...	...	...	...
[set n]	200	-0.3	17	...	1.0

`StartParamDistrib` (optional) [numeric] matrix of parameter values used for grid-screening calibration procedure (values in columns, percentiles in line)

Example:

	[X1]	[X2]	[X3]	[...]	[Xi]
[value1]	800	-0.7	25	...	1.0
[value2]	1000	NA	50	...	1.2
[value3]	1200	NA	NA	...	1.6

Details

Users wanting to use `FUN_MOD`, `FUN_CALIB` or `FUN_TRANSFO` functions that are not included in the package must create their own `CalibOptions` object accordingly.

Value

list object of class *CalibOptions* containing the data required to evaluate the model outputs; it can include the following:

$FixedParam	[numeric] vector giving the values to allocate to non-optimised parameter values
$SearchRanges	[numeric] matrix giving the ranges of raw parameters
$StartParamList	[numeric] matrix of parameter sets used for grid-screening calibration procedure
$StartParamDistrib	[numeric] matrix of parameter values used for grid-screening calibration procedure

Author(s)

Laurent Coron

See Also

`Calibration`, `RunModel`

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## calibration period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)
```

```
## calibration criterion: preparation of the InputsCrit object
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])

## preparation of CalibOptions object
CalibOptions <- CreateCalibOptions(FUN_MOD = RunModel_GR4J, FUN_CALIB = Calibration_Michel)

## calibration
OutputsCalib <- Calibration(InputsModel = InputsModel, RunOptions = RunOptions,
                            InputsCrit = InputsCrit, CalibOptions = CalibOptions,
                            FUN_MOD = RunModel_GR4J, FUN_CRIT = ErrorCrit_NSE,
                            FUN_CALIB = Calibration_Michel)

## simulation
Param <- OutputsCalib$ParamFinalR
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions,
                         Param = Param, FUN = RunModel_GR4J)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`CreateIniStates` *Creation of the IniStates object possibly required by the CreateRunOptions functions*

Description

Creation of the *IniStates* object possibly required by the `CreateRunOptions` function.

Usage

```
CreateIniStates(FUN_MOD, InputsModel,
  ProdStore = 350, RoutStore = 90, ExpStore = NULL,
  UH1 = NULL, UH2 = NULL,
  GCemaNeigeLayers = NULL, eTGCemaNeigeLayers = NULL,
  verbose = TRUE)
```

Arguments

FUN_MOD	[function] hydrological model function (e.g. RunModel_GR4J, RunModel_CemaNeigeGR4J)
InputsModel	[object of class InputsModel] see CreateInputsModel for details
ProdStore	[numeric] production store level [mm]
RoutStore	[numeric] routing store level [mm]
ExpStore	(optional) [numeric] series of exponential store level (negative) [mm] for the GR6J model
UH1	(optional) [numeric] unit hydrograph 1 levels [mm]
UH2	(optional) [numeric] unit hydrograph 2 levels [mm]
GCemaNeigeLayers	(optional) [numeric] snow pack [mm], possibly used to create the CemaNeige model initial state
eTGCemaNeigeLayers	(optional) [numeric] snow pack thermal state [°C], possibly used to create the CemaNeige model initial state
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Details

20 numeric values are required for UH1 and 40 numeric values are required for UH2 if GR4J, GR5J or GR6J are used (respectively 20*24 and 40*24 for the hourly model GR4H).Please note that depending on the X4 parameter value that will be provided when running the model, not all the values may be used (only the first int(X4)+1 values are used for UH1 and the first 2*int(X4)+1 for UH2).
GCemaNeigeLayers and eTGCemaNeigeLayers require each numeric values as many as given in CreateInputsModel with the NLayersargument. eTGCemaNeigeLayers values can be negatives.
The structure of the object of class IniStates returned is always exactly the same for all models (except for the unit hydrographs levels that contain more values with GR4H), even if some states do nt exist (e.g. *UHUH1* for GR2M).
If CemaNeige is not used, *$CemaNeigeLayers$G* and *$CemaNeigeLayers$eTG* are set to NA.

Nota: the StateEnd objects from the outputs of RunModel* functions already respect the format given by the CreateIniStates function.

Value

list object of class IniStates containing the initial model internal states; it always includes the following:

$Store	[numeric] list of store levels (*$Prod*, *$Rout* and *$Exp*)
$UH	[numeric] list of unit hydrographs levels (*$UH1* and *$UH2*)
$CemaNeigeLayers	[numeric] list of CemaNeige variables (*$G* and *$eTG*)

Author(s)

Olivier Delaigue

See Also

CreateRunOptions

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y %H:%M")=="01/01/1990 00:00"),
            which(format(BasinObs$DatesR, format = "%d/%m/%Y %H:%M")=="31/12/1991 00:00"))


### preparation of the IniStates object with low values of ProdStore and RoutStore
IniStates <- CreateIniStates(FUN_MOD = RunModel_GR4J, InputsModel = InputsModel,
                             ProdStore = 0, RoutStore = 0, ExpStore = NULL,
                             UH1 = c(0.52, 0.54, 0.15, rep(0, 17)),
                             UH2 = c(0.057, 0.042, 0.015, 0.005, rep(0, 36)),
                             GCemaNeigeLayers = NULL, eTGCemaNeigeLayers = NULL)
str(IniStates)

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J, InputsModel = InputsModel,
                               IndPeriod_WarmUp = 0L,
                               IndPeriod_Run = Ind_Run, IniStates = IniStates)

## simulation
Param <- c(257.238, 1.012, 88.235, 2.208)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])


### preparation of the IniStates object with high values of ProdStore and RoutStore
IniStates <- CreateIniStates(FUN_MOD = RunModel_GR4J, InputsModel = InputsModel,
                             ProdStore = 450, RoutStore = 100, ExpStore = NULL,
                             UH1 = c(0.52, 0.54, 0.15, rep(0, 17)),
                             UH2 = c(0.057, 0.042, 0.015, 0.005, rep(0, 36)),
                             GCemaNeigeLayers = NULL, eTGCemaNeigeLayers = NULL)
str(IniStates)

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J, InputsModel = InputsModel,
                               IndPeriod_WarmUp = 0L,
                               IndPeriod_Run = Ind_Run, IniStates = IniStates)
```

```
## simulation
Param <- c(257.238, 1.012, 88.235, 2.208)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])
```

`CreateInputsCrit` *Creation of the InputsCrit object required to the ErrorCrit functions*

Description

Creation of the *InputsCrit* object required to the `ErrorCrit*` functions.

Usage

```
CreateInputsCrit(FUN_CRIT, InputsModel, RunOptions, Qobs, BoolCrit = NULL,
  transfo = "", Ind_zeroes = NULL, epsilon = NULL, verbose = TRUE)
```

Arguments

`FUN_CRIT`	[function] error criterion function (e.g. ErrorCrit_RMSE, ErrorCrit_NSE)
`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Qobs`	[numeric] series of observed discharges [mm/time step]
`BoolCrit`	(optional) [boolean] boolean giving the time steps to consider in the computation (all time steps are consider by default)
`transfo`	(optional) [character] name of the transformation (e.g. `""`, `"sqrt"`, `"log"`, `"inv"`, `"sort"`)
`Ind_zeroes`	(optional) [numeric] indices of the time steps where zeroes are observed
`epsilon`	(optional) [numeric] epsilon to add to all Qobs and Qsim if *$Ind_zeroes* is not empty
`verbose`	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = `TRUE`

Details

Users wanting to use `FUN_CRIT` functions that are not included in the package must create their own InputsCrit object accordingly.
We do not advise computing KGE or KGE' with log-transformation as it might be wrongly influenced by discharge values close to 0 or 1 and it is dependent on the discharge unit. See Santos et al. (in review) for more details and alternative solutions (see the reference below).

Value

list object of class *InputsCrit* containing the data required to evaluate the model outputs; it can include the following:

$BoolCrit	[boolean] boolean giving the time steps considered in the computation
$Qobs	[numeric] series of observed discharges [mm/time step]
$transfo	[character] name of the transformation (e.g. "", "sqrt", "log", "inv", "sort")
$Ind_zeroes	[numeric] indices of the time steps where zeroes are observed
$epsilon	[numeric] epsilon to add to all Qobs and Qsim if *$Ind_zeroes* is not empty

Author(s)

Laurent Coron, Olivier Delaigue, Guillaume Thirel

References

Santos, L., Thirel, G. and Perrin, C. (in review), Technical note : Pitfalls in using log-transformed flows within the KGE criterion, Hydrology and Earth System Sciences Discussions, doi:10.5194/hess-2018-298.

See Also

RunModel, CreateInputsModel, CreateRunOptions, CreateCalibOptions

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(734.568, -0.840, 109.809, 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions,
                         Param = Param, FUN = RunModel_GR4J)

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Nash-Sutcliffe Efficiency on log-transformed flows
transfo <- "log"
```

```
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run],
                               transfo = transfo)
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Nash-Sutcliffe Efficiency above a threshold (q75%)
BoolCrit <- rep(TRUE, length(BasinObs$Qmm[Ind_Run]));
BoolCrit[BasinObs$Qmm[Ind_Run]<quantile(BasinObs$Qmm[Ind_Run], 0.75, na.rm = TRUE)] <- FALSE
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run],
                               BoolCrit = BoolCrit)
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency below a threshold (q10%) on log-transformed flows
transfo  <- "log"
BoolCrit <- rep(TRUE, length(BasinObs$Qmm[Ind_Run]));
BoolCrit[BasinObs$Qmm[Ind_Run]>quantile(BasinObs$Qmm[Ind_Run], 0.10, na.rm = TRUE)] <- FALSE
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE,
                               InputsModel = InputsModel,RunOptions = RunOptions,
                            Qobs = BasinObs$Qmm[Ind_Run], BoolCrit = BoolCrit, transfo = transfo)
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`CreateInputsModel` *Creation of the InputsModel object required to the RunModel functions*

Description

Creation of the *InputsModel* object required to the `RunModel*` functions.

Usage

```
CreateInputsModel(FUN_MOD, DatesR, Precip, PrecipScale = TRUE, PotEvap = NULL,
  TempMean = NULL, TempMin = NULL, TempMax = NULL, ZInputs = NULL, HypsoData = NULL,
  NLayers = 5, verbose = TRUE)
```

Arguments

`FUN_MOD`	[function] hydrological model function (e.g. `RunModel_GR4J`, `RunModel_CemaNeigeGR4J`)
`DatesR`	[POSIXt] vector of dates required to create the GR model and CemaNeige module inputs
`Precip`	[numeric] time series of total precipitation (catchment average) [mm/time step], required to create the GR model and CemaNeige module inputs

PrecipScale	(optional) [boolean] indicating if the mean of the precipitation interpolated on the elevation layers must be kept or not, required to create CemaNeige module inputs, default = TRUE (the mean of the precipitation is kept to the original value)
PotEvap	[numeric] time series of potential evapotranspiration (catchment average) [mm/time step], required to create the GR model inputs
TempMean	(optional) [numeric] time series of mean air temperature [°C], required to create the CemaNeige module inputs
TempMin	(optional) [numeric] time series of min air temperature [°C], possibly used to create the CemaNeige module inputs
TempMax	(optional) [numeric] time series of max air temperature [°C], possibly used to create the CemaNeige module inputs
ZInputs	(optional) [numeric] real giving the mean elevation of the Precip and Temp series (before extrapolation) [m], possibly used to create the CemaNeige module inputs
HypsoData	(optional) [numeric] vector of 101 reals: min, q01 to q99 and max of catchment elevation distribution [m], if not defined a single elevation is used for CemaNeige
NLayers	(optional) [numeric] integer giving the number of elevation layers requested [-], required to create CemaNeige module inputs, default=5
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Details

Users wanting to use FUN_MOD functions that are not included in the package must create their own InputsModel object accordingly.
Please note that if CemaNeige is used, and ZInputs is different than HypsoData, then precipitation and temperature are interpolated with the DataAltiExtrapolation_Valery function.

Value

list object of class *InputsModel* containing the data required to evaluate the model outputs; it can include the following:

$DatesR	[POSIXlt] vector of dates
$Precip	[numeric] time series of total precipitation (catchment average) [mm/time step]
$PotEvap	[numeric] time series of potential evapotranspiration (catchment average) [mm/time step], defined if FUN_MOD includes GR4H, GR4J, GR5J, GR6J, GR2M or GR1A
$LayerPrecip	[list] list of time series of precipitation (layer average) [mm/time step], defined if FUN_MOD includes CemaNeige
$LayerTempMean	[list] list of time series of mean air temperature (layer average) [°C], defined if FUN_MOD includes CemaNeige
$LayerFracSolidPrecip	[list] list of time series of solid precipitation fraction (layer average) [-], defined if FUN_MOD includes CemaNeige

Author(s)

Laurent Coron

See Also

RunModel, CreateRunOptions, CreateInputsCrit, CreateCalibOptions, DataAltiExtrapolation_Valery

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(734.568, -0.840, 109.809, 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param,
                         FUN_MOD = RunModel_GR4J)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`CreateRunOptions` *Creation of the RunOptions object required to the RunModel functions*

Description

Creation of the RunOptions object required to the `RunModel*` functions.

Usage

```
CreateRunOptions(FUN_MOD, InputsModel, IndPeriod_WarmUp = NULL, IndPeriod_Run,
  IniStates = NULL, IniResLevels = NULL, Outputs_Cal = NULL,
  Outputs_Sim = "all", RunSnowModule, MeanAnSolidPrecip = NULL,
  verbose = TRUE)
```

Arguments

FUN_MOD	[function] hydrological model function (e.g. RunModel_GR4J, RunModel_CemaNeigeGR4J)
InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
IndPeriod_WarmUp	(optional) [numeric] index of period to be used for the model warm-up [-]
IndPeriod_Run	[numeric] index of period to be used for the model run [-]
IniStates	(optional) [numeric] object of class IniStates [mm and °C], see CreateIniStates for details
IniResLevels	(optional) [numeric] vector of initial fillings for the GR stores (2 or 3 values according to the model) [- and/or mm]; see details
Outputs_Cal	(optional) [character] vector giving the outputs needed for the calibration (e.g. c("Qsim")), the fewer outputs the faster the calibration
Outputs_Sim	(optional) [character] vector giving the requested outputs (e.g. c("DatesR", "Qsim", "SnowPack")), default = "all"
RunSnowModule	(deprecated) [boolean] option indicating whether CemaNeige should be activated. Please adapt FUN_MOD instead
MeanAnSolidPrecip	(optional) [numeric] vector giving the annual mean of average solid precipitation for each layer (computed from InputsModel if not defined) [mm/y]
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Details

Users wanting to use FUN_MOD functions that are not included in the package must create their own RunOptions object accordingly.

Initialisation options

The model initialisation options can either be set to a default configuration or be defined by the user.

This is done via three vectors:
IndPeriod_WarmUp, IniStates, IniResLevels.
A default configuration is used for initialisation if these vectors are not defined.

(1) Default initialisation options:

- IndPeriod_WarmUp default setting ensures a one-year warm-up using the time steps preceding the IndPeriod_Run. The actual length of this warm-up might be shorter depending on data availability (no missing value of climate inputs being allowed in model input series).

- `IniStates` and `IniResLevels` are automatically set to initialise all the model states at 0, except for the production and routing stores which are respectively initialised at 30 % and 50 % of their capacity. In case GR6J is used, the exponential store is initialised by default with 0 mm. This initialisation is made at the very beginning of the model call (i.e. at the beginning of `IndPeriod_WarmUp` or at the beginning of `IndPeriod_Run` if the warm-up period is disabled).

(2) Customisation of initialisation options:

- `IndPeriod_WarmUp` can be used to specify the indices of the warm-up period (within the time series prepared in InputsModel).
  - remark 1: for most common cases, indices corresponding to one or several years preceding `IndPeriod_Run` are used (e.g. `IndPeriod_WarmUp = 1000:1365` and `IndPeriod_Run = 1366:5000`). However, it is also possible to perform a long-term initialisation if other indices than the warm-up ones are set in `IndPeriod_WarmUp` (e.g. `IndPeriod_WarmUp = c(1:5000, 1:5000, 1:5000, 1000:136`
  - remark 2: it is also possible to completely disable the warm-up period when using `IndPeriod_WarmUp = 0L`. This is necessary if you want `IniStates` and/or `IniResLevels` to be the actual initial values of the model variables from your simulation (e.g. to perform a forecast form a given initial state).
- `IniStates` and `IniResLevels` can be used to specify the initial model states.
  - remark 1: `IniStates` and `IniResLevels` can not be used with GR1A.
  - remark 2: if `IniStates` is used, two possibilities are offered:
    - `IniStates` can be set to the *$StateEnd* output of a previous `RunModel` call, as *$StateEnd* already respects the correct format;
    - `IniStates` can be created with the `CreateIniStates` function.
  - remark 3: in addition to `IniStates`, `IniResLevels` allows to set the filling rate of the production and routing stores for the GR models. For instance for GR4J and GR5J: `IniResLevels = c(0.3, 0.5)` should be used to obtain initial fillings of 30 % and 50 % for the production and routing stores, respectively. For GR6J, `IniResLevels = c(0.3, 0.5, 0)` should be use to obtain initial fillings of 30 % and 50 % for the production, routing stores and 0 mm for the exponential store, respectively. `IniResLevels` is optional and can only be used if `IniStates` is also defined (the state values corresponding to these two other stores in `IniStates` are not used in such case).

Value

list object of class *RunOptions* containing the data required to evaluate the model outputs; it can include the following:

IndPeriod_WarmUp	[numeric] index of period to be used for the model warm-up [-]
IndPeriod_Run	[numeric] index of period to be used for the model run [-]
IniStates	[numeric] vector of initial model states [mm and °C]
IniResLevels	[numeric] vector of initial filling rates for production and routing stores [-] and level for the exponentia
Outputs_Cal	[character] character vector giving only the outputs needed for the calibration
Outputs_Sim	[character] character vector giving the requested outputs
MeanAnSolidPrecip	[numeric] vector giving the annual mean of average solid precipitation for each layer [mm/y]

Author(s)

Laurent Coron, Olivier Delaigue, Guillaume Thirel

See Also

RunModel, CreateInputsModel, CreateInputsCrit, CreateCalibOptions, CreateIniStates

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(734.568, -0.840, 109.809, 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param,
                         FUN_MOD = RunModel_GR4J)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

DataAltiExtrapolation_Valery

Altitudinal extrapolation of precipitation and temperature series described by A. Valery

---

Description

Function which extrapolates the precipitation and air temperature series for different elevation layers (method from Valéry, 2010).

Usage

```
DataAltiExtrapolation_Valery(DatesR, Precip, PrecipScale = TRUE,
  TempMean, TempMin = NULL, TempMax = NULL,
  ZInputs, HypsoData, NLayers, verbose = TRUE)
```

Arguments

DatesR	[POSIXt] vector of dates
Precip	[numeric] time series of daily total precipitation (catchment average) [mm/d]
PrecipScale	(optional) [boolean] indicating if the mean of the precipitation interpolated on the elevation layers must be kept or not, required to create CemaNeige module inputs, default = TRUE (the mean of the precipitation is kept to the original value)
TempMean	[numeric] time series of daily mean air temperature [°C]
TempMin	(optional) [numeric] time series of daily min air temperature [°C]
TempMax	(optional) [numeric] time series of daily max air temperature [°C]
ZInputs	[numeric] real giving the mean elevation of the Precip and Temp series (before extrapolation) [m]
HypsoData	[numeric] vector of 101 reals: min, q01 to q99 and max of catchment elevation distribution [m]
NLayers	[numeric] integer giving the number of elevation layers requested [-]
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Details

Elevation layers of equal surface are created the 101 elevation quantiles (HypsoData) and the number requested elevation layers (NLayers).
Forcing data (precipitation and air temperature) are extrapolated using gradients from Valery (2010). (e.g. gradP = 0.0004 [m-1] for France and gradT = 0.434 [°C/100m] for January, 1st).
This function is used by the CreateInputsModel function.

Value

list containing the extrapolated series of precip. and air temp. on each elevation layer

$LayerPrecip	[list] list of time series of daily precipitation (layer average) [mm/d]
$LayerTempMean	[list] list of time series of daily mean air temperature (layer average) [°C]
$LayerTempMin	[list] list of time series of daily min air temperature (layer average) [°C]
$LayerTempMax	[list] list of time series of daily max air temperature (layer average) [°C]
$LayerFracSolidPrecip	[list] list of time series of daily solid precip. fract. (layer average) [-]
$ZLayers	[numeric] vector of median elevation for each layer

Author(s)

Laurent Coron, Audrey Valéry, Pierre Brigode, Olivier Delaigue, Guillaume Thirel

References

Turcotte, R., L.-G. Fortin, V. Fortin, J.-P. Fortin and J.-P. Villeneuve (2007), Operational analysis of the spatial distribution and the temporal evolution of the snowpack water equivalent in southern Quebec, Canada, Nordic Hydrology, 38(3), 211, doi:10.2166/nh.2007.009.
Valéry, A. (2010), Modélisation précipitations-débit sous influence nivale ? : Elaboration d'un module neige et évaluation sur 380 bassins versants, PhD thesis (in french), AgroParisTech, Paris, France.
USACE (1956), Snow Hydrology, pp. 437, U.S. Army Coprs of Engineers (USACE) North Pacific Division, Portland, Oregon, USA.

See Also

CreateInputsModel, RunModel_CemaNeigeGR4J

ErrorCrit *Error criterion using the provided function*

Description

Function which computes an error criterion with the provided function.

Usage

```
ErrorCrit(InputsCrit, OutputsModel, FUN_CRIT, warnings = TRUE, verbose = TRUE)
```

Arguments

InputsCrit	[object of class *InputsCrit*] see CreateInputsCrit for details
OutputsModel	[object of class *OutputsModel*] see RunModel_GR4J or RunModel_CemaNeigeGR4J for details
FUN_CRIT	[function] error criterion function (e.g. ErrorCrit_RMSE, ErrorCrit_NSE)
warnings	(optional) [boolean] boolean indicating if the warning messages are shown, default = TRUE
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Value

list list containing the function outputs, see ErrorCrit_RMSE or ErrorCrit_NSE for details

Author(s)

Laurent Coron

See Also

ErrorCrit_RMSE, ErrorCrit_NSE, ErrorCrit_KGE

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(734.568, -0.840, 109.809, 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions,
                         Param = Param, FUN = RunModel_GR4J)

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Nash-Sutcliffe Efficiency on log-transformed flows
transfo <- "log"
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run],
                               transfo = transfo)
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Nash-Sutcliffe Efficiency above a threshold (q75%)
BoolCrit <- rep(TRUE, length(BasinObs$Qmm[Ind_Run]))
BoolCrit[BasinObs$Qmm[Ind_Run]<quantile(BasinObs$Qmm[Ind_Run], 0.75, na.rm = TRUE)] <- FALSE
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run],
                               BoolCrit = BoolCrit)
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency below a threshold (q10%) on log-transformed flows
transfo  <- "log"
BoolCrit <- rep(TRUE, length(BasinObs$Qmm[Ind_Run]))
BoolCrit[BasinObs$Qmm[Ind_Run]>quantile(BasinObs$Qmm[Ind_Run], 0.10, na.rm = TRUE)] <- FALSE
```

```
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run],
                               BoolCrit = BoolCrit, transfo = transfo)
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`ErrorCrit_KGE` *Error criterion based on the KGE formula*

Description

Function which computes an error criterion based on the KGE formula proposed by Gupta et al. (2009).

Usage

```
ErrorCrit_KGE(InputsCrit, OutputsModel, warnings = TRUE, verbose = TRUE)
```

Arguments

`InputsCrit`	[object of class *InputsCrit*] see `CreateInputsCrit` for details
`OutputsModel`	[object of class *OutputsModel*] see `RunModel_GR4J` or `RunModel_CemaNeigeGR4J` for details
`warnings`	(optional) [boolean] boolean indicating if the warning messages are shown, default = `TRUE`
`verbose`	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = `TRUE`

Details

In addition to the criterion value, the function outputs include a multiplier (-1 or +1) which allows the use of the function for model calibration: the product CritValue * Multiplier is the criterion to be minimised (Multiplier = -1 for KGE).

The KGE formula is

$$KGE = 1 - \sqrt{(r-1)^2 + (\alpha-1)^2 + (\beta-1)^2}$$

with the following sub-criteria:

$$r = \text{the linear correlation coefficient between } sim \text{ and } obs$$

$$\alpha = \frac{\sigma_{sim}}{\sigma_{obs}}$$

$$\beta = \frac{\mu_{sim}}{\mu_{obs}}$$

Value

list list containing the function outputs organised as follows:

$CritValue	[numeric] value of the criterion
$CritName	[character] name of the criterion
$SubCritValues	[numeric] values of the sub-criteria
$SubCritNames	[character] names of the components of the criterion
$CritBestValue	[numeric] theoretical best criterion value
$Multiplier	[numeric] integer indicating whether the criterion is indeed an error (+1) or an efficiency (-1)
$Ind_notcomputed	[numeric] indices of the time steps where *InputsCrit$BoolCrit* = FALSE or no data is available

Author(s)

Laurent Coron

References

Gupta, H. V., Kling, H., Yilmaz, K. K. and Martinez, G. F. (2009), Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, Journal of Hydrology, 377(1-2), 80-91, doi:10.1016/j.jhydrol.2009.08.003.

See Also

ErrorCrit, ErrorCrit_RMSE, ErrorCrit_NSE, ErrorCrit_KGE2

Examples

```
## see example of the ErrorCrit function
```

ErrorCrit_KGE2 *Error criterion based on the KGE' formula*

Description

Function which computes an error criterion based on the KGE' formula proposed by Kling et al. (2012).

Usage

```
ErrorCrit_KGE2(InputsCrit, OutputsModel, warnings = TRUE, verbose = TRUE)
```

Arguments

InputsCrit	[object of class *InputsCrit*] see CreateInputsCrit for details
OutputsModel	[object of class *OutputsModel*] see RunModel_GR4J or RunModel_CemaNeigeGR4J for details
warnings	(optional) [boolean] boolean indicating if the warning messages are shown, default = TRUE
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Details

In addition to the criterion value, the function outputs include a multiplier (-1 or +1) which allows the use of the function for model calibration: the product CritValue * Multiplier is the criterion to be minimised (Multiplier = -1 for KGE2).

The KGE' formula is

$$KGE' = 1 - \sqrt{(r-1)^2 + (\gamma-1)^2 + (\beta-1)^2}$$

with the following sub-criteria:

$$r = \text{the linear correlation coefficient between } sim \text{ and } obs$$

$$\gamma = \frac{CV_{sim}}{CV_{obs}}$$

$$\beta = \frac{\mu_{sim}}{\mu_{obs}}$$

Value

list list containing the function outputs organised as follows:

$CritValue	[numeric] value of the criterion
$CritName	[character] name of the criterion
$SubCritValues	[numeric] values of the sub-criteria
$SubCritNames	[character] names of the components of the criterion
$CritBestValue	[numeric] theoretical best criterion value
$Multiplier	[numeric] integer indicating whether the criterion is indeed an error (+1) or an efficiency (-1)
$Ind_notcomputed	[numeric] indices of the time steps where *InputsCrit$BoolCrit* = FALSE or no data is available

Author(s)

Laurent Coron

References

Gupta, H. V., Kling, H., Yilmaz, K. K. and Martinez, G. F. (2009), Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, Journal of Hydrology, 377(1-2), 80-91, doi:10.1016/j.jhydrol.2009.08.003.
Kling, H., Fuchs, M. and Paulin, M. (2012), Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios, Journal of Hydrology, 424-425, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

See Also

ErrorCrit, ErrorCrit_RMSE, ErrorCrit_NSE, ErrorCrit_KGE

Examples

```
## see example of the ErrorCrit function
```

`ErrorCrit_NSE` *Error criterion based on the NSE formula*

Description

Function which computes an error criterion based on the NSE formula proposed by Nash & Sutcliffe (1970).

Usage

```
ErrorCrit_NSE(InputsCrit, OutputsModel, warnings = TRUE, verbose = TRUE)
```

Arguments

`InputsCrit`	[object of class *InputsCrit*] see `CreateInputsCrit` for details
`OutputsModel`	[object of class *OutputsModel*] see `RunModel_GR4J` or `RunModel_CemaNeigeGR4J` for details
`warnings`	(optional) [boolean] boolean indicating if the warning messages are shown, default = `TRUE`
`verbose`	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = `TRUE`

Details

In addition to the criterion value, the function outputs include a multiplier (-1 or +1) which allows the use of the function for model calibration: the product CritValue * Multiplier is the criterion to be minimised (Multiplier = -1 for NSE).

Value

list list containing the function outputs organised as follows:

$CritValue	[numeric] value of the criterion
$CritName	[character] name of the criterion
$CritBestValue	[numeric] theoretical best criterion value
$Multiplier	[numeric] integer indicating whether the criterion is indeed an error (+1) or an efficiency (-1)
$Ind_notcomputed	[numeric] indices of the time steps where *InputsCrit$BoolCrit* = `FALSE` or no data is available

Author(s)

Laurent Coron

References

Nash, J.E. and Sutcliffe, J.V. (1970), River flow forecasting through conceptual models part 1. A discussion of principles, Journal of Hydrology, 10(3), 282-290, doi:10.1016/0022-1694(70)90255-

6.

See Also

ErrorCrit_RMSE, ErrorCrit_KGE, ErrorCrit_KGE2

Examples

```
## see example of the ErrorCrit function
```

ErrorCrit_RMSE *Error criterion based on the RMSE*

Description

Function which computes an error criterion based on the root mean square error (RMSE).

Usage

```
ErrorCrit_RMSE(InputsCrit, OutputsModel, warnings = TRUE, verbose = TRUE)
```

Arguments

InputsCrit	[object of class *InputsCrit*] see CreateInputsCrit for details
OutputsModel	[object of class *OutputsModel*] see RunModel_GR4J or RunModel_CemaNeigeGR4J for details
warnings	(optional) [boolean] boolean indicating if the warning messages are shown, default = TRUE
verbose	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = TRUE

Details

In addition to the criterion value, the function outputs include a multiplier (-1 or +1) which allows the use of the function for model calibration: the product CritValue * Multiplier is the criterion to be minimised (Multiplier = +1 for RMSE).

Value

list list containing the function outputs organised as follows:

$CritValue	[numeric] value of the criterion
$CritName	[character] name of the criterion
$CritBestValue	[numeric] theoretical best criterion value
$Multiplier	[numeric] integer indicating whether the criterion is indeed an error (+1) or an efficiency (-1)
$Ind_notcomputed	[numeric] indices of the time steps where *InputsCrit$BoolCrit* = FALSE or no data is available

Author(s)

Laurent Coron, Ludovic Oudin

See Also

ErrorCrit_NSE, ErrorCrit_KGE, ErrorCrit_KGE2

Examples

```
## see example of the ErrorCrit function
```

Param_Sets_GR4J *Generalist parameter sets for the GR4J model*

Description

These parameter sets can be used as an alternative for the grid-screening calibration procedure (i.e. first step in Calibration_Michel). Please note that the given GR4J X4u variable does not correspond to the actual GR4J X4 parameter. As explained in Andréassian et al. (2014; section 2.1), the given GR4J X4u value has to be adjusted (rescaled) using catchment area (S) [km2] as follows: X4 = X4u / 5.995 * S^0.3 (please note that the formula is erroneous in the publication). Please, see the example below.
As shown in Andréassian et al. (2014; figure 4), only using these parameters sets as the tested values for calibration is more efficient than a classical calibration when the amount of data is low (6 months or less).

Format

Data frame of parameters containing four numeric vectors

- GR4J X1 production store capacity [mm]
- GR4J X2 intercatchment exchange coefficient [mm/d]
- GR4J X3 routing store capacity [mm]
- GR4J X4u unajusted unit hydrograph time constant [d]

References

Andréassian, V., F. Bourgin, L. Oudin, T. Mathevet, C. Perrin, J. Lerat, L. Coron, L. Berthet (2014). Seeking genericity in the selection of parameter sets: impact on hydrological model efficiency. Water Resources Research, 50(10), 8356-8366, doi: 10.1002/2013WR014761.

See Also

RunModel_GR4J, Calibration_Michel, CreateCalibOptions.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## loading generalist parameter sets
data(Param_Sets_GR4J)
str(Param_Sets_GR4J)

## computation of the real GR4J X4
Param_Sets_GR4J$X4 <- Param_Sets_GR4J$X4u / 5.995 * BasinInfo$BasinArea^0.3
Param_Sets_GR4J$X4u <- NULL
Param_Sets_GR4J <- as.matrix(Param_Sets_GR4J)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## ---- calibration step

## short calibration period selection (< 6 months)
Ind_Cal <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="28/02/1990"))

## preparation of the RunOptions object for the calibration period
RunOptions_Cal <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                                   InputsModel = InputsModel, IndPeriod_Run = Ind_Cal)

## simulation and efficiency criterion (Nash-Sutcliffe Efficiency)
## with all generalist parameter sets on the calibration period
OutputsCrit_Loop <- apply(Param_Sets_GR4J, 1, function(Param) {
  OutputsModel_Cal <- RunModel_GR4J(InputsModel = InputsModel,
                                    RunOptions = RunOptions_Cal,
                                    Param = Param)
  InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                  RunOptions = RunOptions_Cal, Qobs = BasinObs$Qmm[Ind_Cal])
  OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel_Cal)
  return(OutputsCrit$CritValue)
})

## best parameter set
Param_Best <- unlist(Param_Sets_GR4J[which.max(OutputsCrit_Loop), ])


## ---- validation step

## validation period selection
Ind_Val <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/03/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object for the validation period
```

```
RunOptions_Val <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                                   InputsModel = InputsModel, IndPeriod_Run = Ind_Val)

## simulation with the best parameter set on the validation period
OutputsModel_Val <- RunModel_GR4J(InputsModel = InputsModel,
                                  RunOptions = RunOptions_Val,
                                  Param = Param_Best)

## results preview of the simulation with the best parameter set on the validation period
plot(OutputsModel_Val, Qobs = BasinObs$Qmm[Ind_Val])

## efficiency criterion (Nash-Sutcliffe Efficiency) on the validation period
InputsCrit_Val  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                    RunOptions = RunOptions_Val, Qobs = BasinObs$Qmm[Ind_Val])
OutputsCrit_Val <- ErrorCrit_NSE(InputsCrit = InputsCrit_Val, OutputsModel = OutputsModel_Val)
```

`PEdaily_Oudin` *Computation of daily series of potential evapotranspiration with Oudin's formula*

Description

Function which computes daily PE using the formula from Oudin et al. (2005).

Usage

```
PEdaily_Oudin(JD, Temp, LatRad)
```

Arguments

`JD`	[numeric] time series of julian day [-]
`Temp`	[numeric] time series of daily mean air temperature [°C]
`LatRad`	[numeric] latitude of measurement for the temperature series [rad]

Value

numeric time series of daily potential evapotranspiration [mm/d]

Author(s)

Laurent Coron, Ludovic Oudin

References

Oudin, L., F. Hervieu, C. Michel, C. Perrin, V. Andréassian, F. Anctil and C. Loumagne (2005), Which potential evapotranspiration input for a lumped rainfall-runoff model?: Part 2-Towards a simple and efficient potential evapotranspiration model for rainfall-runoff modelling, Journal of Hydrology, 303(1-4), 290-306, doi:10.1016/j.jhydrol.2004.08.026.

Examples

```
library(airGR)
data(L0123001)
PotEvap <- PEdaily_Oudin(JD = as.POSIXlt(BasinObs$DatesR)$yday, Temp = BasinObs$T, LatRad = 0.8)
```

`plot.OutputsModel` *Default preview of model outputs*

Description

Function which creates a screen plot giving an overview of the model outputs.

Usage

```
## S3 method for class 'OutputsModel'
plot(x, Qobs = NULL, IndPeriod_Plot = NULL,
  BasinArea = NULL, which = "all", log_scale = FALSE,
  cex.axis = 1, cex.lab = 0.9, cex.leg = 0.9, lwd = 1, verbose = TRUE, ...)
```

Arguments

`x`	[object of class *OutputsModel*] list of model outputs (which must at least include DatesR, Precip and Qsim) [POSIXlt, mm, mm]
`Qobs`	(optional) [numeric] time series of observed flow (for the same time steps than simulated) [mm/time step]
`IndPeriod_Plot`	(optional) [numeric] indices of the time steps to be plotted (among the OutputsModel series)
`BasinArea`	(optional) [numeric] basin area [km2], used to plot flow axes in m3/s
`which`	(optional) [character] choice of plots (e.g. c("Precip", "Temp", "SnowPack", "Flows", "Regime", "CumFreq", "CorQQ")), default = "all"
`log_scale`	(optional) [boolean] indicating if the flow axis is to be logarithmic, default = FALSE
`cex.axis`	(optional) [numeric] the magnification to be used for axis annotation relative to the current setting of cex
`cex.lab`	(optional) [numeric] the magnification to be used for x and y labels relative to the current setting of cex
`cex.leg`	(optional) [numeric] the magnification to be used for the legend labels relative to the current setting of cex
`lwd`	(optional) [numeric] the line width (a positive number)
`verbose`	(optional) [boolean] indicating if the function is run in verbose mode or not, default = TRUE
`...`	other parameters to be passed through to plotting functions

Details

Dashboard of results including various graphs (depending on the model):
(1) time series of total precipitation
(2) time series of temperature (plotted only if CemaNeige is used)
(3) time series of snow pack (plotted only if CemaNeige is used)
(4) time series of simulated flows (and observed flows if provided)
(5) interannual median monthly simulated flow (and observed flows if provided)
(6) correlation plot between simulated and observed flows (if observed flows provided)
(7) cumulative frequency plot for simulated flows (and observed flows if provided)

Value

Screen plot window.

Author(s)

Laurent Coron, Olivier Delaigue, Guillaume Thirel

Examples

```
### See examples of RunModel_GR4J or RunModel_CemaNeigeGR4J functions
```

RunModel *Run with the provided hydrological model function*

Description

Function which performs a single model run with the provided function over the selected period.

Usage

```
RunModel(InputsModel, RunOptions, Param, FUN_MOD)
```

Arguments

InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
Param	[numeric] vector of model parameters
FUN_MOD	[function] hydrological model function (e.g. RunModel_GR4J, RunModel_CemaNeigeGR4J)

Value

list see RunModel_GR4J or RunModel_CemaNeigeGR4J for details

Author(s)

Laurent Coron

See Also

RunModel_GR4J, RunModel_CemaNeigeGR4J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(734.568, -0.840, 109.809, 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param,
                         FUN_MOD = RunModel_GR4J)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_CemaNeige *Run with the CemaNeige snow module*

Description

Function which performs a single run for the CemaNeige daily snow module.

Usage

```
RunModel_CemaNeige(InputsModel, RunOptions, Param)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Param`	[numeric] vector of 2 parameters

CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-day melt coefficient [mm/°C/d]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$CemaNeigeLayers	[list] list of CemaNeige outputs (1 list per layer)
$CemaNeigeLayers[[iLayer]]$Pliq	[numeric] series of liquid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$Psol	[numeric] series of solid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$SnowPack	[numeric] series of snow pack [mm]
$CemaNeigeLayers[[iLayer]]$ThermalState	[numeric] series of snow pack thermal state [°C]
$CemaNeigeLayers[[iLayer]]$Gratio	[numeric] series of Gratio [0-1]
$CemaNeigeLayers[[iLayer]]$PotMelt	[numeric] series of potential snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$Melt	[numeric] series of actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$PliqAndMelt	[numeric] series of liquid precip. + actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$Temp	[numeric] series of air temperature [°C]
$StateEnd	[numeric] states at the end of the run: CemaNeige states [mm & °C], see `CreateIniStates` for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron

References

Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": what is useful in a temperature-based snow-accounting routine? Part 1 - Comparison of six snow accounting routines on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.059.
Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": What is useful in a temperature-based snow-accounting routine? Part 2 - Sensitivity analysis of the Cemaneige snow accounting routine on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeigeGR4J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## load of catchment data
data(L0123002)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_CemaNeige, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P,TempMean = BasinObs$T,
                            ZInputs = BasinInfo$HypsoData[51], HypsoData=BasinInfo$HypsoData,
                                 NLayers = 5)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeige, InputsModel = InputsModel,
                               IndPeriod_Run = Ind_Run)

## simulation
Param <- c(0.962, 2.249)
OutputsModel <- RunModel_CemaNeige(InputsModel = InputsModel,
                                   RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel)
```

RunModel_CemaNeigeGR4J

Run with the CemaNeigeGR4J hydrological model

Description

Function which performs a single run for the CemaNeige-GR4J daily lumped model over the test period.

Usage

```
RunModel_CemaNeigeGR4J(InputsModel, RunOptions, Param)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Param`	[numeric] vector of 6 parameters

GR4J X1	production store capacity [mm]
GR4J X2	intercatchment exchange coefficient [mm/d]
GR4J X3	routing store capacity [mm]
GR4J X4	unit hydrograph time constant [d]
CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-day melt coefficient [mm/°C/d]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d]
$Precip	[numeric] series of input total precipitation [mm/d]
$Prod	[numeric] series of production store level [mm]
$Pn	[numeric] series of net rainfall [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store [mm/d]
$AE	[numeric] series of actual evapotranspiration [mm/d]
$Perc	[numeric] series of percolation (PERC) [mm/d]
$PR	[numeric] series of PR=Pn-Ps+Perc [mm/d]
$Q9	[numeric] series of UH1 outflow (Q9) [mm/d]
$Q1	[numeric] series of UH2 outflow (Q1) [mm/d]
$Rout	[numeric] series of routing store level [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/d]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/d]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/d]
$AExch	[numeric] series of actual exchange between catchments (1+2) [mm/d]
$QR	[numeric] series of routing store outflow (QR) [mm/d]
$QD	[numeric] series of direct flow from UH2 after exchange (QD) [mm/d]
$Qsim	[numeric] series of simulated discharge [mm/d]
$CemaNeigeLayers	[list] list of CemaNeige outputs (1 list per layer)
$CemaNeigeLayers[[iLayer]]$Pliq	[numeric] series of liquid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$Psol	[numeric] series of solid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$SnowPack	[numeric] series of snow pack [mm]
$CemaNeigeLayers[[iLayer]]$ThermalState	[numeric] series of snow pack thermal state [°C]
$CemaNeigeLayers[[iLayer]]$Gratio	[numeric] series of Gratio [0-1]
$CemaNeigeLayers[[iLayer]]$PotMelt	[numeric] series of potential snow melt [mm/d]

$CemaNeigeLayers[[iLayer]]$Melt	[numeric] series of actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$PliqAndMelt	[numeric] series of liquid precip. + actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$Temp	[numeric] series of air temperature [°C]
$StateEnd	[numeric] states at the end of the run: store & unit hydrographs levels [mm], CemaNeige states [mm & °C], see CreateIniStates for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Audrey Valéry, Claude Michel, Charles Perrin, Vazken Andréassian

References

Perrin, C., C. Michel and V. Andréassian (2003), Improvement of a parsimonious model for streamflow simulation, Journal of Hydrology, 279(1-4), 275-289, doi:10.1016/S0022-1694(03)00225-7.
Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": what is useful in a temperature-based snow-accounting routine? Part 1 - Comparison of six snow accounting routines on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.059.
Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": What is useful in a temperature-based snow-accounting routine? Part 2 - Sensitivity analysis of the Cemaneige snow accounting routine on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeigeGR5J, RunModel_CemaNeigeGR6J, RunModel_GR4J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## loading catchment data
data(L0123002)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_CemaNeigeGR4J, DatesR = BasinObs$DatesR,
                           Precip = BasinObs$P, PotEvap = BasinObs$E, TempMean = BasinObs$T,
                                  ZInputs = median(BasinInfo$HypsoData),
                                  HypsoData = BasinInfo$HypsoData, NLayers = 5)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR4J, InputsModel = InputsModel,
```

```
                                IndPeriod_Run = Ind_Run)

## simulation
Param <- c(408.774, 2.646, 131.264, 1.174, 0.962, 2.249)
OutputsModel <- RunModel_CemaNeigeGR4J(InputsModel = InputsModel,
                                       RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_CemaNeigeGR5J

Run with the CemaNeigeGR5J hydrological model

Description

Function which performs a single run for the CemaNeige-GR5J daily lumped model.

Usage

```
RunModel_CemaNeigeGR5J(InputsModel, RunOptions, Param)
```

Arguments

InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
Param	[numeric] vector of 7 parameters

GR5J X1	production store capacity [mm]
GR5J X2	intercatchment exchange coefficient [mm/d]
GR5J X3	routing store capacity [mm]
GR5J X4	unit hydrograph time constant [d]
GR5J X5	intercatchment exchange threshold [-]
CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-day melt coefficient [mm/°C/d]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see CreateRunOptions.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d]
$Precip	[numeric] series of input total precipitation [mm/d]
$Prod	[numeric] series of production store level [mm]
$Pn	[numeric] series of net rainfall [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store [mm/d]
$AE	[numeric] series of actual evapotranspiration [mm/d]
$Perc	[numeric] series of percolation (PERC) [mm/d]
$PR	[numeric] series of PR=Pn-Ps+Perc [mm/d]
$Q9	[numeric] series of UH1 outflow (Q9) [mm/d]
$Q1	[numeric] series of UH2 outflow (Q1) [mm/d]
$Rout	[numeric] series of routing store level [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/d]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/d]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/d]
$AExch	[numeric] series of actual exchange between catchments (1+2) [mm/d]
$QR	[numeric] series of routing store outflow (QR) [mm/d]
$QD	[numeric] series of direct flow from UH2 after exchange (QD) [mm/d]
$Qsim	[numeric] series of simulated discharge [mm/d]
$CemaNeigeLayers	[list] list of CemaNeige outputs (1 list per layer)
$CemaNeigeLayers[[iLayer]]$Pliq	[numeric] series of liquid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$Psol	[numeric] series of solid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$SnowPack	[numeric] series of snow pack [mm]
$CemaNeigeLayers[[iLayer]]$ThermalState	[numeric] series of snow pack thermal state [°C]
$CemaNeigeLayers[[iLayer]]$Gratio	[numeric] series of Gratio [0-1]
$CemaNeigeLayers[[iLayer]]$PotMelt	[numeric] series of potential snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$Melt	[numeric] series of actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$PliqAndMelt	[numeric] series of liquid precip. + actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$Temp	[numeric] series of air temperature [°C]
$StateEnd	[numeric] states at the end of the run: store & unit hydrographs levels [mm], CemaNeige states [mm & °C], see `CreateIniStates` for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Audrey Valéry, Claude Michel, Nicolas Le Moine, Charles Perrin, Vazken Andréassian

References

Le Moine, N. (2008), Le bassin versant de surface vu par le souterrain : une voie d'amélioration des performances et du réalisme des modèles pluie-débit ?, PhD thesis (french), UPMC, Paris, France.
Pushpalatha, R., C. Perrin, N. Le Moine, T. Mathevet and V. Andréassian (2011), A downward structural sensitivity analysis of hydrological models to improve low-flow simulation, Journal of Hydrology, 411(1-2), 66-76, doi:10.1016/j.jhydrol.2011.09.034.
Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": what is useful in a temperature-based snow-accounting routine? Part 1 - Comparison of six snow accounting routines on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.059.
Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": What is useful in a temperature-based snow-accounting routine? Part 2 - Sensitivity analysis of the Cemaneige snow accounting routine on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeigeGR4J, RunModel_CemaNeigeGR6J, RunModel_GR5J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## loading catchment data
data(L0123002)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_CemaNeigeGR5J, DatesR = BasinObs$DatesR,
                           Precip = BasinObs$P, PotEvap = BasinObs$E, TempMean = BasinObs$T,
                                  ZInputs = median(BasinInfo$HypsoData),
                                  HypsoData = BasinInfo$HypsoData, NLayers = 5)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR5J, InputsModel = InputsModel,
                               IndPeriod_Run = Ind_Run)

## simulation
Param <- c(179.139, -0.100, 203.815, 1.174, 2.478, 0.977, 2.774)
OutputsModel <- RunModel_CemaNeigeGR5J(InputsModel = InputsModel,
                                       RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_CemaNeigeGR6J

Run with the CemaNeigeGR6J hydrological model

Description

Function which performs a single run for the CemaNeige-GR6J daily lumped model.

Usage

```
RunModel_CemaNeigeGR6J(InputsModel, RunOptions, Param)
```

Arguments

InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
Param	[numeric] vector of 8 parameters

GR6J X1	production store capacity [mm]
GR6J X2	intercatchment exchange coefficient [mm/d]
GR6J X3	routing store capacity [mm]
GR6J X4	unit hydrograph time constant [d]
GR6J X5	intercatchment exchange threshold [-]
GR6J X6	coefficient for emptying exponential store [mm]
CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-day melt coefficient [mm/°C/d]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see CreateRunOptions.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d]
$Precip	[numeric] series of input total precipitation [mm/d]
$Prod	[numeric] series of production store level [mm]
$Pn	[numeric] series of net rainfall [mm/d]
$Ps	[numeric] series of the part of Ps filling the production store [mm/d]
$AE	[numeric] series of actual evapotranspiration [mm/d]
$Perc	[numeric] series of percolation (PERC) [mm/d]
$PR	[numeric] series of PR=PN-PS+PERC [mm/d]
$Q9	[numeric] series of UH1 outflow (Q9) [mm/d]

$Q1	[numeric] series of UH2 outflow (Q1) [mm/d]
$Rout	[numeric] series of routing store level [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/d]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/d]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/d]
$AExch	[numeric] series of actual exchange between catchments (1+2) [mm/d]
$QR	[numeric] series of routing store outflow (QR) [mm/d]
$QRExp	[numeric] series of exponential store outflow (QRExp) [mm/d]
$Exp	[numeric] series of exponential store level (negative) [mm]
$QD	[numeric] series of direct flow from UH2 after exchange (QD) [mm/d]
$Qsim	[numeric] series of Qsim [mm/d]
$CemaNeigeLayers	[list] list of CemaNeige outputs (1 list per layer)
$CemaNeigeLayers[[iLayer]]$Pliq	[numeric] series of liquid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$Psol	[numeric] series of solid precip. [mm/d]
$CemaNeigeLayers[[iLayer]]$SnowPack	[numeric] series of snow pack [mm]
$CemaNeigeLayers[[iLayer]]$ThermalState	[numeric] series of snow pack thermal state [°C]
$CemaNeigeLayers[[iLayer]]$Gratio	[numeric] series of Gratio [0-1]
$CemaNeigeLayers[[iLayer]]$PotMelt	[numeric] series of potential snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$Melt	[numeric] series of actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$PliqAndMelt	[numeric] series of liquid precip. + actual snow melt [mm/d]
$CemaNeigeLayers[[iLayer]]$Temp	[numeric] series of air temperature [°C]
$StateEnd	[numeric] states at the end of the run: store & unit hydrographs levels [mm], CemaNeige states [mm & °C], see `CreateIniStates` for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Audrey Valéry, Claude Michel, Charles Perrin, Raji Pushpalatha, Nicolas Le Moine, Vazken Andréassian

References

Pushpalatha, R., C. Perrin, N. Le Moine, T. Mathevet and V. Andréassian (2011), A downward structural sensitivity analysis of hydrological models to improve low-flow simulation, Journal of Hydrology, 411(1-2), 66-76, doi:10.1016/j.jhydrol.2011.09.034.

Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": what is useful in a temperature-based snow-accounting routine? Part 1 - Comparison of six snow accounting routines on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.059.

Valéry, A., V. Andréassian and C. Perrin (2014), "As simple as possible but not simpler": What is useful in a temperature-based snow-accounting routine? Part 2 - Sensitivity analysis of the Cemaneige snow accounting routine on 380 catchments, Journal of Hydrology, doi:10.1016/j.jhydrol.2014.04.058.

See Also

`RunModel_CemaNeigeGR4J`, `RunModel_CemaNeigeGR5J`, `RunModel_GR6J`, `CreateInputsModel`, `CreateRunOptions`.

Examples

```
library(airGR)

## loading catchment data
data(L0123002)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_CemaNeigeGR6J, DatesR = BasinObs$DatesR,
                           Precip = BasinObs$P, PotEvap = BasinObs$E, TempMean = BasinObs$T,
                                  ZInputs = median(BasinInfo$HypsoData),
                                  HypsoData = BasinInfo$HypsoData, NLayers = 5)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR6J, InputsModel = InputsModel,
                               IndPeriod_Run = Ind_Run)

## simulation
Param <- c(116.482, 0.500, 72.733, 1.224, 0.278, 30.333, 0.977, 2.776)
OutputsModel <- RunModel_CemaNeigeGR6J(InputsModel = InputsModel,
                                       RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`RunModel_GR1A` *Run with the GR1A hydrological model*

Description

Function which performs a single run for the GR1A annual lumped model over the test period.

Usage

```
RunModel_GR1A(InputsModel, RunOptions, Param)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Param`	[numeric] vector of 1 parameter

GR1A X1 model parameter [-]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/y]
$Precip	[numeric] series of input total precipitation [mm/y]
$Qsim	[numeric] series of simulated discharge [mm/y]
$StateEnd	[numeric] states at the end of the run (NULL) [-]

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Claude Michel

References

Mouelhi S. (2003), Vers une chaîne cohérente de modèles pluie-débit conceptuels globaux aux pas de temps pluriannuel, annuel, mensuel et journalier, PhD thesis (in French), ENGREF, Cemagref Antony, France.

See Also

`CreateInputsModel`, `CreateRunOptions`.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## conversion of example data from daily to yearly time step
TabSeries      <- data.frame(BasinObs$DatesR, BasinObs$P, BasinObs$E, BasinObs$T, BasinObs$Qmm)
TimeFormat      <- "daily"
NewTimeFormat   <- "yearly"
ConvertFun      <- c("sum", "sum", "mean", "sum")
YearFirstMonth  <- 09;
```

```
NewTabSeries    <- SeriesAggreg(TabSeries = TabSeries, TimeFormat = TimeFormat,
                                NewTimeFormat = NewTimeFormat, ConvertFun = ConvertFun,
                                YearFirstMonth = YearFirstMonth)
BasinObs        <- NewTabSeries
names(BasinObs) <- c("DatesR", "P", "E", "T", "Qmm")

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR1A, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%Y")=="1990"),
               which(format(BasinObs$DatesR, format = "%Y")=="1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR1A,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(0.840)
OutputsModel <- RunModel_GR1A(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`RunModel_GR2M` *Run with the GR2M hydrological model*

Description

Function which performs a single run for the GR2M monthly lumped model over the test period.

Usage

```
RunModel_GR2M(InputsModel, RunOptions, Param)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Param`	[numeric] vector of 2 parameters

GR2M X1	production store capacity [mm]
GR2M X2	groundwater exchange coefficient [-]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/month]
$Precip	[numeric] series of input total precipitation [mm/month]
$AE	[numeric] series of actual evapotranspiration [mm/month]
$Pn	[numeric] series of net rainfall (P1) [mm/month]
$Perc	[numeric] series of percolation (P2) [mm/month]
$PR	[numeric] series of PR=Pn+Perc (P3) [mm/month]
$Exch	[numeric] series of potential exchange between catchments [mm/month]
$Prod	[numeric] series of production store level [mm]
$Rout	[numeric] series of routing store level [mm]
$Qsim	[numeric] series of simulated discharge [mm/month]
$StateEnd	[numeric] states at the end of the run (production store level and routing store level) [mm], see `CreateIniStates` for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Claude Michel, Safouane Mouelhi

References

Mouelhi S. (2003), Vers une chaîne cohérente de modèles pluie-débit conceptuels globaux aux pas de temps pluriannuel, annuel, mensuel et journalier, PhD thesis (in French), ENGREF, Cemagref Antony, France.
Mouelhi, S., C. Michel, C. Perrin and V. Andréassian (2006), Stepwise development of a two-parameter monthly water balance model, Journal of Hydrology, 318(1-4), 200-214, doi:10.1016/j.jhydrol.2005.06.014.

See Also

`CreateInputsModel`, `CreateRunOptions`, `CreateIniStates`.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)
```

```
## conversion of example data from daily to monthly time step
TabSeries      <- data.frame(BasinObs$DatesR, BasinObs$P, BasinObs$E, BasinObs$T, BasinObs$Qmm)
TimeFormat      <- "daily"
NewTimeFormat   <- "monthly"
ConvertFun      <- c("sum", "sum", "mean", "sum")
NewTabSeries    <- SeriesAggreg(TabSeries = TabSeries, TimeFormat = TimeFormat,
                                NewTimeFormat = NewTimeFormat, ConvertFun = ConvertFun)
BasinObs        <- NewTabSeries
names(BasinObs) <- c("DatesR", "P", "E", "T", "Qmm")

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR2M, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%m/%Y")=="01/1990"),
               which(format(BasinObs$DatesR, format = "%m/%Y")=="12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR2M,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(265.072, 1.040)
OutputsModel <- RunModel_GR2M(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`RunModel_GR4H` *Run with the GR4H hydrological model*

Description

Function which performs a single run for the GR4H hourly lumped model.

Usage

```
RunModel_GR4H(InputsModel, RunOptions, Param)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Param`	[numeric] vector of 4 parameters

GR4H X1	production store capacity [mm]
GR4H X2	groundwater exchange coefficient [mm/h]
GR4H X3	routing store capacity [mm]
GR4H X4	unit hydrograph time constant [h]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/h]
$Precip	[numeric] series of input total precipitation [mm/h]
$Prod	[numeric] series of production store level [mm]
$AE	[numeric] series of actual evapotranspiration [mm/h]
$Perc	[numeric] series of percolation (PERC) [mm/h]
$PR	[numeric] series of PR=Pn-Ps+Perc [mm/h]
$Q9	[numeric] series of UH1 outflow (Q9) [mm/h]
$Q1	[numeric] series of UH2 outflow (Q1) [mm/h]
$Rout	[numeric] series of routing store level [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/h]
$AExch	[numeric] series of actual exchange between catchments (1+2) [mm/h]
$QR	[numeric] series of routing store outflow (QR) [mm/h]
$QD	[numeric] series of direct flow from UH2 after exchange (QD) [mm/h]
$Qsim	[numeric] series of simulated discharge [mm/h]
$StateEnd	[numeric] states at the end of the run (res. levels, UH1 levels, UH2 levels) [mm], see `CreateIniStates` for more

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Charles Perrin, Thibaut Mathevet

References

Mathevet, T. (2005), Quels modèles pluie-débit globaux pour le pas de temps horaire ? Développement empirique et comparaison de modèles sur un large échantillon de bassins versants, PhD thesis (in French), ENGREF - Cemagref (Antony), Paris, France.
Le Moine, N. (2008), Le bassin versant de surface vu par le souterrain : une voie d'amélioration des performances et du réalisme des modèles pluie-débit ?, PhD thesis (french), UPMC, Paris, France.

See Also

RunModel_GR4J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## load of catchment data
data(L0123003)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4H, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y %H:%M")=="01/03/2004 00:00"),
            which(format(BasinObs$DatesR, format = "%d/%m/%Y %H:%M")=="31/12/2008 23:00"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4H,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(521.113, -2.918, 218.009, 4.124)
OutputsModel <- RunModel_GR4H(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_GR4J *Run with the GR4J hydrological model*

Description

Function which performs a single run for the GR4J daily lumped model over the test period.

Usage

```
RunModel_GR4J(InputsModel, RunOptions, Param)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Param`	[numeric] vector of 4 parameters

GR4J X1	production store capacity [mm]
GR4J X2	intercatchment exchange coefficient [mm/d]
GR4J X3	routing store capacity [mm]
GR4J X4	unit hydrograph time constant [d]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d]
$Precip	[numeric] series of input total precipitation [mm/d]
$Prod	[numeric] series of production store level [mm]
$Pn	[numeric] series of net rainfall [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store [mm/d]
$AE	[numeric] series of actual evapotranspiration [mm/d]
$Perc	[numeric] series of percolation (PERC) [mm/d]
$PR	[numeric] series of PR=Pn-Ps+Perc [mm/d]
$Q9	[numeric] series of UH1 outflow (Q9) [mm/d]
$Q1	[numeric] series of UH2 outflow (Q1) [mm/d]
$Rout	[numeric] series of routing store level [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/d]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/d]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/d]
$AExch	[numeric] series of actual exchange between catchments (1+2) [mm/d]
$QR	[numeric] series of routing store outflow (QR) [mm/d]
$QD	[numeric] series of direct flow from UH2 after exchange (QD) [mm/d]
$Qsim	[numeric] series of simulated discharge [mm/d]
$StateEnd	[numeric] states at the end of the run (res. levels, UH1 levels, UH2 levels) [mm], see `CreateIniStates` for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Claude Michel, Charles Perrin

References

Perrin, C., C. Michel and V. Andréassian (2003), Improvement of a parsimonious model for streamflow simulation, Journal of Hydrology, 279(1-4), 275-289, doi:10.1016/S0022-1694(03)00225-7.

See Also

RunModel_GR5J, RunModel_GR6J, RunModel_CemaNeigeGR4J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(257.238, 1.012, 88.235, 2.208)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_GR5J *Run with the GR5J hydrological model*

Description

Function which performs a single run for the GR5J daily lumped model over the test period.

Usage

```
RunModel_GR5J(InputsModel, RunOptions, Param)
```

Arguments

`InputsModel`	[object of class *InputsModel*] see `CreateInputsModel` for details
`RunOptions`	[object of class *RunOptions*] see `CreateRunOptions` for details
`Param`	[numeric] vector of 5 parameters

GR5J X1	production store capacity [mm]
GR5J X2	intercatchment exchange coefficient [mm/d]
GR5J X3	routing store capacity [mm]
GR5J X4	unit hydrograph time constant [d]
GR5J X5	intercatchment exchange threshold [-]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d]
$Precip	[numeric] series of input total precipitation [mm/d]
$Prod	[numeric] series of production store level [mm]
$Pn	[numeric] series of net rainfall [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store [mm/d]
$AE	[numeric] series of actual evapotranspiration [mm/d]
$Perc	[numeric] series of percolation (PERC) [mm/d]
$PR	[numeric] series of PR=Pn-Ps+Perc [mm/d]
$Q9	[numeric] series of UH1 outflow (Q9) [mm/d]
$Q1	[numeric] series of UH2 outflow (Q1) [mm/d]
$Rout	[numeric] series of routing store level [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/d]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/d]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/d]
$AExch	[numeric] series of actual exchange between catchments (1+2) [mm/d]
$QR	[numeric] series of routing store outflow (QR) [mm/d]
$QD	[numeric] series of direct flow from UH2 after exchange (QD) [mm/d]
$Qsim	[numeric] series of simulated discharge [mm/d]
$StateEnd	[numeric] states at the end of the run (res. levels, UH1 levels, UH2 levels) [mm], see `CreateIniStates` for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Claude Michel, Nicolas Le Moine

References

Le Moine, N. (2008), Le bassin versant de surface vu par le souterrain : une voie d'amélioration des performances et du réalisme des modèles pluie-débit ?, PhD thesis (french), UPMC, Paris, France.
Pushpalatha, R., C. Perrin, N. Le Moine, T. Mathevet, and V. Andréassian (2011), A downward structural sensitivity analysis of hydrological models to improve low-flow simulation, Journal of Hydrology, 411(1-2), 66-76, doi:10.1016/j.jhydrol.2011.09.034.

See Also

RunModel_GR4J, RunModel_GR6J, RunModel_CemaNeigeGR5J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR5J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR5J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(245.918, 1.027, 90.017, 2.198, 0.434)
OutputsModel <- RunModel_GR5J(InputsModel = InputsModel,
                              RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_GR6J *Run with the GR6J hydrological model*

Description

Function which performs a single run for the GR6J daily lumped model over the test period.

Usage

```
RunModel_GR6J(InputsModel, RunOptions, Param)
```

Arguments

InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
Param	[numeric] vector of 6 parameters

GR6J X1	production store capacity [mm]
GR6J X2	intercatchment exchange coefficient [mm/d]
GR6J X3	routing store capacity [mm]
GR6J X4	unit hydrograph time constant [d]
GR6J X5	intercatchment exchange threshold [-]
GR6J X6	coefficient for emptying exponential store [mm]

Details

For further details on the model, see the references section. For further details on the argument structures and initialisation options, see CreateRunOptions.

Value

list list containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d]
$Precip	[numeric] series of input total precipitation [mm/d]
$Prod	[numeric] series of production store level [mm]
$Pn	[numeric] series of net rainfall [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store [mm/d]
$AE	[numeric] series of actual evapotranspiration [mm/d]
$Perc	[numeric] series of percolation (PERC) [mm/d]
$PR	[numeric] series of PR=Pn-Ps+Perc [mm/d]
$Q9	[numeric] series of UH1 outflow (Q9) [mm/d]
$Q1	[numeric] series of UH2 outflow (Q1) [mm/d]
$Rout	[numeric] series of routing store level [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/d]

$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/d]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/d]
$AExch	[numeric] series of actual exchange between catchments (1+2) [mm/d]
$QR	[numeric] series of routing store outflow (QR) [mm/d]
$QRExp	[numeric] series of exponential store outflow (QRExp) [mm/d]
$Exp	[numeric] series of exponential store level (negative) [mm]
$QD	[numeric] series of direct flow from UH2 after exchange (QD) [mm/d]
$Qsim	[numeric] series of Qsim [mm/d]
$StateEnd	[numeric] states at the end of the run (res. levels, UH1 levels, UH2 levels) [mm], see CreateIniStates for more details

(refer to the provided references or to the package source code for further details on these model outputs)

Author(s)

Laurent Coron, Claude Michel, Charles Perrin, Raji Pushpalatha, Nicolas Le Moine

References

Pushpalatha, R., C. Perrin, N. Le Moine, T. Mathevet and V. Andréassian (2011), A downward structural sensitivity analysis of hydrological models to improve low-flow simulation, Journal of Hydrology, 411(1-2), 66-76, doi:10.1016/j.jhydrol.2011.09.034.

See Also

RunModel_GR4J, RunModel_GR5J, RunModel_CemaNeigeGR6J, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## loading catchment data
data(L0123001)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR6J, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="01/01/1990"),
               which(format(BasinObs$DatesR, format = "%d/%m/%Y")=="31/12/1999"))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR6J,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)
```

```
## simulation
Param <- c(242.257, 0.637, 53.517, 2.218, 0.424, 4.759)
OutputsModel <- RunModel_GR6J(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel, Qobs = BasinObs$Qmm[Ind_Run])

## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Qobs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`SeriesAggreg` *Conversion of time series to another time step (aggregation only)*

Description

Conversion of time series to another time step (aggregation only).
Warning : on the aggregated outputs, the dates correspond to the beginning of the time step
(e.g. for daily time-series 01/03/2005 00:00 = value for period 01/03/2005 00:00 - 01/03/2005 23:59)
(e.g. for monthly time-series 01/03/2005 00:00 = value for period 01/03/2005 00:00 - 31/03/2005 23:59)
(e.g. for yearly time-series 01/03/2005 00:00 = value for period 01/03/2005 00:00 - 28/02/2006 23:59)

Usage

```
SeriesAggreg(TabSeries, TimeFormat, NewTimeFormat, ConvertFun,
  YearFirstMonth = 1, TimeLag = 0, verbose = TRUE)
```

Arguments

`TabSeries`	[POSIXt+numeric] data.frame containing the vector of dates (POSIXt) and the time series values numeric)
`TimeFormat`	[character] desired format (i.e. `"hourly"`, `"daily"`, `"monthly"` or `"yearly"`)
`NewTimeFormat`	[character] desired format (i.e. `"hourly"`, `"daily"`, `"monthly"` or `"yearly"`)
`ConvertFun`	[character] names of aggregation functions (e.g. for P[mm], T[degC], Q[mm] : `ConvertFun = c("sum", "mean", "sum")`)
`YearFirstMonth`	(optional) [numeric] integer used when `NewTimeFormat = "yearly"` to set when the starting month of the year (e.g. 01 for calendar year or 09 for hydrological year starting in September)
`TimeLag`	(optional) [numeric] numeric indicating a time lag (in seconds) for the time series aggregation (especially useful to aggregate hourly time series in daily time series)
`verbose`	(optional) [boolean] boolean indicating if the function is run in verbose mode or not, default = `FALSE`

Value

POSIXct+numeric data.frame containing a vector of aggregated dates (POSIXct) and time series values numeric)

Author(s)

Laurent Coron

Examples

```
library(airGR)

## loading catchment data
data(L0123002)

## preparation of the initial time series data frame at the daily time step
TabSeries  <- data.frame(BasinObs$DatesR, BasinObs$P, BasinObs$E, BasinObs$T, BasinObs$Qmm)
TimeFormat <- "daily"

## conversion at the monthly time step
NewTimeFormat <- "monthly"
ConvertFun    <- c("sum", "sum", "mean", "sum")
NewTabSeries  <- SeriesAggreg(TabSeries = TabSeries, TimeFormat, NewTimeFormat, ConvertFun)

## conversion at the yearly time step
NewTimeFormat <- "yearly"
ConvertFun <- c("sum", "sum", "mean", "sum")
NewTabSeries <- SeriesAggreg(TabSeries = TabSeries, TimeFormat, NewTimeFormat, ConvertFun)
```

TransfoParam *Transformation of the parameters using the provided function*

Description

Function which transforms model parameters using the provided function (from raw to transformed parameters and vice versa).

Usage

```
TransfoParam(ParamIn, Direction, FUN_TRANSFO)
```

Arguments

ParamIn	[numeric] matrix of parameter sets (sets in line, parameter values in column)
Direction	[character] direction of the transformation: use "RT" for Raw -> Transformed and "TR" for Transformed -> Raw
FUN_TRANSFO	[function] model parameters transformation function (e.g. TransfoParam_GR4J, TransfoParam_CemaNeige)

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron

See Also

TransfoParam_GR4J, TransfoParam_GR5J, TransfoParam_GR6J, TransfoParam_CemaNeige

Examples

```
library(airGR)

## transformation Raw -> Transformed for the GR4J model
Xraw  <- matrix(c(+221.41, -3.63,  +30.00, +1.37,
                  +347.23, -1.03,  +60.34, +1.76,
                  +854.06, -0.10, +148.41, +2.34),
                  ncol = 4, byrow = TRUE)
Xtran <- TransfoParam(ParamIn = Xraw, Direction = "RT", FUN_TRANSFO = TransfoParam_GR4J)

## transformation Transformed -> Raw for the GR4J model
Xtran <- matrix(c(+3.60, -2.00, +3.40, -9.10,
                  +3.90, -0.90, +4.10, -8.70,
                  +4.50, -0.10, +5.00, -8.10),
                  ncol = 4, byrow = TRUE)
Xraw  <- TransfoParam(ParamIn = Xtran, Direction = "TR", FUN_TRANSFO = TransfoParam_GR4J)
```

TransfoParam_CemaNeige

Transformation of the parameters of the CemaNeige module

Description

Function which transforms model parameters of the CemaNeige module (from raw to transformed parameters and vice versa).

Usage

```
TransfoParam_CemaNeige(ParamIn, Direction)
```

Arguments

ParamIn	[numeric] matrix of parameter sets (sets in line, parameter values in column)
Direction	[character] direction of the transformation: use "RT" for Raw -> Transformed and "TR" for Transformed -> Raw

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron

See Also

TransfoParam, TransfoParam_GR4J, TransfoParam_GR5J, TransfoParam_GR6J

Examples

```
library(airGR)

## transformation Raw -> Transformed for the CemaNeige module
Xraw  <- matrix(c(+0.19, +1.73,
                  +0.39, +2.51,
                  +0.74, +4.06),
                  ncol = 2, byrow = TRUE)
Xtran <- TransfoParam_CemaNeige(ParamIn = Xraw , Direction = "RT")

## transformation Transformed -> Raw for the CemaNeige module
Xtran <- matrix(c(-6.26, +0.55,
                  -2.13, +0.92,
                  +4.86, +1.40),
                  ncol = 2, byrow = TRUE)
Xraw  <- TransfoParam_CemaNeige(ParamIn = Xtran, Direction = "TR")
```

TransfoParam_GR1A *Transformation of the parameters of the GR1A model*

Description

Function which transforms model parameters of the GR1A model (from real to transformed parameters and vice versa).

Usage

```
TransfoParam_GR1A(ParamIn, Direction)
```

Arguments

ParamIn	[numeric] matrix of parameter sets (sets in line, parameter values in column)
Direction	[character] direction of the transformation: use "RT" for Raw -> Transformed and "TR" for Transformed -> Raw

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron

`TransfoParam_GR2M` *Transformation of the parameters of the GR2M model*

Description

Function which transforms model parameters of the GR2M model (from real to transformed parameters and vice versa).

Usage

```
TransfoParam_GR2M(ParamIn, Direction)
```

Arguments

`ParamIn`	[numeric] matrix of parameter sets (sets in line, parameter values in column)
`Direction`	[character] direction of the transformation: use `"RT"` for Raw -> Transformed and `"TR"` for Transformed -> Raw

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron

`TransfoParam_GR4H` *Transformation of the parameters of the GR4H model*

Description

Function which transforms model parameters of the GR4H model (from real to transformed parameters and vice versa).

Usage

```
TransfoParam_GR4H(ParamIn, Direction)
```

Arguments

ParamIn	[numeric] matrix of parameter sets (sets in line, parameter values in column)
Direction	[character] direction of the transformation: use "RT" for Raw -> Transformed and "TR" for Transformed -> Raw

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron, Claude Michel, Thibault Mathevet

TransfoParam_GR4J *Transformation of the parameters of the GR4J model*

Description

Function which transforms model parameters of the GR4J model (from real to transformed parameters and vice versa).

Usage

```
TransfoParam_GR4J(ParamIn, Direction)
```

Arguments

ParamIn	[numeric] matrix of parameter sets (sets in line, parameter values in column)
Direction	[character] direction of the transformation: use "RT" for Raw -> Transformed and "TR" for Transformed -> Raw

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron

See Also

TransfoParam, TransfoParam_GR5J, TransfoParam_GR6J, TransfoParam_CemaNeige

Examples

```
library(airGR)

## transformation Raw -> Transformed for the GR4J model
Xraw  <- matrix(c(+221.41, -3.63,  +30.00, +1.37,
                  +347.23, -1.03,  +60.34, +1.76,
                  +854.06, -0.10, +148.41, +2.34),
                  ncol = 4, byrow = TRUE)
Xtran <- TransfoParam_GR4J(ParamIn = Xraw , Direction = "RT")

## transformation Transformed -> Raw for the GR4J model
Xtran <- matrix(c(+3.60, -2.00, +3.40, -9.10,
                  +3.90, -0.90, +4.10, -8.70,
                  +4.50, -0.10, +5.00, -8.10),
                  ncol = 4, byrow = TRUE)
Xraw  <- TransfoParam_GR4J(ParamIn = Xtran, Direction = "TR")
```

TransfoParam_GR5J *Transformation of the parameters of the GR5J model*

Description

Function which transforms model parameters of the GR5J model (from real to transformed parameters and vice versa).

Usage

```
TransfoParam_GR5J(ParamIn, Direction)
```

Arguments

ParamIn	[numeric] matrix of parameter sets (sets in line, parameter values in column)
Direction	[character] direction of the transformation: use "RT" for Raw -> Transformed and "TR" for Transformed -> Raw

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron

See Also

TransfoParam, TransfoParam_GR4J, TransfoParam_GR6J, TransfoParam_CemaNeige

Examples

```
library(airGR)

## transformation Raw -> Transformed for the GR5J model
Xraw  <- matrix(c(+221.41, -2.65,  +27.11, +1.37, -0.76,
                  +347.23, -0.64,  +60.34, +1.76, +0.30,
                  +854.01, -0.10, +148.41, +2.34, +0.52),
                  ncol = 5, byrow = TRUE)
Xtran <- TransfoParam_GR5J(ParamIn = Xraw , Direction = "RT")

## transformation Transformed -> Raw for the GR5J model
Xtran <- matrix(c(+3.60, -1.70, +3.30, -9.10, -0.70,
                  +3.90, -0.60, +4.10, -8.70, +0.30,
                  +4.50, -0.10, +5.00, -8.10, +0.50),
                  ncol = 5, byrow = TRUE)
Xraw  <- TransfoParam_GR5J(ParamIn = Xtran, Direction = "TR")
```

`TransfoParam_GR6J` *Transformation of the parameters of the GR6J model*

Description

Function which transforms model parameters of the GR6J model (from real to transformed parameters and vice versa).

Usage

```
TransfoParam_GR6J(ParamIn, Direction)
```

Arguments

`ParamIn`	[numeric] matrix of parameter sets (sets in line, parameter values in column)
`Direction`	[character] direction of the transformation: use `"RT"` for Raw -> Transformed and `"TR"` for Transformed -> Raw

Value

ParamOut [numeric] matrix of parameter sets (sets in line, parameter values in column)

Author(s)

Laurent Coron

See Also

`TransfoParam`, `TransfoParam_GR4J`, `TransfoParam_GR5J`, `TransfoParam_CemaNeige`

Examples

```
library(airGR)

## transformation Raw -> Transformed for the GR6J model
Xraw  <- matrix(c(+221.41, -1.18,  +27.11, 1.37, -0.18,  +20.09,
                  +347.23, -0.52,  +60.34, 1.76, +0.02,  +54.60,
                  +854.06, +0.52, +148.41, 2.34, +0.22, +148.41),
                  ncol = 6, byrow = TRUE)
Xtran <- TransfoParam_GR6J(ParamIn = Xraw , Direction = "RT")

## transformation Transformed -> Raw for the GR6J model
Xtran <- matrix(c(+3.60, -1.00, +3.30, -9.10, -0.90, +3.00,
                  +3.90, -0.50, +4.10, -8.70, +0.10, +4.00,
                  +4.50, +0.50, +5.00, -8.10, +1.10, +5.00),
                  ncol = 6, byrow = TRUE)
Xraw  <- TransfoParam_GR6J(ParamIn = Xtran, Direction = "TR")
```

Index