Package 'airGR'

July 22, 2025

Type Package

Title Suite of GR Hydrological Models for Precipitation-Runoff Modelling

Version 1.7.6

Date 2023-10-25

Depends R (>= 3.1.0)

Imports graphics, grDevices, stats, utils

Suggests knitr, markdown, rmarkdown, caRamel, coda, DEoptim, FME, ggmcmc, Rmalschains, GGally, ggplot2, testthat

Description Hydrological modelling tools developed at INRAE-Antony (HYCAR Research Unit, France). The package includes several conceptual rainfall-runoff models (GR4H, GR5H, GR4J, GR5J, GR6J, GR2M, GR1A) that can be applied either on a lumped or semi-distributed way. A snow accumulation and melt model (CemaNeige) and the associated functions for the calibration and evaluation of models are also included. Use help(airGR) for package description and references.

License GPL-2

URL https://hydrogr.github.io/airGR/

BugReports https://gitlab.irstea.fr/HYCAR-Hydro/airgr/-/issues

NeedsCompilation yes

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.1.1

Author Laurent Coron [aut, trl] (ORCID: <https://orcid.org/0000-0002-1503-6204>), Olivier Delaigue [aut, cre] (ORCID: <https://orcid.org/0000-0002-7668-8468>), Guillaume Thirel [aut, ths] (ORCID: <https://orcid.org/0000-0002-1444-1830>), David Dorchies [aut] (ORCID: <https://orcid.org/0000-0002-6595-7984>), Charles Perrin [aut, ths] (ORCID: <https://orcid.org/0000-0001-8552-1881>),

Claude Michel [aut, ths],
Vazken Andréassian [ctb, ths] (ORCID:
<https://orcid.org/0000-0001-7124-9303>),
François Bourgin [ctb] (ORCID: <https://orcid.org/0000-0002-2820-7260>),
Pierre Brigode [ctb] (ORCID: <https://orcid.org/0000-0001-8257-0741>),
Nicolas Le Moine [ctb],
Thibaut Mathevet [ctb] (ORCID: <https://orcid.org/0000-0002-4142-4454>),
Safouane Mouelhi [ctb],
Ludovic Oudin [ctb] (ORCID: <https://orcid.org/0000-0002-3712-0933>),
Raji Pushpalatha [ctb],
Audrey Valéry [ctb]

Maintainer Olivier Delaigue <airGR@inrae.fr>

Repository CRAN

Date/Publication 2023-10-26 07:30:05 UTC

Contents

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

Description

Hydrological modelling tools developed at INRAE-Antony (HYCAR Research Unit, France). The package includes several conceptual rainfall-runoff models (GR4H, GR5H, GR4J, GR5J, GR6J, GR2M, GR1A) that can be applied either on a lumped or semi-distributed way. A snow accumulation and melt model (CemaNeige) and the associated functions for the calibration and evaluation of models are also included. Use help(airGR) for package description and references. 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

Seven hydrological models and one snow melt and accumulation model are implemented in airGR. The hydrological models can be applied either on a lumped way or on a semi-distributed way (on sub-catchments). The snow model can either be used alone or with the daily or hourly hydrological models. Naturally each hydrological model can also be used alone.
These models can be called within airGR using the following functions:

- `RunModel_GR4H`: four-parameter hourly lumped hydrological model (Mathevet, 2005)
- `RunModel_GR5H`: five-parameter hourly lumped hydrological model (Ficchi, 2017; Ficchi *et al.*, 2019)
- `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; Riboust *et al.*, 2019)
- `RunModel_CemaNeigeGR4H`: combined use of GR4H and CemaNeige
- `RunModel_CemaNeigeGR5H`: combined use of GR5H and CemaNeige
- `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.

To get started with the package, you can refer to the 'get_started' vignette (`vignette("V01_get_started", package = "airGR")`). To know how to use the models on a semi-distributed way, you can refer to the 'sd_model' vignette (`vignette("V05_sd_model", package = "airGR")`). For more information, please visit the airGR website.

— References

- Ficchi, A. (2017). An adaptive hydrological model for multiple time-steps: Diagnostics and improvements based on fluxes consistency. PhD thesis, UPMC - Irstea Antony, Paris, France.
- Ficchi, A., Perrin, C. and Andréassian, V. (2019). Hydrological modelling at multiple sub-daily time steps: model improvement via flux-matching. Journal of Hydrology, 575, 1308-1327, doi:10.1016/j.jhydrol.2019.05.084.
- 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., Michel, C., Perrin, C. and Andréassian, V. (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., Michel, C., Perrin, C. and Andréassian, V. (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., Michel, C. and Andréassian, V. (2003). Improvement of a parsimonious model for streamflow simulation. Journal of Hydrology, 279(1-4), 275-289, doi:10.1016/S0022-1694(03)002257.
- Pushpalatha, R., Perrin, C., Le Moine, N., Mathevet, T. and Andréassian, V. (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.
- Riboust, P., Thirel, G., Le Moine N. and Ribstein P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.1016/j.jhydrol.2014.04.058.
- Valéry, A., Andréassian, V. and Perrin, C. (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:10.1016/j.jhydrol.2014.04.058.

BasinInfo *Data sample: characteristics of a different catchments*

Description

L0123001, L0123002 and L0123003 are fictional catchments.
X0310010 contains actual data from the Durance River at Embrun [La Clapière] (Hautes-Alpes, France).

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 different catchments*

Description

L0123001, L0123002 or L0123003 are fictional catchments.
X0310010 contains actual data from the Durance River at Embrun [La Clapière] (Hautes-Alpes, France). The flows are provided by Electricity of France (EDF) and were retrieved from the Banque Hydro database (http://www.hydro.eaufrance.fr). The meteorological forcing are derived from the SAFRAN reanalysis from Météo-France (Vidal et al., 2010).
R-object containing the times series of precipitation, temperature, potential evapotranspiration and discharge. X0310010 contains in addition MODIS snow cover area (SCA) data retrieved from the National Snow and Ice Data Center (NSIDC) repository (https://nsidc.org/). Five SCA time series are given, corresponding to 5 elevation bands of the CemaNeige model (default configuration). SCA data for days with important cloudiness (> 40 %) were set to missing values for the sake of

data representativeness. .

Times series for L0123001, L0123002 and X0310010 are at the daily time step for use with daily models such as GR4J, GR5J, GR6J, CemaNeigeGR4J, CemaNeigeGR5J and CemaNeigeGR6J.
Times series for X0310010 are provided in order to test hysteresis version of CemaNeige (see `CreateRunOptions` (Riboust et al., 2019).
Times series for L0123003 are at the hourly time step for use with hourly models such as GR4H or GR5H.

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]

References

Riboust, P., Thirel, G., Le Moine, N. and Ribstein P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.2478/johh20180004.

Vidal, J.-P., Martin, E., Franchistéguy, L., Baillon, M. and Soubeyroux, J. (2010). A 50-year high-resolution atmospheric reanalysis over France with the Safran system. International Journal of Climatology, 30, 1627–1644, doi:10.1002/joc.2003.

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	(deprecated) [function] error criterion function (e.g. ErrorCrit_RMSE, ErrorCrit_NSE)
FUN_CALIB	(deprecated) [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
...	Further arguments to pass to RunModel

Value

[list] see Calibration_Michel

Author(s)

Laurent Coron, Olivier Delaigue

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 = "%Y-%m-%d")=="1990-01-01"),
```

```
                which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

## 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, Obs = 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_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, Obs = 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, Obs = 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 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	(deprecated) [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
...	(optional) arguments to pass to RunModel

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, Charles Perrin, Thibault Mathevet, Olivier Delaigue, Guillaume Thirel, David Dorchies

References

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

See Also

Calibration, RunModel_GR4J, TransfoParam, 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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

## 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, Obs = 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)

## 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, Obs = 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, Obs = 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, IsHyst = FALSE, IsSD = FALSE,
                   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
`IsHyst`	[boolean] boolean indicating if the hysteresis version of CemaNeige is used. See details
`IsSD`	[boolean] boolean indicating if the semi-distributed lag model is used. See details

`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.

— CemaNeige version

If `IsHyst = FALSE`, the original CemaNeige version from Valéry et al. (2014) is used.
If `IsHyst = TRUE`, the CemaNeige version from Riboust et al. (2019) is used. Compared to the original version, this version of CemaNeige needs two more parameters and it includes a representation of the hysteretic relationship between the Snow Cover Area (SCA) and the Snow Water Equivalent

(SWE) in the catchment. The hysteresis included in airGR is the Modified Linear hysteresis (LH*); it is represented on panel b) of Fig. 3 in Riboust et al. (2019). Riboust et al. (2019) advise to use the LH* version of CemaNeige with parameters calibrated using an objective function combining 75 % of KGE calculated on discharge simulated from a rainfall-runoff model compared to observed discharge and 5 % of KGE calculated on SCA on 5 CemaNeige elevation bands compared to satellite (e.g. MODIS) SCA (see Eq. (18), Table 3 and Fig. 6). Riboust et al. (2019)'s tests were realized with GR4J as the chosen rainfall-runoff model.

If `InputsModel` parameter has been created for using a semi-distributed (SD) model (See `CreateInputsModel`), the parameter `isSD` should be set to `TRUE`.

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, Olivier Delaigue, Guillaume Thirel, David Dorchies

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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

## 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, Obs = 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_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, Obs = 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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`CreateErrorCrit_GAPX` *Creation of the ErrorCrit_GAPX function*

Description

Function which creates the function `ErrorCrit_GAPX`.

The produced function `ErrorCrit_GAPX` allows computing an error criterion based on the GAPX formula proposed by Lavenne et al. (2019).

Usage

```
CreateErrorCrit_GAPX(FUN_TRANSFO)
```

Arguments

`FUN_TRANSFO`	[function] The parameter transformation function used with the model

Details

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

Value

[function] function ErrorCrit_GAPX embedding the parameter transformation function used with the model

Author(s)

David Dorchies

References

de Lavenne, A., Andréassian, V., Thirel, G., Ramos, M.-H. and Perrin, C. (2019). A Regularization Approach to Improve the Sequential Calibration of a Semidistributed Hydrological Model. Water Resources Research 55, 8821–8839. doi:10.1029/2018WR024266

See Also

CreateInputsCrit, ErrorCrit_RMSE, ErrorCrit_NSE, ErrorCrit_KGE, ErrorCrit_KGE2

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)

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

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

## simulation
Param <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel,
                              RunOptions = RunOptions, Param = Param)

## Creation of the ErrorCrit GAPX function
ErrorCrit_GAPX <- CreateErrorCrit_GAPX(TransfoParam_GR4J)

## The "a priori" parameters for GAPX
AprParamR <- c(X1 = 157, X2 = 0.8, X3 = 100, X4 = 1.5)
AprParamT <- TransfoParam_GR4J(AprParamR, "RT")

## Single efficiency criterion: GAPX with a priori parameters
InputsCrit <- CreateInputsCrit(ErrorCrit_GAPX,
```

```
                                InputsModel,
                                RunOptions,
                                Obs = AprParamT,
                                VarObs = "ParamT")
ErrorCrit <- ErrorCrit_GAPX(InputsCrit, OutputsModel)
str(ErrorCrit)
```

`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,
                IsHyst = FALSE, IsIntStore = FALSE,
                ProdStore = 350, RoutStore = 90,
                ExpStore = NULL, IntStore = NULL,
                UH1 = NULL, UH2 = NULL,
                GCemaNeigeLayers = NULL, eTGCemaNeigeLayers = NULL,
                GthrCemaNeigeLayers = NULL, GlocmaxCemaNeigeLayers = NULL,
                SD = 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
`IsHyst`	[boolean] boolean indicating if the hysteresis version of CemaNeige is used. See details
`IsIntStore`	[boolean] boolean indicating if the interception store is used in GR5H. See details
`ProdStore`	[numeric] production store level [mm] for all GR models except GR1A
`RoutStore`	[numeric] routing store level [mm] for all GR models except GR1A
`ExpStore`	(optional) [numeric] series of exponential store level (negative) [mm] for the GR6J model
`IntStore`	(optional) [numeric] series of rainfall neutralisation or interception store level [mm] for the GR5H 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

`GthrCemaNeigeLayers`
: (optional) [numeric] melt threshold [mm], possibly used to create the CemaNeige model initial state in case the Linear Hysteresis version is used

`GlocmaxCemaNeigeLayers`
: (optional) [numeric] local melt threshold for hysteresis [mm], possibly used to create the CemaNeige model initial state in case the Linear Hysteresis version is used

`SD`
: (optional) [list] of [numeric] states of delayed upstream flows for semi-distributed models, the nature of the state and the unit depend on the model and the unit of the upstream flow

`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 models GR4H and GR5H). 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 `NLayers` argument. `eTGCemaNeigeLayers` values can be negative.
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 and GR5H), even if some states do not exist (e.g. *UHUH1* for GR2M).
If CemaNeige is not used, *$CemaNeigeLayers$G*, *$CemaNeigeLayers$eTG* *$CemaNeigeLayers$GthrCemaNeigeLayers* and *$CemaNeigeLayers$GlocmaxCemaNeigeLayers* 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*, *$eTG*, *$GthrCemaNeigeLayers* and *$GlocmaxCemaNeigeL*

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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))


### 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,
                             GthrCemaNeigeLayers = NULL, GlocmaxCemaNeigeLayers = 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(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 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,
                             GthrCemaNeigeLayers = NULL, GlocmaxCemaNeigeLayers = 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(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 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. This function is used to define whether the user wants to calculate a single criterion, multiple criteria in the same time, or a composite criterion, which averages several criteria.

Usage

```
CreateInputsCrit(FUN_CRIT, InputsModel, RunOptions,
                 Obs, VarObs = "Q", BoolCrit = NULL,
                 transfo = "", Weights = NULL,
                 epsilon = NULL,
                 warnings = TRUE)
```

Arguments

`FUN_CRIT`	[function (atomic or list)] 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
`Obs`	[numeric (atomic or list)] series of observed variable ([mm/time step] for discharge or SWE, [-] for SCA)
`VarObs`	(optional) [character (atomic or list)] names of the observed variable (`"Q"` by default, or one of `"SCA"`, `"SWE"`])
`BoolCrit`	(optional) [boolean (atomic or list)] boolean (the same length as `Obs`) giving the time steps to consider in the computation (all time steps are considered by default. See details)
`transfo`	(optional) [character (atomic or list)] name of the transformation applied to the variables (e.g. `""`, `"sqrt"`, `"log"`, `"inv"`, `"sort"`, `"boxcox"` or a numeric value for power transformation . See details)
`Weights`	(optional) [numeric (atomic or list)] vector of weights necessary to calculate a composite criterion (the same length as `FUN_CRIT`) giving the weights to use for elements of `FUN_CRIT` [-]. See details

epsilon	(optional) [numeric (atomic or list)] small value to add to all observations and simulations when "log" or "inv" transformations are used [same unit as Obs]. See details
warnings	(optional) [boolean] boolean indicating if the warning messages are shown, default = TRUE

Details

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

— Period of calculation

Criteria can be calculated over discontinuous periods (i.e. only over winter periods, or when observed discharge is below a certain threshold). To do so, please indicate in Bool_Crit which indices must be used for calcullation. Discontinuous periods are allowed in the Bool_Crit argument.

— Transformations

Transformations are simple functions applied to the observed and simulated variables used in order to change their distribution. Transformations are often used in hydrology for modifying the weight put on errors made for high flows or low flows. The following transformations are available:

- "": no transformation is used (default case)
- "sqrt": squared root transformation
- "log": logarithmic transformation (see below regarding the specific case of KGE or KGE2)
- "inv": inverse transformation
- "sort": sort transformation (the simulated and observed variables are sorted from lowest to highest)
- "boxcox": Box-Cox transformation (see below for details)
- numeric: power transformation (see below for details)

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 the criterion value is dependent on the discharge unit. See Santos et al. (2018) for more details and alternative solutions (see the references list below).

In order to make sure that KGE and KGE2 remain dimensionless and are not impacted by zero values, the Box-Cox transformation (transfo = "boxcox") uses the formulation given in Equation 10 of Santos et al. (2018). Lambda is set to 0.25 accordingly.

The syntax of the power transformation allows a numeric or a string of characters. For example for a squared transformation, the following can be used: transfo = 2, transfo = "2" or transfo = "^2". Negative values are allowed. Fraction values are not allowed (e.g., "-1/2" must instead be written "-0.5").

— The epsilon value

The epsilon value is useful when "log" or "inv" transformations are used (to avoid calculation of the inverse or of the logarithm of zero). If an epsilon value is provided, then it is added to the observed and simulated variable time series at each time step and before the application of a transformation. The epsilon value has no effect when the "boxcox" transformation is used. The impact of this value and a recommendation about the epsilon value to use (usually one hundredth of average observation) are discussed in Pushpalatha et al. (2012) for NSE and in Santos et al. (2018) for KGE and KGE'.

— Single, multiple or composite criteria calculation

Users can set the following arguments as atomic or list: FUN_CRIT, Obs, VarObs, BoolCrit, transfo, Weights. If the list format is chosen, all the lists must have the same length.
Calculation of a single criterion (e.g. NSE computed on discharge) is prepared by providing to CreateInputsCrit arguments atomics only.
Calculation of multiple criteria (e.g. NSE computed on discharge and RMSE computed on discharge) is prepared by providing to CreateInputsCrit arguments lists except for Weights that must be set as NULL.
Calculation of a composite criterion (e.g. the average between NSE computed on discharge and NSE computed on log of discharge) is prepared by providing to CreateInputsCrit arguments lists including Weights.
ErrorCrit_RMSE cannot be used in a composite criterion since it is not a unitless value.

Value

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

$FUN_CRIT	[function] error criterion function (e.g. ErrorCrit_RMSE, ErrorCrit_NSE)
$Obs	[numeric] series of observed variable(s) ([mm/time step] for discharge or SWE, [-] for SCA)
$VarObs	[character] names of the observed variable(s)
$BoolCrit	[boolean] boolean giving the time steps considered in the computation
$transfo	[character] name of the transformation (e.g. "", "sqrt", "log", "inv", "sort", "boxcox" or a number for po
$epsilon	[numeric] small value to add to all observations and simulations when "log" or "inv" transformations are use
$Weights	[numeric] vector (same length as VarObs) giving the weights to use for elements of FUN_CRIT [-]

When Weights = NULL, CreateInputsCrit returns an object of class *Single* that is a list such as the one described above.
When Weights contains at least one NULL value and Obs contains a list of observations, CreateInputsCrit returns an object of class *Multi* that is a list of lists such as the one described above. The ErrorCrit function will then compute the different criteria prepared by CreateInputsCrit.
When Weights is a list of at least 2 numerical values, CreateInputsCrit returns an object of class *Compo* that is a list of lists such as the one described above. This object will be useful to compute composite criterion with the ErrorCrit function.
To calculate composite or multiple criteria, it is necessary to use the ErrorCrit function. The other

`ErrorCrit_*` functions (e.g. `ErrorCrit_RMSE`, `ErrorCrit_NSE`) can only use objects of class *Single* (and not *Multi* or *Compo*).

Author(s)

Olivier Delaigue, Laurent Coron, Guillaume Thirel

References

Pushpalatha, R., Perrin, C., Le Moine, N. and Andréassian, V. (2012). A review of efficiency criteria suitable for evaluating low-flow simulations. Journal of Hydrology, 420-421, 171-182, doi: 10.1016/j.jhydrol.2011.11.055.

Santos, L., Thirel, G. and Perrin, C. (2018). Technical note: Pitfalls in using log-transformed flows within the KGE criterion. Hydrol. Earth Syst. Sci., 22, 4583-4591, doi: 10.5194/hess-22-4583-2018.

See Also

`RunModel`, `CreateInputsModel`, `CreateRunOptions`, `CreateCalibOptions`, `ErrorCrit`

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)

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

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

## simulation
Param <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## single efficiency criterion: Nash-Sutcliffe Efficiency
InputsCritSingle <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE,
                                     InputsModel = InputsModel, RunOptions = RunOptions,
                                     Obs = list(BasinObs$Qmm[Ind_Run]),
                                     VarObs = "Q", transfo = "",
                                     Weights = NULL)
str(InputsCritSingle)
```

```
invisible(ErrorCrit(InputsCrit = InputsCritSingle, OutputsModel = OutputsModel))

## 2 efficiency criteria: RMSE and Nash-Sutcliffe Efficiency
InputsCritMulti <- CreateInputsCrit(FUN_CRIT = list(ErrorCrit_RMSE, ErrorCrit_NSE),
                                    InputsModel = InputsModel, RunOptions = RunOptions,
                                    Obs = list(BasinObs$Qmm[Ind_Run],
                                               BasinObs$Qmm[Ind_Run]),
                                    VarObs = list("Q", "Q"), transfo = list("", "sqrt"),
                                    Weights = NULL)
str(InputsCritMulti)
invisible(ErrorCrit(InputsCrit = InputsCritMulti, OutputsModel = OutputsModel))

## efficiency composite criterion: Nash-Sutcliffe Efficiency mixing
##                                 both raw and log-transformed flows
InputsCritCompo <- CreateInputsCrit(FUN_CRIT = list(ErrorCrit_NSE, ErrorCrit_NSE),
                                    InputsModel = InputsModel, RunOptions = RunOptions,
                                    Obs = list(BasinObs$Qmm[Ind_Run],
                                               BasinObs$Qmm[Ind_Run]),
                                    VarObs = list("Q", "Q"), transfo = list("", "log"),
                                    Weights = list(0.4, 0.6))
str(InputsCritCompo)
invisible(ErrorCrit(InputsCrit = InputsCritCompo, OutputsModel = OutputsModel))
```

CreateInputsCrit_Lavenne *Creation of the InputsCrit object for Lavenne Criterion*

Description

Creation of the InputsCrit object required to the ErrorCrit function. This function defines a composite criterion based on the formula proposed by Lavenne et al. (2019).

Usage

```
CreateInputsCrit_Lavenne(FUN_CRIT = ErrorCrit_KGE,
                         InputsModel,
                         RunOptions,
                         Obs,
                         VarObs = "Q",
                         AprParamR,
                         AprCrit = 1,
                         k = 0.15,
                         BoolCrit = NULL,
                         transfo = "sqrt",
                         epsilon = NULL)
```

Arguments

FUN_CRIT	[function] error criterion function (e.g. ErrorCrit_KGE, ErrorCrit_NSE). Default ErrorCrit_KGE
InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
Obs	[numeric (atomic or list)] series of observed variable ([mm/time step] for discharge or SWE, [-] for SCA)
VarObs	(optional) [character (atomic or list)] names of the observed variable ("Q" by default, or one of "SCA", "SWE")
AprParamR	[numeric] a priori parameter set from a donor catchment
AprCrit	(optional) [numeric] performance criterion of the donor catchment (1 by default)
k	(optional) [numeric] coefficient used for the weighted average between FUN_CRIT and the gap between the optimised parameter set and an a priori parameter set calculated with the function produced by CreateErrorCrit_GAPX
BoolCrit	(optional) [boolean] boolean (the same length as Obs) giving the time steps to consider in the computation (all time steps are considered by default. See details)
transfo	(optional) [character] name of the transformation applied to the variables (e.g. "", "sqrt", "log", "inv", "sort", "boxcox" or a numeric value for power transformation for FUN_CRIT. Default value is "sqrt". See details of CreateInputsCrit
epsilon	(optional) [numeric] small value to add to all observations and simulations for FUN_CRIT when "log" or "inv" transformations are used [same unit as Obs]. See details of CreateInputsCrit

Details

The parameters FUN_CRIT, Obs, VarObs, BoolCrit, transfo, and epsilon must be used as they would be used for CreateInputsCrit in the case of a single criterion.

ErrorCrit_RMSE cannot be used in a composite criterion since it is not a unitless value.

CreateInputsCrit_Lavenne creates a composite criterion in respect with Equations 1 and 2 of de Lavenne et al. (2019).

Value

[list] object of class InputsCrit containing the data required to evaluate the model outputs (see CreateInputsCrit for more details).

CreateInputsCrit_Lavenne returns an object of class *Compo*.

Items Weights of the criteria are respectively equal to k and k * max(0, AprCrit).

To calculate the Lavenne criterion, it is necessary to use the ErrorCrit function as for any other composite criterion.

Author(s)

David Dorchies

References

de Lavenne, A., Andréassian, V., Thirel, G., Ramos, M.-H. and Perrin, C. (2019). A Regularization Approach to Improve the Sequential Calibration of a Semidistributed Hydrological Model. Water Resources Research 55, 8821–8839. doi:10.1029/2018WR024266

See Also

RunModel, CreateInputsModel, CreateRunOptions, CreateCalibOptions, ErrorCrit

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)

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

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

## simulation
Param <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel,
                              RunOptions = RunOptions, Param = Param)

## The "a priori" parameters for the Lavenne formula
AprParamR <- c(X1 = 157, X2 = 0.8, X3 = 100, X4 = 1.5)

## Single efficiency criterion: GAPX with a priori parameters
IC_DL <- CreateInputsCrit_Lavenne(InputsModel = InputsModel,
                                  RunOptions = RunOptions,
                                  Obs = BasinObs$Qmm[Ind_Run],
                                  AprParamR = AprParamR)
str(ErrorCrit(InputsCrit = IC_DL, 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,
                  Qupstream = NULL, LengthHydro = NULL, BasinAreas = NULL,
                  QupstrUnit = "mm", verbose = TRUE)

## S3 method for class 'InputsModel'
x[i]
```

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
Qupstream	(optional) [numerical matrix] time series of upstream flows (catchment average), its unit is defined by the QupstrUnit parameter, required to create the SD model inputs. See details
LengthHydro	(optional) [numeric] real giving the distance between the downstream outlet and each upstream inlet of the sub-catchment [km], required to create the SD model inputs . See details

BasinAreas	(optional) [numeric] real giving the area of each upstream sub-catchment [km2] and the area of the downstream sub-catchment in the last item, required to create the SD model inputs . See details
QupstrUnit	(optional) [character] unit of the flow in the argument Qupstream, available units are: "mm" for mm/time-step (default), "m3" for m3/time-step, "m3/s" and "l/s". See details
x	[InputsModel] object of class InputsModel
i	[integer] of the indices to subset a time series or [character] names of the elements to extract

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.

Users wanting to use a semi-distributed (SD) model should provide valid Qupstream, LengthHydro, and BasinAreas arguments. Each upstream sub-catchment is described by an upstream flow time series (one column in Qupstream matrix), a distance between the downstream outlet and the upstream inlet (one item in LengthHydro) and an area (one item in BasinAreas). The order of the columns or of the items should be consistent for all these parameters. BasinAreas should contain one extra information (stored in the last item) which is the area of the downstream sub-catchment. Upstream flows that are not related to a sub-catchment such as release or withdraw spots are identified by an area equal to NA, and if unit="mm" the upstream flow must be expressed in m3/time step instead of mm/time step which is not possible in absence of a related area. Please note that the use of SD model requires to use the RunModel function instead of RunModel_GR4J or the other RunModel_* functions.

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, GR5H, 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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 734.568, X2 = -0.840, X3 = 109.809, X4 = 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, Obs = 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, Imax = NULL,
                 Outputs_Cal = NULL, Outputs_Sim = "all",
                 MeanAnSolidPrecip = NULL, IsHyst = FALSE,
                 warnings = TRUE, 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 [-]. See details
IndPeriod_Run	[numeric] index of period to be used for the model run [-]. See details
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 (4 values; use NA when not relevant for a given model) [- and/or mm]. See details
Imax	(optional) [numeric] an atomic vector of the maximum capacity of the GR5H interception store [mm]; see RunModel_GR5H
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"
MeanAnSolidPrecip	(optional) [numeric] vector giving the annual mean of average solid precipitation for each layer (computed from InputsModel if not defined) [mm/y]
IsHyst	[boolean] boolean indicating if the hysteresis version of CemaNeige is used. See 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

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

— IndPeriod_WarmUp and IndPeriod_Run

Since the hydrological models included in airGR are continuous models, meaning that internal states of the models are propagated to the next time step, IndPeriod_WarmUp and IndPeriod_Run must be continuous periods, represented by continuous indices values; no gaps are allowed. To calculate

criteria or to calibrate a model over discontinuous periods, please see the Bool_Crit argument of the CreateInputsCrit function.

— 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 levels which are respectively initialised at 30 % and 50 % of their capacity. In case GR5H is used with an interception store, the intercetion store level is initialised by default with 0 mm. In case GR6J is used, the exponential store level 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, 1000:1365)).
  - 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, NA, NA) 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, NA)` should be used to obtain initial fillings of 30 % and 50 % for the production and routing stores levels and 0 mm for the exponential store level, respectively. For GR5H with an interception store, `IniResLevels` = `c(0.3, 0.5, NA, 0.4)` should be used to obtain initial fillings of 30 %, 50 % and 40 % for the production, routing and interception stores levels, 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).

— CemaNeige version

If `IsHyst` = `FALSE`, the original CemaNeige version from Valéry et al. (2014) is used.
If `IsHyst` = `TRUE`, the CemaNeige version from Riboust et al. (2019) is used. Compared to the original version, this version of CemaNeige needs two more parameters and it includes a representation of the hysteretic relationship between the Snow Cover Area (SCA) and the Snow Water Equivalent (SWE) in the catchment. The hysteresis included in airGR is the Modified Linear hysteresis (LH*); it is represented on panel b) of Fig. 3 in Riboust et al. (2019). Riboust et al. (2019) advise to use the LH* version of CemaNeige with parameters calibrated using an objective function combining 75 % of KGE calculated on discharge simulated from a rainfall-runoff model compared to observed discharge and 5 % of KGE calculated on SCA on 5 CemaNeige elevation bands compared to satellite (e.g. MODIS) SCA (see Eq. (18), Table 3 and Fig. 6). Riboust et al. (2019)'s tests were realized with GR4J as the chosen rainfall-runoff model.

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
Imax	[numeric] vector giving the maximal capacity of the GR5H interception store
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`, `Imax`

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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 734.568, X2 = -0.840, X3 = 109.809, X4 = 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,
                                Obs = 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/time step]
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/time step]
$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, Olivier Delaigue, Pierre Brigode, Guillaume Thirel

References

Turcotte, R., Fortin, L.-G., Fortin, V., Fortin, J.-P. and Villeneuve, J.-P. (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 - Cemagref Antony, 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, 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`

Value

If `InputsCrit` is of class *Single*:

[list] containing the `ErrorCrit_*` functions outputs, see `ErrorCrit_RMSE` or `ErrorCrit_NSE` for details

If `InputsCrit` is of class *Multi*:

[list] of list containing the `ErrorCrit_*` functions outputs, see `ErrorCrit_RMSE` or `ErrorCrit_NSE` for details

If `InputsCrit` is of class *Compo*:

$CritValue	[numeric] value of the composite criterion
$CritName	[character] name of the composite criterion
$CritBestValue	[numeric] theoretical best criterion value
$Multiplier	[numeric] integer indicating whether the criterion is indeed an error (+1) or an efficiency (-1
$CritCompo$MultiCritValues	[numeric] values of the sub-criteria
$CritCompo$MultiCritNames	[numeric] names of the sub-criteria
$CritCompo$MultiCritWeights	[character] weighted values of the sub-criteria
$MultiCrit	[list] of list containing the `ErrorCrit_*` functions outputs, see `ErrorCrit_NSE` or `ErrorCri`

Author(s)

Olivier Delaigue

See Also

`CreateInputsCrit`, `ErrorCrit_RMSE`, `ErrorCrit_NSE`, `ErrorCrit_KGE`, `ErrorCrit_KGE2`

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)

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

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

## simulation
Param <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel,
                              RunOptions = RunOptions, Param = Param)

## single efficiency criterion: Nash-Sutcliffe Efficiency
InputsCritSingle <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE,
                                     InputsModel = InputsModel, RunOptions = RunOptions,
                                     Obs = list(BasinObs$Qmm[Ind_Run]),
```

```
                                    VarObs = "Q", transfo = "",
                                    Weights = NULL)
str(ErrorCrit(InputsCrit = InputsCritSingle, OutputsModel = OutputsModel))

## 2 efficiency critera:  RMSE and the Nash-Sutcliffe Efficiency
InputsCritMulti <- CreateInputsCrit(FUN_CRIT = list(ErrorCrit_RMSE, ErrorCrit_NSE),
                                    InputsModel = InputsModel, RunOptions = RunOptions,
                                    Obs = list(BasinObs$Qmm[Ind_Run],
                                               BasinObs$Qmm[Ind_Run]),
                                    VarObs = list("Q", "Q"), transfo = list("", "sqrt"),
                                    Weights = NULL)
str(ErrorCrit(InputsCrit = InputsCritMulti, OutputsModel = OutputsModel))

## efficiency composite criterion: Nash-Sutcliffe Efficiency mixing
##                                 both raw and log-transformed flows
InputsCritCompo <- CreateInputsCrit(FUN_CRIT = list(ErrorCrit_NSE, ErrorCrit_NSE),
                                    InputsModel = InputsModel, RunOptions = RunOptions,
                                    Obs = list(BasinObs$Qmm[Ind_Run],
                                               BasinObs$Qmm[Ind_Run]),
                                    VarObs = list("Q", "Q"), transfo = list("", "log"),
                                    Weights = list(0.4, 0.6))
str(ErrorCrit(InputsCrit = InputsCritCompo, 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 $CritValue \times Multiplier$ 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, Olivier Delaigue

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

```
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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 734.568, X2 = -0.840, X3 = 109.809, X4 = 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions,
                         Param = Param, FUN = RunModel_GR4J)

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

## efficiency criterion: Kling-Gupta Efficiency on square-root-transformed flows
transfo <- "sqrt"
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run],
                               transfo = transfo)
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency above a threshold (quant. 75 %)
BoolCrit <- BasinObs$Qmm[Ind_Run] >= quantile(BasinObs$Qmm[Ind_Run], 0.75, na.rm = TRUE)
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run],
                               BoolCrit = BoolCrit)
OutputsCrit <- ErrorCrit_KGE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

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 $CritValue \times Multiplier$ 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, Olivier Delaigue

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

```
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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 734.568, X2 = -0.840, X3 = 109.809, X4 = 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions,
                         Param = Param, FUN = RunModel_GR4J)

## efficiency criterion: Kling-Gupta Efficiency
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE2, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_KGE2(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency on square-root-transformed flows
transfo <- "sqrt"
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE2, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run],
                               transfo = transfo)
OutputsCrit <- ErrorCrit_KGE2(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency above a threshold (quant. 75 %)
BoolCrit <- BasinObs$Qmm[Ind_Run] >= quantile(BasinObs$Qmm[Ind_Run], 0.75, na.rm = TRUE)
```

```
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_KGE2, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run],
                               BoolCrit = BoolCrit)
OutputsCrit <- ErrorCrit_KGE2(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`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 $CritValue \times Multiplier$ 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, Olivier Delaigue

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:10.1016/00221694(70)902556.

See Also

ErrorCrit_RMSE, ErrorCrit_KGE, ErrorCrit_KGE2

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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 734.568, X2 = -0.840, X3 = 109.809, X4 = 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, Obs = 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, Obs = BasinObs$Qmm[Ind_Run],
                               transfo = transfo)
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency above a threshold (quant. 75 %)
BoolCrit <- BasinObs$Qmm[Ind_Run] >= quantile(BasinObs$Qmm[Ind_Run], 0.75, na.rm = TRUE)
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run],
                               BoolCrit = BoolCrit)
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`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 $CritValue \times Multiplier$ 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, Olivier Delaigue

See Also

`ErrorCrit_NSE`, `ErrorCrit_KGE`, `ErrorCrit_KGE2`

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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 734.568, X2 = -0.840, X3 = 109.809, X4 = 1.971)
OutputsModel <- RunModel(InputsModel = InputsModel, RunOptions = RunOptions,
                         Param = Param, FUN = RunModel_GR4J)

## efficiency criterion: root-mean-square error
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_RMSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_RMSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: root-mean-square error on log-transformed flows
transfo <- "log"
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_RMSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run],
                               transfo = transfo)
OutputsCrit <- ErrorCrit_RMSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## efficiency criterion: Kling-Gupta Efficiency above a threshold (quant. 75 %)
BoolCrit <- BasinObs$Qmm[Ind_Run] >= quantile(BasinObs$Qmm[Ind_Run], 0.75, na.rm = TRUE)
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_RMSE, InputsModel = InputsModel,
                               RunOptions = RunOptions, Obs = BasinObs$Qmm[Ind_Run],
                               BoolCrit = BoolCrit)
OutputsCrit <- ErrorCrit_RMSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

Imax *Computation of the maximum capacity of the GR5H interception store*

Description

Function which calculates the maximal capacity of the GR5H interception store. This function compares the interception evapotranspiration from the GR5H interception store for different maximal capacity values with the interception evapotranspiration classically used in the daily GR models

(e.g. GR4J). Among all the TestedValues, the value that gives the closest interception evapotranspiration flux over the whole period is kept.

Usage

```
Imax(InputsModel, IndPeriod_Run,
     TestedValues = seq(from = 0.1, to = 3, by = 0.1))
```

Arguments

InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
IndPeriod_Run	[numeric] index of period to be used for the model run [-]
TestedValues	[numeric] vector of tested Imax values [mm]

Value

Optimal Imax value [mm].

Author(s)

Guillaume Thirel, Olivier Delaigue

References

Ficchi, A. (2017). An adaptive hydrological model for multiple time-steps: Diagnostics and improvements based on fluxes consistency. PhD thesis, UPMC - Irstea Antony, Paris, France.

Ficchi, A., Perrin, C. and Andréassian, V. (2019). Hydrological modelling at multiple sub-daily time steps: model improvement via flux-matching. Journal of Hydrology, 575, 1308-1327, doi:10.1016/j.jhydrol.2019.05.084.

See Also

RunModel_GR5H, CreateInputsModel, CreateRunOptions.

Examples

```
library(airGR)

## loading catchment data
data(L0123003)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR5H, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%Y-%m-%d %H")=="2006-01-01 00"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d %H")=="2006-12-31 23"))

## Imax computation
```

```
Imax <- Imax(InputsModel = InputsModel, IndPeriod_Run = Ind_Run,
             TestedValues = seq(from = 0, to = 3, by = 0.2))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR5H, Imax = Imax,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(X1 = 706.912, X2 = -0.163, X3 = 188.880, X4 = 2.575, X5 = 0.104)
OutputsModel <- RunModel_GR5H(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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`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: $X4 = X4u/5.995 \times S^{0.3}$ (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., Bourgin, F., Oudin, L., Mathevet, T., Perrin, C., Lerat, J., Coron, L. and Berthet, L. (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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1990-02-28"))

## 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, Obs = 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$DatesR, format = "%Y-%m-%d")=="1990-03-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

## 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, Obs = BasinObs$Qmm[Ind_Val])
OutputsCrit_Val <- ErrorCrit_NSE(InputsCrit = InputsCrit_Val, OutputsModel = OutputsModel_Val)
```

`PE_Oudin` *Computation of series of potential evapotranspiration at the daily or hourly time steps with Oudin's formula*

Description

Function which computes PE using the formula from Oudin et al. (2005). PE can be computed at the daily time step from hourly or daily temperature and at the hourly time step with hourly or daily temperature through a disaggregation of daily PE . See details.

Usage

```
PE_Oudin(JD, Temp, Lat, LatUnit =  c("rad", "deg"),
         TimeStepIn = "daily", TimeStepOut = "daily",
         RunFortran = FALSE)
```

Arguments

`JD`	[numeric] time series of Julian day of the year [-]; see details
`Temp`	[numeric] time series of daily (or hourly) mean air temperature [°C]
`Lat`	[numeric] latitude of measurement for the temperature series [radians or degrees]. Atomic vector, except if `RunFortran = TRUE`, it can be a vector of the same length as `Temp`
`LatUnit`	[character] latitude unit (default = `"rad"` or `"deg"`)
`TimeStepIn`	[character] time step of inputs (e.g. `"daily"` or `"hourly"`, default = `"daily"`)
`TimeStepOut`	[character] time step of outputs (e.g. `"daily"` or `"hourly"`, default = `"daily"`)
`RunFortran`	[boolean] to run the code in the Fortran mode or in the R mode (default)

Details

To calculate basin-wide Oudin potential evapotranspiration, it is advised, when possible, to use either station temperature or gridded temperature data to calculate PE and then average these PE values at the basin scale.

In the `JD` argument, the Julian day of the year of the 1st of January is equal to 1 and the 31st of December to 365 (366 in leap years). If the Julian day of the year is computed on an object of the `POSIXlt` class, the user has to add 1 to the returned value (e.g. `as.POSIXlt("2016-12-31")$yday + 1`).

When hourly temperature is provided, all the values of the same day have to be set to the same Julian day of the year (e.g. `as.POSIXlt("2016-12-31 00:00:00")$yday + 1` and `as.POSIXlt("2016-12-31 00:01:00")$yday + 1`). Each single day must be provided 24 identical Julian day values (one for each hour).

Four cases are possible:

- `TimeStepIn = "daily"` and `TimeStepOut = "daily"`: this is the classical application of the Oudin et al. (2005) formula
- `TimeStepIn = "daily"` and `TimeStepOut = "hourly"`: the daily temperature is used inside the `PE_Oudin` function to calculate daily PE, which is then disaggregated at the hourly time step with use of a sinusoidal function (see Lobligeois, 2014, p. 78)
- `TimeStepIn = "hourly"` and `TimeStepOut = "daily"`: the hourly temperature is aggregated at the daily time step and the daily PE is calculated normally within `PE_Oudin`
- `TimeStepIn = "hourly"` and `TimeStepOut = "hourly"`: the hourly temperature is aggregated at the daily time step, the daily PE is then calculated normally within `PE_Oudin`, which is finally disaggregated at the hourly time step with use of a sinusoidal function (see Lobligeois, 2014, p. 78)

The use of the `PEdaily_Oudin` corresponds to the first case of the use of `PE_Oudin`.

Value

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

Author(s)

Laurent Coron, Ludovic Oudin, Olivier Delaigue, Guillaume Thirel, François Bourgin

References

Oudin, L., Hervieu, F., Michel, C., Perrin, C., Andréassian, V., Anctil, F. and Loumagne, C. (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.

Lobligeois, F. (2014). Mieux connaitre la distribution spatiale des pluies améliore-t-il la modélisation des crues ? Diagnostic sur 181 bassins versants français. PhD thesis (in French), AgroParisTech - Irstea Antony, Paris, France.

Examples

```
library(airGR)
data(L0123001)
PotEvap <- PE_Oudin(JD = as.POSIXlt(BasinObs$DatesR)$yday + 1,
                    Temp = BasinObs$T,
                    Lat = 0.8, LatUnit = "rad")
```

`plot` *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 = "synth", log_scale = FALSE,
     cex.axis = 1, cex.lab = 0.9, cex.leg = 0.9, lwd = 1,
     AxisTS = function(x) axis.POSIXct(side = 1, x = x$DatesR, ...),
     LayoutMat = NULL,
     LayoutWidths = rep.int(1, ncol(LayoutMat)),
     LayoutHeights = rep.int(1, nrow(LayoutMat)),
     verbose = TRUE, ...)
```

Arguments

`x`	[object of class *OutputsModel*] list of model outputs (which must at least include DatesR, Precip and Qsim) [POSIXlt, mm/time step, mm/time step]
`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", "Error", "Regime", "CumFreq", "CorQQ")), default = "synth", see details below
`log_scale`	(optional) [boolean] indicating if the flow and the error time series axis and the flow error time series axis are 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)
AxisTS	(optional) [function] to manage x-axis representing calendar dates and times on time series plots (see axis and axis.POSIXct)
LayoutMat	(optional) [numeric] a matrix object specifying the location of the next N figures on the output device. Each value in the matrix must be 0 or a positive integer. If N is the largest positive integer in the matrix, then the integers $1, \ldots, N-1$ must also appear at least once in the matrix (see layout)
LayoutWidths	(optional) [numeric] a vector of values for the widths of columns on the device (see layout)
LayoutHeights	(optional) [numeric] a vector of values for the heights of rows on the device (see layout)
verbose	(optional) [boolean] indicating if the function is run in verbose mode or not, default = TRUE
...	(optional) other parameters to be passed through to plotting functions

Details

Different types of independent graphs are available (depending on the model, but always drawn in this order):

- "Precip": time series of total precipitation
- "PotEvap": time series of potential evapotranspiration
- "ActuEvap": time series of simulated actual evapotranspiration (overlaid to "PotEvap" if already drawn)
- "Temp": time series of temperature (plotted only if CemaNeige is used)
- "SnowPack": time series of snow water equivalent (plotted only if CemaNeige is used)
- "Flows": time series of simulated flows (and observed flows if provided)
- "Error": time series of simulated flows minus observed flows (and observed flows if provided)
- "Regime": centred 30-day rolling mean applied on interannual average of daily simulated flows (and observed flows if provided)
- "CorQQ": correlation plot between simulated and observed flows (only if observed flows provided)
- "CumFreq": cumulative frequency plot for simulated flows (and observed flows if provided)

Different dashboards of results including various graphs are available:

- "perf": corresponds to "Error", "Regime", "CumFreq" and "CorQQ"
- "ts": corresponds to "Precip", "PotEvap", "Temp", "SnowPack" and "Flows"
- "synth": corresponds to "Precip", "Temp", "SnowPack", "Flows", "Regime", "CumFreq" and "CorQQ"

- "all": corresponds to "Precip", "PotEvap", "ActuEvap", "Temp", "SnowPack", "Flows", "Error", "Regime", "CumFreq" and "CorQQ"

If several dashboards are selected, or if an independent graph is called with a dashboard, the graphical device will include all the requested graphs without redundancy.

Value

Screen plot window.

Author(s)

Laurent Coron, Olivier Delaigue, Guillaume Thirel

Examples

```
### see examples of RunModel_GR4J or RunModel_CemaNeigeGR4J functions
### to understand how the example datasets have been prepared


## loading examples dataset for GR4J and GR4J + CemaNeige
data(exampleSimPlot)


### Qobs and outputs from GR4J and GR4J + CemaNeige models
str(simGR4J, max.level = 1)
str(simCNGR4J, max.level = 1)


### default dashboard (which = "synth")

## GR models whithout CemaNeige
plot(simGR4J$OutputsModel, Qobs = simGR4J$Qobs)

## GR models whith CemaNeige ("Temp" and "SnowPack" added)
plot(simCNGR4J$OutputsModel, Qobs = simCNGR4J$Qobs)


### "Error" and "CorQQ" plots cannot be display whithout Qobs
plot(simGR4J$OutputsModel, which = "all", Qobs = simGR4J$Qobs)
plot(simGR4J$OutputsModel, which = "all", Qobs = NULL)


### complex plot arrangements
plot(simGR4J$OutputsModel, Qobs = simGR4J$Qobs,
     which = c("Flows", "Regime", "CumFreq", "CorQQ"),
     LayoutMat = matrix(c(1, 2, 3, 1, 4, 4), ncol = 2),
     LayoutWidths  = c(1.5, 1),
     LayoutHeights = c(0.5, 1, 1))


### customize x-axis on time series plot
FunAxisTS <- function(x) {
```

```
  axis.POSIXct(side = 1, x = x$DatesR,
  at = pretty(x$DatesR, n = 10),
  format = "%Y-%m-%d")
}
plot(simGR4J$OutputsModel, Qobs = simGR4J$Qobs, AxisTS = FunAxisTS)


### add a main title

## the whole list of settable par's
opar <- par(no.readonly = TRUE)

## define outer margins and a title inside it
par(oma = c(0, 0, 3, 0))
plot(simGR4J$OutputsModel, Qobs = simGR4J$Qobs)
title(main = "GR4J outputs", outer = TRUE, line = 1.2, cex.main = 1.4)

## reset original par
par(opar)
```

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 (See details for SD lag model)
FUN_MOD	[function] hydrological model function (e.g. RunModel_GR4J, RunModel_CemaNeigeGR4J)
...	(optional) arguments to pass to FUN_MOD

Details

If InputsModel parameter has been created for using a semi-distributed (SD) lag model (See CreateInputsModel), the first item of Param parameter should contain a constant lag parameter expressed as a velocity in m/s, parameters for the hydrological model are then shift one position to the right.

Value

[list] see RunModel_GR4J or RunModel_CemaNeigeGR4J for details.

If InputsModel parameter has been created for using a semi-distributed (SD) lag model (See CreateInputsModel), the list value contains an extra item named QsimDown which is a numeric series of simulated discharge [mm/time step] related to the run-off contribution of the downstream sub-catchment.

Author(s)

Laurent Coron, Olivier Delaigue

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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 734.568, X2 = -0.840, X3 = 109.809, X4 = 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, Obs = 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 snow module at the daily or hourly time step.

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 (or 4 parameters if `IsHyst = TRUE`, see `CreateRunOptions` for details)

CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-day melt coefficient [mm/°C/time step]
CemaNeige X3	(optional) accumulation threshold [mm] (needed if `IsHyst = TRUE`)
CemaNeige X4	(optional) percentage (between 0 and 1) of annual snowfall defining the melt threshold [-] (needed if `IsHyst`

Details

The choice of the CemaNeige version (i.e. with or without hysteresis) is explained in `CreateRunOptions`.
For further details on the model, see the references section.
For further details on the argument structures and initialisation options, see `CreateRunOptions`.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$CemaNeigeLayers	[list] CemaNeige outputs (1 element per layer)
$CemaNeigeLayers[[iLayer]]$Pliq	[numeric] series of liquid precip. [mm/time step]
$CemaNeigeLayers[[iLayer]]$Psol	[numeric] series of solid precip. [mm/time step]
$CemaNeigeLayers[[iLayer]]$SnowPack	[numeric] series of snow pack (snow water equivalent) [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/time step]
$CemaNeigeLayers[[iLayer]]$Melt	[numeric] series of actual snow melt [mm/time step]
$CemaNeigeLayers[[iLayer]]$PliqAndMelt	[numeric] series of liquid precip. + actual snow melt [mm/time step]
$CemaNeigeLayers[[iLayer]]$Temp	[numeric] series of air temperature [°C]

$CemaNeigeLayers[[iLayer]]$Gthreshold	[numeric] series of melt threshold [mm]
$CemaNeigeLayers[[iLayer]]$Glocalmax	[numeric] series of local melt threshold for hysteresis [mm]
$StateEnd	[numeric] states at the end of the run: CemaNeige states [mm & °C]. See Crea

Refer to the provided references or to the package source code for further details on these model outputs.

Author(s)

Laurent Coron, Audrey Valéry, Vazken Andréassian, Olivier Delaigue, Guillaume Thirel

References

Riboust, P., Thirel, G., Le Moine, N. and Ribstein, P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.2478/johh20180004.

Valéry, A., Andréassian, V. and Perrin, C. (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, 517(0), 1166-1175, doi:10.1016/j.jhydrol.2014.04.059.

Valéry, A., Andréassian, V. and Perrin, C. (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:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeigeGR4J, CreateInputsModel, CreateRunOptions, CreateIniStates, CreateCalibOptions.

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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))
```

```
## --- original version of CemaNeige

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

## simulation
Param <- c(CNX1 = 0.962, CNX2 = 2.249)
OutputsModel <- RunModel_CemaNeige(InputsModel = InputsModel,
                                   RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel)


## --- version of CemaNeige with the Linear Hysteresis

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

## simulation
Param <- c(CNX1 = 0.962, CNX2 = 2.249, CNX3 = 100, CNX4 = 0.4)
OutputsModel <- RunModel_CemaNeige(InputsModel = InputsModel,
                                   RunOptions = RunOptions, Param = Param)

## results preview
plot(OutputsModel)
```

`RunModel_CemaNeigeGR4H` *Run with the CemaNeigeGR4H hydrological model*

Description

Function which performs a single run for the CemaNeige-GR4H hourly lumped model over the test period.

Usage

```
RunModel_CemaNeigeGR4H(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 (or 8 parameters if `IsHyst = TRUE`, see `CreateRunOptions` for details)

GR4H X1	production store capacity [mm]
GR4H X2	intercatchment exchange coefficient [mm/h]
GR4H X3	routing store capacity [mm]
GR4H X4	unit hydrograph time constant [h]
CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-hour melt coefficient [mm/°C/h]
CemaNeige X3	(optional) accumulation threshold [mm] (needed if `IsHyst = TRUE`)
CemaNeige X4	(optional) percentage (between 0 and 1) of annual snowfall defining the melt threshold [-] (needed if `IsHyst`

Details

It is advised to run the GR5H model with an interception store (see Ficchi et al. (2019)) as it improves the consistency of the model fluxes and provides better performance. To do so, the `Imax` functions allows to estimates the maximal capacity of the interception store, which can then be given to `CreateRunOptions`.

The choice of the CemaNeige version is explained in `CreateRunOptions`.
For further details on the model, see the references section.
For further details on the argument structures and initialisation options, see `CreateRunOptions`.

See `RunModel_GR4J` to look at the diagram of the hydrological model.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/h]
$Precip	[numeric] series of input total precipitation (P) [mm/h]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/h]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [mm/h]
$AE	[numeric] series of actual evapotranspiration [mm/h]
$Perc	[numeric] series of percolation (Perc) [mm/h]
$PR	[numeric] series of Pr=Pn-Ps+Perc (Pr) [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 (R1) [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/h]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/h]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/h]
$AExch	[numeric] series of actual exchange between catchments (AExch1+AExch2) [r
$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 (Q) [mm/h]
$CemaNeigeLayers	[list] CemaNeige outputs (1 element per layer)
$CemaNeigeLayers[[iLayer]]$Pliq	[numeric] series of liquid precip. [mm/h]

$CemaNeigeLayers[[iLayer]]$Psol	[numeric] series of solid precip. [mm/h]
$CemaNeigeLayers[[iLayer]]$SnowPack	[numeric] series of snow pack (snow water equivalent)[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/h]
$CemaNeigeLayers[[iLayer]]$Melt	[numeric] series of actual snow melt [mm/h]
$CemaNeigeLayers[[iLayer]]$PliqAndMelt	[numeric] series of liquid precip. + actual snow melt [mm/h]
$CemaNeigeLayers[[iLayer]]$Temp	[numeric] series of air temperature [°C]
$CemaNeigeLayers[[iLayer]]$Gthreshold	[numeric] series of melt threshold [mm]
$CemaNeigeLayers[[iLayer]]$Glocalmax	[numeric] series of local melt threshold for hysteresis [mm]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/h]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run: store & unit hydrographs levels [mm], C

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, Thibault Mathevet, Audrey Valéry, Vazken Andréassian, Olivier Delaigue, Guillaume Thirel

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 (in French), UPMC - Cemagref Antony, Paris, France.

Riboust, P., Thirel, G., Le Moine, N. and Ribstein, P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.2478/johh20180004.

Valéry, A., Andréassian, V. and Perrin, C. (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, 517(0), 1166-1175, doi:10.1016/j.jhydrol.2014.04.059.

Valéry, A., Andréassian, V. and Perrin, C. (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:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeige, RunModel_CemaNeigeGR4J, RunModel_CemaNeigeGR5J, RunModel_CemaNeigeGR6J, RunModel_GR4H, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
## Not run:
library(airGR)

## loading catchment data
data(U2345030)


## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_CemaNeigeGR4H, DatesR = BasinObs$DatesR,
                          Precip = BasinObs$P, PotEvap = BasinObs$E, TempMean = BasinObs$T,
                                  ZInputs = BasinInfo$ZInputs,
                                  HypsoData = BasinInfo$HypsoData, NLayers = 5)

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


## --- original version of CemaNeige

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR4H, InputsModel = InputsModel,
                               IndPeriod_Run = Ind_Run)

## simulation
Param <- c(X1 = 149.905, X2 = -0.487, X3 = 391.506, X4 = 9.620,
           CNX1 = 0.520, CNX2 = 0.133)
OutputsModel <- RunModel_CemaNeigeGR4H(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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## End(Not run)
```

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 (or 8 parameters if `IsHyst = TRUE`, see `CreateRunOptions` for details)

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]
CemaNeige X3	(optional) accumulation threshold [mm] (needed if `IsHyst = TRUE`)
CemaNeige X4	(optional) percentage (between 0 and 1) of annual snowfall defining the melt threshold [-] (needed if `IsHyst`

Details

The choice of the CemaNeige version is explained in `CreateRunOptions`.
For further details on the model, see the references section.
For further details on the argument structures and initialisation options, see `CreateRunOptions`.

See `RunModel_GR4J` to look at the diagram of the hydrological model.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d] (E)
$Precip	[numeric] series of input total precipitation (P) [mm/d]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [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 (Pr) [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 (R1) [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 (Q) [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 (snow water equivalent) [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]
$CemaNeigeLayers[[iLayer]]$Gthreshold	[numeric] series of melt threshold [mm]
$CemaNeigeLayers[[iLayer]]$Glocalmax	[numeric] series of local melt threshold for hysteresis [mm]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/d]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run: store & unit hydrographs levels [mm], C

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, Audrey Valéry, Vazken Andréassian, Olivier Delaigue, Guillaume Thirel

References

Perrin, C., Michel, C. and Andréassian, V. (2003). Improvement of a parsimonious model for streamflow simulation. Journal of Hydrology, 279(1-4), 275-289, doi:10.1016/S00221694(03)00225-7.

Riboust, P., Thirel, G., Le Moine, N. and Ribstein, P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.2478/johh20180004.

Valéry, A., Andréassian, V. and Perrin, C. (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, 517(0), 1166-1175, doi:10.1016/j.jhydrol.2014.04.059.

Valéry, A., Andréassian, V. and Perrin, C. (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:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeige, 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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))


## --- original version of CemaNeige

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR4J, InputsModel = InputsModel,
                               IndPeriod_Run = Ind_Run)

## simulation
Param <- c(X1 = 408.774, X2 = 2.646, X3 = 131.264, X4 = 1.174,
           CNX1 = 0.962, CNX2 = 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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)


## --- version of CemaNeige with the Linear Hysteresis

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

```
                                IndPeriod_Run = Ind_Run, IsHyst = TRUE)

## simulation
Param <- c(X1 = 408.774, X2 = 2.646, X3 = 131.264, X4 = 1.174,
           CNX1 = 0.962, CNX2 = 2.249, CNX3 = 100, CNX4 = 0.4)
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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

`RunModel_CemaNeigeGR5H` *Run with the CemaNeigeGR5H hydrological model*

Description

Function which performs a single run for the CemaNeige-GR5H hourly lumped model over the test period.

Usage

```
RunModel_CemaNeigeGR5H(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 (or 9 parameters if `IsHyst = TRUE`, see `CreateRunOptions` for details)

GR5H X1	production store capacity [mm]
GR5H X2	intercatchment exchange coefficient [mm/h]
GR5H X3	routing store capacity [mm]
GR5H X4	unit hydrograph time constant [h]
GR5H X5	intercatchment exchange threshold [-]
CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-hour melt coefficient [mm/°C/h]
CemaNeige X3	(optional) accumulation threshold [mm] (needed if `IsHyst = TRUE`)
CemaNeige X4	(optional) percentage (between 0 and 1) of annual snowfall defining the melt threshold [-] (needed if `IsHyst`

Details

The choice of the CemaNeige version is explained in `CreateRunOptions`.
For further details on the model, see the references section.
For further details on the argument structures and initialisation options, see `CreateRunOptions`.

See `RunModel_GR5H` to look at the diagram of the hydrological model or `RunModel_GR5J` when no interception store is used.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/h]
$Precip	[numeric] series of input total precipitation (P) [mm/h]
$Interc	[numeric] series of interception store level (I) [mm]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/h]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [mm/h]
$AE	[numeric] series of actual evapotranspiration (Ei+Es) [mm/h]
$EI	[numeric] series of evapotranspiration from rainfall neutralisation or intercepti
$ES	[numeric] series of evapotranspiration from production store (Es) [mm/h]
$Perc	[numeric] series of percolation (Perc) [mm/h]
$PR	[numeric] series of Pr=Pn-Ps+Perc (Pr) [mm/h]
$Q9	[numeric] series of UH outflow going into branch 9 (Q9) [mm/h]
$Q1	[numeric] series of UH outflow going into branch 1 (Q1) [mm/h]
$Rout	[numeric] series of routing store level (R1) [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/h]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/h]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/h]
$AExch	[numeric] series of actual exchange between catchments (AExch1+AExch2) [
$QR	[numeric] series of routing store outflow (Qr) [mm/h]
$QD	[numeric] series of direct flow from UH after exchange (Qd) [mm/h]
$Qsim	[numeric] series of simulated discharge (Q) [mm/h]
$CemaNeigeLayers	[list] CemaNeige outputs (1 element per layer)
$CemaNeigeLayers[[iLayer]]$Pliq	[numeric] series of liquid precip. [mm/h]
$CemaNeigeLayers[[iLayer]]$Psol	[numeric] series of solid precip. [mm/h]
$CemaNeigeLayers[[iLayer]]$SnowPack	[numeric] series of snow pack (snow water equivalent) [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/h]
$CemaNeigeLayers[[iLayer]]$Melt	[numeric] series of actual snow melt [mm/h]
$CemaNeigeLayers[[iLayer]]$PliqAndMelt	[numeric] series of liquid precip. + actual snow melt [mm/h]
$CemaNeigeLayers[[iLayer]]$Temp	[numeric] series of air temperature [°C]
$CemaNeigeLayers[[iLayer]]$Gthreshold	[numeric] series of melt threshold [mm]
$CemaNeigeLayers[[iLayer]]$Glocalmax	[numeric] series of local melt threshold for hysteresis [mm]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/h]
RunOptions$Param	[numeric] parameter set parameter set used by the model

$StateEnd [numeric] states at the end of the run: store & unit hydrographs levels [mm], C

Refer to the provided references or to the package source code for further details on these model outputs.

Author(s)

Laurent Coron, Guillaume Thirel, Olivier Delaigue, Audrey Valéry, Vazken Andréassian

References

Ficchi, A. (2017). An adaptive hydrological model for multiple time-steps: Diagnostics and improvements based on fluxes consistency. PhD thesis, UPMC - Irstea Antony, Paris, France.

Ficchi, A., Perrin, C. and Andréassian, V. (2019). Hydrological modelling at multiple sub-daily time steps: model improvement via flux-matching. Journal of Hydrology, 575, 1308-1327, doi:10.1016/j.jhydrol.2019.05.084.

Perrin, C., Michel, C. and Andréassian, V. (2003). Improvement of a parsimonious model for streamflow simulation. Journal of Hydrology, 279(1-4), 275-289, doi:10.1016/S00221694(03)00225-7.

Riboust, P., Thirel, G., Le Moine, N. and Ribstein, P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.2478/johh20180004.

Valéry, A., Andréassian, V. and Perrin, C. (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, 517(0), 1166-1175, doi:10.1016/j.jhydrol.2014.04.059.

Valéry, A., Andréassian, V. and Perrin, C. (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:10.1016/j.jhydrol.2014.04.058.

See Also

`RunModel_CemaNeige`, `RunModel_CemaNeigeGR4H`, `RunModel_GR5H`, `Imax`, `CreateInputsModel`, `CreateRunOptions`, `CreateIniStates`.

Examples

```
## Not run:
library(airGR)

## loading catchment data
data(U2345030)
```

```
## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_CemaNeigeGR5H, DatesR = BasinObs$DatesR,
                          Precip = BasinObs$P, PotEvap = BasinObs$E, TempMean = BasinObs$T,
                                   ZInputs = BasinInfo$ZInputs,
                                   HypsoData = BasinInfo$HypsoData, NLayers = 5)

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


## --- original version of CemaNeige

## Imax computation
Imax <- Imax(InputsModel = InputsModel, IndPeriod_Run = Ind_Run,
             TestedValues = seq(from = 0, to = 3, by = 0.2))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR5H, InputsModel = InputsModel,
                               Imax = Imax, IndPeriod_Run = Ind_Run)

## simulation
Param <- c(X1 = 218.537, X2 = -0.009, X3 = 174.862, X4 = 6.674, X5 = 0.000,
           CNX1 = 0.002, CNX2 = 3.787)
OutputsModel <- RunModel_CemaNeigeGR5H(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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)

## End(Not run)
```

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 (or 9 parameters if `IsHyst = TRUE`, see `CreateRunOptions` for details)

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]
CemaNeige X3	(optional) accumulation threshold [mm] (needed if `IsHyst = TRUE`)
CemaNeige X4	(optional) percentage (between 0 and 1) of annual snowfall defining the melt threshold [-] (needed if `IsHyst`

Details

The choice of the CemaNeige version is explained in `CreateRunOptions`.
For further details on the model, see the references section.
For further details on the argument structures and initialisation options, see `CreateRunOptions`.

See `RunModel_GR5J` to look at the diagram of the hydrological model.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/d]
$Precip	[numeric] series of input total precipitation (P) [mm/d]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [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 (Pr) [mm/d]
$Q9	[numeric] series of UH outflow going into branch 9 (Q9) [mm/d]
$Q1	[numeric] series of UH outflow going into branch 1 (Q1) [mm/d]
$Rout	[numeric] series of routing store level (R1) [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 (AExch1+AExch2) [r
$QR	[numeric] series of routing store outflow (Qr) [mm/d]
$QD	[numeric] series of direct flow from UH after exchange (Qd) [mm/d]
$Qsim	[numeric] series of simulated discharge (Q) [mm/d]

$CemaNeigeLayers	[list] CemaNeige outputs (1 element 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 (snow water equivalent) [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]
$CemaNeigeLayers[[iLayer]]$Gthreshold	[numeric] series of melt threshold [mm]
$CemaNeigeLayers[[iLayer]]$Glocalmax	[numeric] series of local melt threshold for hysteresis [mm]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/d]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run: store & unit hydrographs levels [mm], C

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, Audrey Valéry, Vazken Andréassian, Olivier Delaigue, Guillaume Thirel

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.

Pushpalatha, R., Perrin, C., Le Moine, N., Mathevet, T. and Andréassian, V. (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.

Riboust, P., Thirel, G., Le Moine, N. and Ribstein, P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.2478/johh20180004.

Valéry, A., Andréassian, V. and Perrin, C. (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, 517(0), 1166-1175, doi:10.1016/j.jhydrol.2014.04.059.

Valéry, A., Andréassian, V. and Perrin, C. (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:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeige, 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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 179.139, X2 = -0.100, X3 = 203.815, X4 = 1.174, X5 = 2.478,
           CNX1 = 0.977, CNX2 = 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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)


## simulation with the Linear Hysteresis
## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR5J, InputsModel = InputsModel,
                               IndPeriod_Run = Ind_Run, IsHyst = TRUE)
Param <- c(179.139, -0.100, 203.815, 1.174, 2.478, 0.977, 2.774, 100, 0.4)
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, Obs = 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 (or 10 parameters if `IsHyst = TRUE`, see `CreateRunOptions` for details)

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	exponential store depletion coefficient [mm]
CemaNeige X1	weighting coefficient for snow pack thermal state [-]
CemaNeige X2	degree-day melt coefficient [mm/°C/d]
CemaNeige X3	(optional) accumulation threshold [mm] (needed if `IsHyst = TRUE`)
CemaNeige X4	(optional) percentage (between 0 and 1) of annual snowfall defining the melt threshold [-] (needed if `IsHyst`

Details

The choice of the CemaNeige version is explained in `CreateRunOptions`.
For further details on the model, see the references section.
For further details on the argument structures and initialisation options, see `CreateRunOptions`.

See `RunModel_GR6J` to look at the diagram of the hydrological model.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/d]
$Precip	[numeric] series of input total precipitation (P) [mm/d]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [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 (Pr) [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 (R1) [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 (AExch1+AExch2) [r
$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) (R2) [mm]
$QD	[numeric] series of direct flow from UH2 after exchange (Qd) [mm/d]
$Qsim	[numeric] series of simulated discharge (Q) [mm/d]
$CemaNeigeLayers	[list] CemaNeige outputs (1 element 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 (snow water equivalent) [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]
$CemaNeigeLayers[[iLayer]]$Gthreshold	[numeric] series of melt threshold [mm]
$CemaNeigeLayers[[iLayer]]$Glocalmax	[numeric] series of local melt threshold for hysteresis [mm]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/d]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run: store & unit hydrographs levels [mm], C

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, Audrey Valéry, Vazken Andréassian, Olivier Delaigue, Guillaume Thirel

References

Pushpalatha, R., Perrin, C., Le Moine, N., Mathevet, T. and Andréassian, V. (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.

Riboust, P., Thirel, G., Le Moine, N. and Ribstein, P. (2019). Revisiting a simple degree-day model for integrating satellite data: Implementation of SWE-SCA hystereses. Journal of Hydrology and Hydromechanics, 67(1), 70–81, doi:10.2478/johh20180004.

Valéry, A., Andréassian, V. and Perrin, C. (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, 517(0), 1166-1175, doi:10.1016/j.jhydrol.2014.04.059.

Valéry, A., Andréassian, V. and Perrin, C. (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:10.1016/j.jhydrol.2014.04.058.

See Also

RunModel_CemaNeige, 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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))


## --- original version of CemaNeige

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

## simulation
Param <- c(X1 = 116.482, X2 = 0.500, X3 = 72.733, X4 = 1.224, X5 = 0.278, X6 = 30.333,
```

```
           CNX1 = 0.977, CNX2 = 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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)


## --- version of CemaNeige with the Linear Hysteresis

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

## simulation
Param <- c(X1 = 116.482, X2 = 0.500, X3 = 72.733, X4 = 1.224, X5 = 0.278, X6 = 30.333,
           CNX1 = 0.977, CNX2 = 2.774, CNX3 = 100, CNX4 = 0.4)
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, Obs = 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] 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]
$Qsim	[numeric] series of simulated discharge [mm/y]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/y]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$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, Olivier Delaigue, Guillaume Thirel

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(DatesR = BasinObs$DatesR,
                        P = BasinObs$P,
```

```
                        E = BasinObs$E,
                        Qmm = BasinObs$Qmm)
TabSeries <- TabSeries[TabSeries$DatesR < as.POSIXct("2012-09-01", tz = "UTC"), ]
BasinObs <- SeriesAggreg(TabSeries, Format = "%Y",
                         YearFirstMonth = 09,
                         ConvertFun = c("sum", "sum", "sum"))

## 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(X1 = 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, Obs = 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)
```

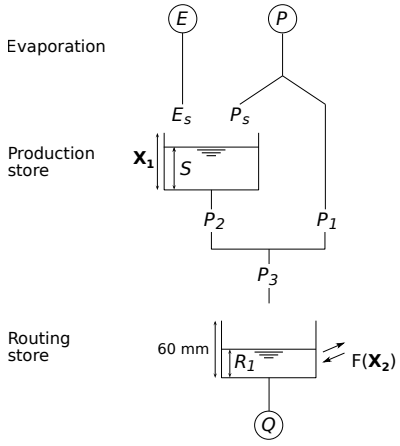
Arguments

`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] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/month] (E)
$Precip	[numeric] series of input total precipitation [mm/month] (P)
$AE	[numeric] series of actual evapotranspiration [mm/month]
$Pn	[numeric] series of net rainfall (P1) [mm/month]
$Ps	[numeric] series of part of P filling the production store [mm/month]
$Perc	[numeric] series of percolation (P2) [mm/month]
$PR	[numeric] series of PR=Pn+Perc (P3) [mm/month]
$AExch	[numeric] series of actual exchange between catchments [mm/month]
$Prod	[numeric] series of production store level (S) [mm]
$Rout	[numeric] series of routing store level (R1) [mm]
$Qsim	[numeric] series of simulated discharge [mm/month] (Q)
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/month]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run (production store level and routing store level) [mm]. See

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, Olivier Delaigue, Guillaume Thirel

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., Michel, C., Perrin, C. and Andréassian, V. (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)

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

## conversion of InputsModel to monthly time step
InputsModel <- SeriesAggreg(InputsModel, Format = "%Y%m")

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

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

## simulation
Param <- c(X1 = 265.072, X2 = 1.040)
OutputsModel <- RunModel_GR2M(InputsModel = InputsModel, RunOptions = RunOptions, Param = Param)

## conversion of observed discharge to monthly time step
Qobs <- SeriesAggreg(data.frame(BasinObs$DatesR, BasinObs$Qmm),
                     Format = "%Y%m",
                     ConvertFun = "sum")
Qobs <- Qobs[Ind_Run, 2]

## results preview
plot(OutputsModel, Qobs = Qobs)
```

```
## efficiency criterion: Nash-Sutcliffe Efficiency
InputsCrit  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = InputsModel,
                                RunOptions = RunOptions, Obs = Qobs)
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`.

See `RunModel_GR4J` to look at the diagram of the hydrological model.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/h]
$Precip	[numeric] series of input total precipitation (P) [mm/h]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/h]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [mm/h
$AE	[numeric] series of actual evapotranspiration [mm/h]

$Perc	[numeric] series of percolation (Perc) [mm/h]
$PR	[numeric] series of Pr=Pn-Ps+Perc (Pr) [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 (R1) [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/h]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/h]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/h]
$AExch	[numeric] series of actual exchange between catchments (AExch1+AExch2) [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]
$Qsim	[numeric] series of simulated discharge (Q) [mm/h]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/h]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run (res. levels, UH1 levels, UH2 levels) [mm]. See CreateIn

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, Olivier Delaigue, Guillaume Thirel

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 (in French), UPMC - Cemagref Antony, Paris, France.

See Also

RunModel_GR4J, RunModel_CemaNeigeGR4H, 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$DatesR, format = "%Y-%m-%d %H")=="2005-01-01 00"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d %H")=="2008-12-31 23"))

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

## simulation
Param <- c(X1 = 756.930, X2 = -0.773, X3 = 138.638, X4 = 5.247)
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, Obs = 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)
```

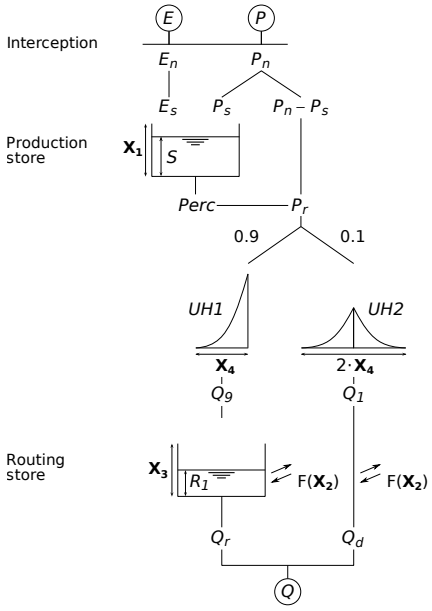
Arguments

`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] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration [mm/d] (E)
$Precip	[numeric] series of input total precipitation (P) [mm/d]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [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 (Pr) [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 (R1) [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]
$Qsim	[numeric] series of simulated discharge (Q) [mm/d]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/d]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run (res. levels, UH1 levels, UH2 levels) [mm]. See CreateIn

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, Olivier Delaigue

References

Perrin, C., Michel, C. and Andréassian, V. (2003). Improvement of a parsimonious model for streamflow simulation. Journal of Hydrology, 279(1-4), 275-289, doi:10.1016/S00221694(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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_GR5H *Run with the GR5H hydrological model*

Description

Function which performs a single run for the GR5H hourly lumped model.

Usage

```
RunModel_GR5H(InputsModel, RunOptions, Param)
```

Arguments

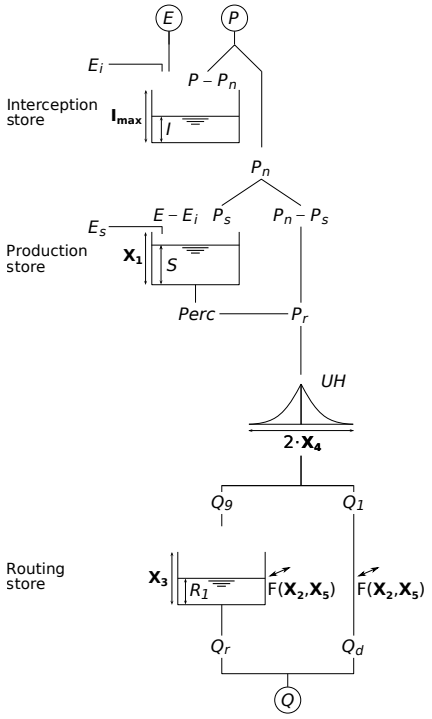
InputsModel	[object of class *InputsModel*] see CreateInputsModel for details
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
Param	[numeric] vector of 5 parameters

GR5H X1	production store capacity [mm]
GR5H X2	intercatchment exchange coefficient [mm/h]
GR5H X3	routing store capacity [mm]
GR5H X4	unit hydrograph time constant [h]
GR5H X5	intercatchment exchange threshold [-]

Details

It is advised to run the GR5H model with an interception store (see Ficchi (2017) and Ficchi et al. (2019)) as it improves the consistency of the model fluxes and provides better performance. To do so, the Imax function allows to estimate the maximal capacity of the interception store, which can then be given to CreateRunOptions.

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



See RunModel_GR5J to look at the diagram of the hydrological model when no interception store is used.

Value

[list] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/h]
$Precip	[numeric] series of input total precipitation (P) [mm/h]
$Interc	[numeric] series of interception store level (I) [mm]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/h]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [mm/h]
$AE	[numeric] series of actual evapotranspiration (Ei+Es) [mm/h]
$EI	[numeric] series of evapotranspiration from rainfall neutralisation or interception store (Ei) [mm
$ES	[numeric] series of evapotranspiration from production store (Es) [mm/h]
$Perc	[numeric] series of percolation (Perc) [mm/h]
$PR	[numeric] series of Pr=Pn-Ps+Perc (Pr) [mm/h]
$Q9	[numeric] series of UH outflow going into branch 9 (Q9) [mm/h]
$Q1	[numeric] series of UH outflow going into branch 1 (Q1) [mm/h]
$Rout	[numeric] series of routing store level (R1) [mm]
$Exch	[numeric] series of potential semi-exchange between catchments [mm/h]
$AExch1	[numeric] series of actual exchange between catchments for branch 1 [mm/h]
$AExch2	[numeric] series of actual exchange between catchments for branch 2 [mm/h]

$AExch	[numeric] series of actual exchange between catchments (AExch1+AExch2) [mm/h]
$QR	[numeric] series of routing store outflow (Qr) [mm/h]
$QD	[numeric] series of direct flow from UH after exchange (Qd) [mm/h]
$Qsim	[numeric] series of simulated discharge (Q) [mm/h]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/h]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run (res. levels, UH levels) [mm]. See CreateIniStates for r

Refer to the provided references or to the package source code for further details on these model outputs.

Author(s)

Laurent Coron, Guillaume Thirel, Olivier Delaigue

References

Ficchi, A. (2017). An adaptive hydrological model for multiple time-steps: Diagnostics and improvements based on fluxes consistency. PhD thesis, UPMC - Irstea Antony, Paris, France.

Ficchi, A., Perrin, C. and Andréassian, V. (2019). Hydrological modelling at multiple sub-daily time steps: model improvement via flux-matching. Journal of Hydrology, 575, 1308-1327, doi:10.1016/j.jhydrol.2019.05.084.

See Also

RunModel_GR4H, RunModel_CemaNeigeGR5H, Imax, CreateInputsModel, CreateRunOptions, CreateIniStates.

Examples

```
library(airGR)

## load of catchment data
data(L0123003)

## preparation of the InputsModel object
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR5H, DatesR = BasinObs$DatesR,
                                 Precip = BasinObs$P, PotEvap = BasinObs$E)

## run period selection
Ind_Run <- seq(which(format(BasinObs$DatesR, format = "%Y-%m-%d %H")=="2006-01-01 00"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d %H")=="2006-12-31 23"))

## Imax computation
Imax <- Imax(InputsModel = InputsModel, IndPeriod_Run = Ind_Run,
             TestedValues = seq(from = 0, to = 3, by = 0.2))

## preparation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR5H, Imax = Imax,
                               InputsModel = InputsModel, IndPeriod_Run = Ind_Run)
```

```
## simulation
Param <- c(X1 = 706.912, X2 = -0.163, X3 = 188.880, X4 = 2.575, X5 = 0.104)
OutputsModel <- RunModel_GR5H(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, Obs = 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)
```

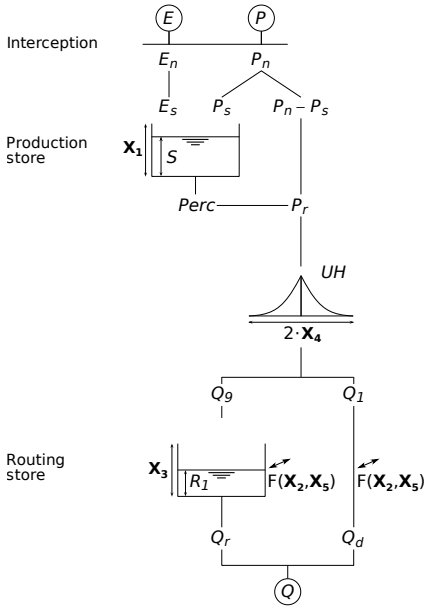
Arguments

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] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/d]
$Precip	[numeric] series of input total precipitation (P) [mm/d]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [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 (Pr) [mm/d]
$Q9	[numeric] series of UH outflow going into branch 9 (Q9) [mm/d]
$Q1	[numeric] series of UH outflow going into branch 1 (Q1) [mm/d]
$Rout	[numeric] series of routing store level (R1) [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 (AExch1+AExch2) [mm/d]
$QR	[numeric] series of routing store outflow (Qr) [mm/d]
$QD	[numeric] series of direct flow from UH after exchange (Qd) [mm/d]
$Qsim	[numeric] series of simulated discharge (Q) [mm/d]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/d]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run (res. levels, UH levels) [mm]. See `CreateIniStates` for r

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, Olivier Delaigue, Guillaume Thirel

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.

Pushpalatha, R., Perrin, C., Le Moine, N., Mathevet, T. and Andréassian, V. (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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 245.918, X2 = 1.027, X3 = 90.017, X4 = 2.198, X5 = 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, Obs = 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)
```

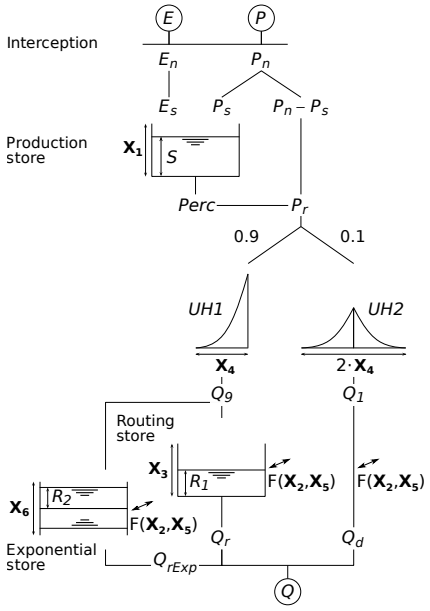
Arguments

`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	exponential store depletion coefficient [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] containing the function outputs organised as follows:

$DatesR	[POSIXlt] series of dates
$PotEvap	[numeric] series of input potential evapotranspiration (E) [mm/d]
$Precip	[numeric] series of input total precipitation (P) [mm/d]
$Prod	[numeric] series of production store level (S) [mm]
$Pn	[numeric] series of net rainfall (Pn) [mm/d]
$Ps	[numeric] series of the part of Pn filling the production store (Ps) [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 (Pr) [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 (R1) [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 (AExch1+AExch2) [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) (R2) [mm]
$QD	[numeric] series of direct flow from UH2 after exchange (Qd) [mm/d]
$Qsim	[numeric] series of simulated discharge (Q) [mm/d]
RunOptions$WarmUpQsim	[numeric] series of simulated discharge (Q) on the warm-up period [mm/d]
RunOptions$Param	[numeric] parameter set parameter set used by the model
$StateEnd	[numeric] states at the end of the run (res. levels, UH1 levels, UH2 levels) [mm]. See CreateIn

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, Olivier Delaigue, Guillaume Thirel

References

Pushpalatha, R., Perrin, C., Le Moine, N., Mathevet, T. and Andréassian, V. (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$DatesR, format = "%Y-%m-%d")=="1990-01-01"),
               which(format(BasinObs$DatesR, format = "%Y-%m-%d")=="1999-12-31"))

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

## simulation
Param <- c(X1 = 242.257, X2 = 0.637, X3 = 53.517, X4 = 2.218, X5 = 0.424, X6 = 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, Obs = BasinObs$Qmm[Ind_Run])
OutputsCrit <- ErrorCrit_NSE(InputsCrit = InputsCrit, OutputsModel = OutputsModel)
```

RunModel_Lag *Run with the Lag model*

Description

Function which performs a single run for the Lag model over the test period.

Usage

```
RunModel_Lag(InputsModel, RunOptions, Param, QcontribDown)
```

Arguments

InputsModel	[object of class *InputsModel*] created with SD model inputs, see CreateInputsModel for details. The object should also contain a key *OutputsModel* of class CreateInputsModel coming from the simulation of the downstream subcatchment runoff.
RunOptions	[object of class *RunOptions*] see CreateRunOptions for details
Param	[numeric] vector of 1 parameter Velocity mean flow velocity [m/s]
QcontribDown	[numeric] vector or [OutputsModel] containing the time series of the runoff contribution of the downstream sub-basin

Value

[list] see RunModel_GR4J or RunModel_CemaNeigeGR4J for details.

The list value contains an extra item named QsimDown which is a copy of the runoff contribution of the downstream sub-basin contained in argument QcontribDown in [mm/time step].

Author(s)

Olivier Delaigue, David Dorchies, Guillaume Thirel

See Also

RunModel, CreateInputsModel, CreateRunOptions.

Examples

```
######################################################################
## Simulation of a reservoir with a purpose of low-flow mitigation ##
######################################################################

## ---- preparation of the InputsModel object

## loading package and catchment data
```

```
library(airGR)
data(L0123001)

## ---- simulation of the hydrological catchment with GR4J

InputsModelDown <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                     Precip = BasinObs$P, PotEvap = BasinObs$E)

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

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

## simulation of the runoff of the catchment with a GR4J model
Param <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
OutputsModelDown <- RunModel_GR4J(InputsModel = InputsModelDown,
                                  RunOptions = RunOptionsDown, Param = Param)

## ---- specifications of the reservoir

## the reservoir withdraws 1 m3/s when it's possible considering the flow observed in the basin
Qupstream <- matrix(-sapply(BasinObs$Qls / 1000 - 1, function(x) {
  min(1, max(0, x, na.rm = TRUE))
}), ncol = 1)

## except between July and September when the reservoir releases 3 m3/s for low-flow mitigation
month <- as.numeric(format(BasinObs$DatesR, "%m"))
Qupstream[month >= 7 & month <= 9] <- 3
Qupstream <- Qupstream * 86400 ## Conversion in m3/day

## the reservoir is not an upstream subcachment: its areas is NA
BasinAreas <- c(NA, BasinInfo$BasinArea)

## delay time between the reservoir and the catchment outlet is 2 days and the distance is 150 km
LengthHydro <- 150

## ---- simulation of the basin with the reservoir influence

InputsModelInf <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = BasinObs$DatesR,
                                    Precip = BasinObs$P, PotEvap = BasinObs$E,
                                    Qupstream = Qupstream, LengthHydro = LengthHydro,
                                    BasinAreas = BasinAreas)
## creation of the RunOptions object
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J,
                               InputsModel = InputsModelInf, IndPeriod_Run = Ind_Run)

## with a delay of 2 days for 150 km, the flow velocity is 75 km per day
Velocity <- (LengthHydro * 1e3 / 2) / (24 * 60 * 60) ## Conversion km/day -> m/s

## run the lag model which routes precipitation-runoff model and upstream flows
```

```
OutputsModel <- RunModel_Lag(InputsModel = InputsModelInf,
                             RunOptions = RunOptions,
                             Param = Velocity,
                             QcontribDown = OutputsModelDown)

## results preview of comparison between naturalised (observed) and influenced flow (simulated)
plot(OutputsModel, Qobs = OutputsModel$QsimDown)
```

SeriesAggreg *Conversion of time series to another time step (aggregation only) and regime computation*

Description

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

Usage

```
## S3 method for class 'data.frame'
SeriesAggreg(x,
             Format,
             ConvertFun,
             TimeFormat = NULL,
             NewTimeFormat = NULL,
             YearFirstMonth = 1,
             TimeLag = 0,
             ...)

## S3 method for class 'list'
SeriesAggreg(x,
             Format,
             ConvertFun,
             NewTimeFormat = NULL,
             simplify = FALSE,
             except = NULL,
             recursive = TRUE,
             ...)

## S3 method for class 'InputsModel'
SeriesAggreg(x, Format, ...)
```

```
## S3 method for class 'OutputsModel'
SeriesAggreg(x, Format, ...)
```

Arguments

x	[InputsModel], [OutputsModel], [list] or [data.frame] containing the vector of dates (*POSIXt*) and the time series of numeric values
Format	[character] output time step format (i.e. yearly times series: "%Y", monthly time series: "%Y%m", daily time series: "%Y%m%d", monthly regimes: "%m", daily regimes: "%d")
TimeFormat	(deprecated) [character] input time step format (i.e. "hourly", "daily", "monthly" or "yearly"). Use the x argument instead
NewTimeFormat	(deprecated) [character] output time step format (i.e. "hourly", "daily", "monthly" or "yearly"). Use the Format argument instead
ConvertFun	[character] names of aggregation functions (e.g. for P[mm], T[degC], Q[mm]: ConvertFun = c("sum", "mean", "sum")) or name of aggregation function to apply to all elements if the parameter 'x' is a [list] . See details
YearFirstMonth	(optional) [numeric] integer used when Format = "%Y" 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 into daily time series)
simplify	(optional) [boolean] if set to TRUE, a data.frame is returned instead of a list. Embedded lists are then ignored. (default = FALSE)
except	(optional) [character] the name of the items to skip in the aggregation (default = NULL)
recursive	(optional) [boolean] if set to FALSE, embedded lists and dataframes are not aggregated (default = TRUE)
...	Arguments passed to SeriesAggreg.list and then to SeriesAggreg.data.frame

Details

SeriesAggreg.InputsModel and SeriesAggreg.OutputsModel call SeriesAggreg.list which itself calls SeriesAggreg.data.frame. So, all arguments passed to any SeriesAggreg method will be passed to SeriesAggreg.data.frame.

Argument ConvertFun also supports quantile calculation by using the syntax "Q[nn]" with [nn] the requested percentile. E.g. use "Q90" for calculating 90th percentile in the aggregation. The formula used is: quantile(x, probs = perc / 100, type = 8, na.rm = TRUE).

As there are multiple ways to take into account missing values in aggregation functions, NAs are not supported by SeriesAggreg and it provides NA values when NAs are present in the x input.

Value

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

Author(s)

Olivier Delaigue, David Dorchies

Examples

```
library(airGR)

## loading catchment data
data(L0123002)

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

## monthly time series
NewTabSeries <- SeriesAggreg(TabSeries,
                             Format = "%Y%m",
                             ConvertFun = c("sum", "sum", "mean", "sum"))
str(NewTabSeries)

## monthly regimes
NewTabSeries <- SeriesAggreg(TabSeries,
                             Format = "%m",
                             ConvertFun = c("sum", "sum", "mean", "sum"))
str(NewTabSeries)

## conversion of InputsModel
example("RunModel_GR2M")

## monthly regimes on OutputsModel object
SimulatedMonthlyRegime <- SeriesAggreg(OutputsModel, Format = "%m")
str(SimulatedMonthlyRegime)
```

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

```
## Generic function
TransfoParam(ParamIn, Direction, FUN_TRANSFO)

## Specific functions
TransfoParam_GR1A(ParamIn, Direction)
TransfoParam_GR2M(ParamIn, Direction)
```

```
TransfoParam_GR4J(ParamIn, Direction)
TransfoParam_GR5J(ParamIn, Direction)
TransfoParam_GR6J(ParamIn, Direction)
TransfoParam_GR4H(ParamIn, Direction)
TransfoParam_GR5H(ParamIn, Direction)
TransfoParam_CemaNeige(ParamIn, Direction)
TransfoParam_CemaNeigeHyst(ParamIn, Direction)
```

Arguments

ParamIn	[numeric] vector or 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)

Details

The transformation functions proposed in airGR for calibrating the GR models result from numerous testings at INRAE-Antony (HYCAR Research Unit, France). The proposed transformations were obtained with the Calibration_Michel algorithm and may differ for the same parameter of different models (e.g. X5 in GR5J and GR6J).

Value

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

Author(s)

Laurent Coron, Olivier Delaigue

Examples

```
library(airGR)

## --- generic function

## 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)


## --- specific function

## 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")
```

Index