Package 'airGRiwrm'

October 12, 2022

Title 'airGR' Integrated Water Resource Management

Version 0.6.1

Date 2022-03-03

Description Semi-distributed Precipitation-Runoff Modelling based on 'airGR' package models integrating human infrastructures and their managements.

License AGPL-3

URL https://airgriwrm.g-eau.fr/,
https://github.com/inrae/airGRiwrm#readme

BugReports https://github.com/inrae/airGRiwrm/issues

Depends airGR (>= 1.7.0), R (>= 2.10)

Imports DiagrammeR, dplyr, graphics, grDevices, utils

Suggests spelling, htmltools, knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Language en-US

NeedsCompilation no

Author David Dorchies [aut, cre] (<https://orcid.org/0000-0002-6595-7984>), Olivier Delaigue [ctb] (<https://orcid.org/0000-0002-7668-8468>), Guillaume Thirel [ctb] (<https://orcid.org/0000-0002-1444-1830>)

Maintainer David Dorchies <david.dorchies@inrae.fr>

Repository CRAN

Date/Publication 2022-03-08 20:20:08 UTC

R topics documented:

Calibration.GRiwrmInputsModel 2
ConvertMeteoSD 4
CreateCalibOptions.GRiwrmInputsModel 5
CreateController 6
CreateGRiwrm 7
CreateInputsCrit.GRiwrmInputsModel 10
CreateInputsModel 12
CreateInputsModel.GRiwrm 12
CreateRunOptions.GRiwrmInputsModel 15
CreateSupervisor 18
getNoSD_Ids 19
getSD_Ids 20
isNodeDownstream 20
plot.GRiwrm 21
plot.GRiwrmOutputsModel 22
plot.Qm3s 24
RunModel 25
RunModel.GR 26
RunModel.GRiwrmInputsModel 26
RunModel.InputsModel 29
RunModel.SD 30
RunModel.Supervisor 30
Severn 31

Index **32**

`Calibration.GRiwrmInputsModel` *Calibration of the parameters of one catchment or a network of sub-catchments*

Description

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

Usage

```
## S3 method for class 'GRiwrmInputsModel'
Calibration(
  InputsModel,
  RunOptions,
  InputsCrit,
  CalibOptions,
  useUpstreamQsim = TRUE,
  ...
```

```
)

## S3 method for class 'InputsModel'
Calibration(InputsModel, ...)

Calibration(InputsModel, ...)
```

Arguments

`InputsModel`	[object of class *InputsModel* or *GRiwrmInputsModel*] see CreateInputsModel
`RunOptions`	[object of class *RunOptions* or *GRiwrmRunOptions*] see CreateRunOptions
`InputsCrit`	[object of class *InputsCrit* or *GRiwrmInputsCrit*] see CreateInputsCrit
`CalibOptions`	[object of class *CalibOptions* or *GRiwrmCalibOptions*] see CreateCalibOptions for details
`useUpstreamQsim`	boolean describing if simulated (`TRUE`) or observed (`FALSE`) flows are used for calibration. Default is `TRUE`
`...`	further arguments passed to airGR::Calibration, see details

Details

This function can be used either for a catchment (with an *InputsModel* object) or for a network (with a *GRiwrmInputsModel* object)

Argument classes should be consistent to the usage:

- a `InputsModel` argument of class *InputsModel* must be followed by a `RunOptions` argument of class *RunOptions*, a `InputsCrit` argument of class *InputsCrit* and a `CalibOptions` of class *CalibOptions*
- – a `InputsModel` argument of class *GRiwrmInputsModel* must be followed by a `RunOptions` argument of class *GRiwrmRunOptions*, a `InputsCrit` argument of class *GRiwrmInputsCrit* and a `CalibOptions` of class *GRiwrmCalibOptions*

See the vignettes for examples.

Value

Depending on the class of `InputsModel` argument (respectively `InputsModel` and `GRiwrmInputsModel` object), the returned value is respectively:

- a `InputsCrit` object (See airGR::CreateInputsCrit)
- a `GRiwrmInputsCrit` object which is a list of `InputsCrit` objects with one item per modeled sub-catchment

ConvertMeteoSD *Conversion of meteorological data from basin scale to sub-basin scale*

Description

Conversion of meteorological data from basin scale to sub-basin scale

Usage

```
ConvertMeteoSD(x, ...)

## S3 method for class 'GRiwrm'
ConvertMeteoSD(x, meteo, ...)

## S3 method for class 'character'
ConvertMeteoSD(x, griwrm, meteo, ...)

## S3 method for class 'matrix'
ConvertMeteoSD(x, areas, temperature = FALSE, ...)
```

Arguments

x	either a GRiwrm network description (See CreateGRiwrm), a character id of a node, or a matrix containing meteorological data
...	Parameters passed to the methods
meteo	matrix or data.frame containing meteorological data. Its colnames should be equal to the ID of the basins
griwrm	GRiwrm object describing the semi-distributed network (See CreateGRiwrm)
areas	numeric vector with the total area of the basin followed by the areas of the upstream basins in km2
temperature	logical TRUE if the meteorological data contain air temperature. If FALSE minimum output values are bounded to zero

Value

matrix a matrix containing the converted meteorological data

`CreateCalibOptions.GRiwrmInputsModel`
Creation of the CalibOptions object

Description

This function can be used either for a catchment (with an *InputsModel* object) or for a network (with a *GRiwrmInputsModel* object)

Usage

```
## S3 method for class 'GRiwrmInputsModel'
CreateCalibOptions(x, ...)

CreateCalibOptions(x, ...)

## S3 method for class 'InputsModel'
CreateCalibOptions(x, ...)

## S3 method for class 'character'
CreateCalibOptions(x, ...)

## S3 method for class '`function`'
CreateCalibOptions(x, ...)
```

Arguments

`x`	For a single catchment, it can be an object of class *InputsModel* or a function or a character corresponding to `FUN_MOD` (compliant with **airGR** call). For a network, it should be an object of class *GRiwrmInputsModel*. See CreateInputsModel for details
`...`	arguments passed to airGR::CreateCalibOptions, see details

Details

See airGR::CreateCalibOptions documentation for a complete list of arguments.

With a *GRiwrmInputsModel* object, all arguments are applied on each sub-catchments of the network.

Value

Depending on the class of `InputsModel` argument (respectively `InputsModel` and `GRiwrmInputsModel` object), the returned value is respectively:

- a `CalibOptions` object (See airGR::CreateCalibOptions)
- a `GRiwrmCalibOptions` object which is a list of `CalibOptions` object with one item per modeled sub-catchment

`CreateController` *Creation and adding of a controller in a supervisor*

Description

Creation and adding of a controller in a supervisor

Usage

```
CreateController(supervisor, ctrl.id, Y, U, FUN)
```

Arguments

`supervisor`	`Supervisor` object, see CreateSupervisor
`ctrl.id`	character id of the controller (see Details)
`Y`	character location of the controlled and/or measured variables in the model.
`U`	character location of the command variables in the model.
`FUN`	function controller logic which calculates `U` from `Y` (see Details)

Details

The `ctrl.id` is a unique id for finding the controller in the supervisor. If a controller with the same id already exists, it is overwritten by this new one.

`FUN` should be a function with one numeric parameter. This parameter will receive the measured values of at `Y` locations as input for the previous time step and returns calculated `U`. These `U` will then be applied at their location for the current time step of calculation of the model.

Value

a `Controller` object which is a list with the following items:

- `id` character: the controller identifier
- `U` matrix: the list of controls for command variables with each column being the location of the variables and the rows being the values of the variable for the current time steps (empty by default)
- `Unames` character: location of the command variables
- `Y` matrix: the lists of controls for controlled variables with each column being the location of the variables and the rows being the values of the variable for the current time steps (empty by default)
- `Ynames` character: location of the controlled variables
- `FUN` function: controller logic which calculates `U` from `Y`

Examples

```
# First create a Supervisor from a model
data(Severn)
nodes <- Severn$BasinsInfo[, c("gauge_id", "downstream_id", "distance_downstream", "area")]
nodes$model <- "RunModel_GR4J"
griwrm <- CreateGRiwrm(nodes,
                  list(id = "gauge_id",
                       down = "downstream_id",
                       length = "distance_downstream"))
BasinsObs <- Severn$BasinsObs
DatesR <- BasinsObs[[1]]$DatesR
PrecipTot <- cbind(sapply(BasinsObs, function(x) {x$precipitation}))
PotEvapTot <- cbind(sapply(BasinsObs, function(x) {x$peti}))
Qobs <- cbind(sapply(BasinsObs, function(x) {x$discharge_spec}))
Precip <- ConvertMeteoSD(griwrm, PrecipTot)
PotEvap <- ConvertMeteoSD(griwrm, PotEvapTot)
InputsModel <- CreateInputsModel(griwrm, DatesR, Precip, PotEvap, Qobs)
sv <- CreateSupervisor(InputsModel)

# A controller which usually releases 0.1 m3/s and provides
# extra release if the downstream flow is below 0.5 m3/s
logicDamRelease <- function(Y) max(0.5 - Y[1], 0.1)
CreateController(sv, "DamRelease", Y = c("54001"), U = c("54095"), FUN = logicDamRelease)
```

`CreateGRiwrm` *Generation of a network description containing all hydraulic nodes and the description of their connections*

Description

Generation of a network description containing all hydraulic nodes and the description of their connections

Usage

```
CreateGRiwrm(
  db,
  cols = list(id = "id", down = "down", length = "length", model = "model", area =
    "area"),
  keep_all = FALSE
)
```

Arguments

db	data.frame description of the network (See details)
cols	list or vector columns of db. By default, mandatory column names are: id, down, length. Other names can be handled with a named list or vector containing items defined as "required name" = "column name in db"

keep_all — logical indicating if all columns of db should be kept or if only columns defined in cols should be kept

Details

db is a data.frame which at least contains in its columns:

- a node identifier (column id),
- the identifier and the hydraulic distance to the downstream node (character columns down and numeric columns length in km). The last downstream node should have fields down and length set to NA,
- the area of the basin (numeric column area in km2)
- the hydrological model to use if necessary (character column model) (NA for using observed flow instead of a runoff model output)

Value

data.frame of class GRiwrm describing the airGR semi-distributed model network, with each line corresponding to a location on the river network and with the following columns:

- id (character): node identifier
- down (character): identifier of the node downstream of the current node (NA for the most downstream node)
- length (numeric): hydraulic distance to the downstream node in km (NA for the most downstream node)
- area (numeric): total area of the basin starting from the current node location in km2
- model (character): hydrological model to use if necessary (NA for using observed flow instead of a runoff model output)

Examples

```
######################################################################
# Run the `airGR::RunModel_Lag` example in the GRiwrm fashion way #
# Simulation of a reservoir with a purpose of low-flow mitigation #
######################################################################

## ---- preparation of the InputsModel object

## loading package and catchment data
library(airGRiwrm)
data(L0123001)

## ---- 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
## 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

# This example is a network of 2 nodes which can be describe like this:
db <- data.frame(id = c("Reservoir", "GaugingDown"),
                 length = c(LengthHydro, NA),
                 down = c("GaugingDown", NA),
                 area = c(NA, BasinInfo$BasinArea),
                 model = c(NA, "RunModel_GR4J"),
                 stringsAsFactors = FALSE)

# Create GRiwrm object from the data.frame
griwrm <- CreateGRiwrm(db)
str(griwrm)

# Formatting observations for the hydrological models
# Each input data should be a matrix or a data.frame with the good id in the name of the column
Precip <- matrix(BasinObs$P, ncol = 1)
colnames(Precip) <- "GaugingDown"
PotEvap <- matrix(BasinObs$E, ncol = 1)
colnames(PotEvap) <- "GaugingDown"

# Observed flows contain flows that are directly injected in the model
Qobs = matrix(Qupstream, ncol = 1)
colnames(Qobs) <- "Reservoir"

# Creation of the GRiwrmInputsModel object (= a named list of InputsModel objects)
InputsModels <- CreateInputsModel(griwrm,
                            DatesR = BasinObs$DatesR,
                            Precip = Precip,
                            PotEvap = PotEvap,
                            Qobs = Qobs)
str(InputsModels)

## 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 GriwmRunOptions object
RunOptions <- CreateRunOptions(InputsModels,
                               IndPeriod_Run = Ind_Run)
str(RunOptions)
```

```
# Parameters of the SD models should be encapsulated in a named list
ParamGR4J <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
Param <- list(`GaugingDown` = c(Velocity, ParamGR4J))

# RunModel for the whole network
OutputsModels <- RunModel(InputsModels,
                          RunOptions = RunOptions,
                          Param = Param)
str(OutputsModels)

# Compare Simulation with reservoir and observation of natural flow
plot(OutputsModels, data.frame(GaugingDown = BasinObs$Qmm[Ind_Run]))
```

`CreateInputsCrit.GRiwrmInputsModel`

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

Description

This function can be used either for a catchment (with an *InputsModel* object) or for a network (with a *GRiwrmInputsModel* object)

Usage

```
## S3 method for class 'GRiwrmInputsModel'
CreateInputsCrit(
  InputsModel,
  FUN_CRIT = ErrorCrit_NSE,
  RunOptions,
  Obs,
  AprioriIds = NULL,
  k = 0.15,
  AprCelerity = 1,
  ...
)

## S3 method for class 'InputsModel'
CreateInputsCrit(InputsModel, FUN_CRIT, ...)

CreateInputsCrit(InputsModel, ...)
```

Arguments

`InputsModel`	object of class *InputsModel* or *GRiwrmInputsModel*. See CreateInputsModel
`FUN_CRIT`	[function (atomic or list)] error criterion function (e.g. airGR::ErrorCrit_RMSE, airGR::ErrorCrit_NSE)
`RunOptions`	object of class *RunOptions* or *GRiwrmRunOptions*, see CreateRunOptions

`Obs`	numeric, matrix or data.frame series of observed flows, see details
`AprioriIds`	(optional) named list or named vector of character used for the parameter regularization (see details)
`k`	(optional) numeric weight coefficient used in the parameter regularization (See airGR::CreateInputsCrit_Lavenne)
`AprCelerity`	(optional) numeric Default celerity used as a priori parameter for upstream catchments
`...`	arguments passed to airGR::CreateInputsCrit, see details

Details

See airGR::CreateInputsCrit documentation for a complete list of arguments.

`Obs` argument is equivalent to the same argument in airGR::CreateInputsCrit except that it must be a matrix or a data.frame if `InputsModel` is a *GRiwrmInputsModel* object. Then, each column of the matrix or data.frame represents the observations of one of the simulated node with the name of the columns representing the id of each node.

With a *GRiwrmInputsModel* object, all arguments are applied on each sub-catchments of the network.

Parameter regularization consists of defining a priori parameters which are used in a composed criterion based on the formula proposed by Lavenne et al. (2019) (See airGR::CreateInputsCrit_Lavenne). The parameter `AprioriIds` allows to define which upstream sub-catchment is used for providing a priori parameters. Its format is as follows: `AprioriIds <- c("Downstream sub-catchment 1" = "A priori upstream sub-catchment 1", ...)` where the quoted strings are the ids of the sub-catchments. See vignettes for more details. The parameter `AprCelerity` is a default value used as a priori for the parameter 'Celerity' in case of an upstream catchment (without celerity parameter) is used as a priori catchment.

Value

Depending on the class of `InputsModel` argument (respectively `InputsModel` and `GRiwrmInputsModel` object), the returned value is respectively:

- a `InputsCrit` object (See airGR::CreateInputsCrit)
- a `GRiwrmInputsCrit` object which is a list of `InputsCrit` objects with one item per modeled sub-catchment

References

De Lavenne, A., Andréassian, V., Thirel, G., Ramos, M.-H., 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

CreateInputsModel *Generic function for creating* InputsModel *object for either* **airGR** *or* **airGRiwrm**

Description

See the methods CreateInputsModel.GRiwrm for **airGRiwrm** and airGR::CreateInputsModel for **airGR**.

Usage

```
CreateInputsModel(x, ...)

## Default S3 method:
CreateInputsModel(x, ...)
```

Arguments

x	First parameter determining which InputsModel object is created
...	further arguments passed to or from other methods.

Value

InputsModel or GRiwrmInputsObject object

CreateInputsModel.GRiwrm *Creation of an InputsModel object for a* **airGRiwrm** *network*

Description

Creation of an InputsModel object for a **airGRiwrm** network

Usage

```
## S3 method for class 'GRiwrm'
CreateInputsModel(
  x,
  DatesR,
  Precip = NULL,
  PotEvap = NULL,
  Qobs = NULL,
  PrecipScale = TRUE,
  TempMean = NULL,
  TempMin = NULL,
```

```
  TempMax = NULL,
  ZInputs = NULL,
  HypsoData = NULL,
  NLayers = 5,
  ...
)
```

Arguments

x	[GRiwrm object] diagram of the semi-distributed model (See CreateGRiwrm)
DatesR	POSIXt vector of dates
Precip	(optional) matrix or data.frame frame of numeric containing precipitation in [mm per time step]. Column names correspond to node IDs
PotEvap	(optional) matrix or data.frame frame of numeric containing potential evaporation [mm per time step]. Column names correspond to node IDs
Qobs	(optional) matrix or data.frame frame of numeric containing observed flows in [mm per time step]. Column names correspond to node IDs
PrecipScale	(optional) named vector of logical 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	(optional) matrix or data.frame of time series of mean air temperature [°C], required to create the CemaNeige module inputs
TempMin	(optional) matrix or data.frame of time series of minimum air temperature [°C], possibly used to create the CemaNeige module inputs
TempMax	(optional) matrix or data.frame of time series of maximum air temperature [°C], possibly used to create the CemaNeige module inputs
ZInputs	(optional) named vector of numeric giving the mean elevation of the Precip and Temp series (before extrapolation) [m], possibly used to create the CemaNeige module input
HypsoData	(optional) matrix or data.frame containing 101 numeric rows: min, q01 to q99 and max of catchment elevation distribution [m], if not defined a single elevation is used for CemaNeige
NLayers	(optional) named vector of numeric integer giving the number of elevation layers requested -, required to create CemaNeige module inputs, default=5
...	used for compatibility with S3 methods

Details

Meteorological data are needed for the nodes of the network that represent a catchment simulated by a rainfall-runoff model. Instead of airGR::CreateInputsModel that has numeric vector as time series inputs, this function uses matrix or data.frame with the id of the sub-catchment as column names. For single values (ZInputs or NLayers), the function requires named vector with the id of the sub-catchment as name item. If an argument is optional, only the column or the named item has to be provided.

See airGR::CreateInputsModel documentation for details concerning each input.

Value

A *GRiwrmInputsModel* object which is a list of *InputsModel* objects created by airGR::CreateInputsModel with one item per modeled sub-catchment.

Examples

```
######################################################################
# Run the `airGR::RunModel_Lag` example in the GRiwrm fashion way #
# Simulation of a reservoir with a purpose of low-flow mitigation #
######################################################################

## ---- preparation of the InputsModel object

## loading package and catchment data
library(airGRiwrm)
data(L0123001)

## ---- 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
## 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

# This example is a network of 2 nodes which can be describe like this:
db <- data.frame(id = c("Reservoir", "GaugingDown"),
                 length = c(LengthHydro, NA),
                 down = c("GaugingDown", NA),
                 area = c(NA, BasinInfo$BasinArea),
                 model = c(NA, "RunModel_GR4J"),
                 stringsAsFactors = FALSE)

# Create GRiwrm object from the data.frame
griwrm <- CreateGRiwrm(db)
str(griwrm)

# Formatting observations for the hydrological models
# Each input data should be a matrix or a data.frame with the good id in the name of the column
Precip <- matrix(BasinObs$P, ncol = 1)
```

```
colnames(Precip) <- "GaugingDown"
PotEvap <- matrix(BasinObs$E, ncol = 1)
colnames(PotEvap) <- "GaugingDown"

# Observed flows contain flows that are directly injected in the model
Qobs = matrix(Qupstream, ncol = 1)
colnames(Qobs) <- "Reservoir"

# Creation of the GRiwrmInputsModel object (= a named list of InputsModel objects)
InputsModels <- CreateInputsModel(griwrm,
                            DatesR = BasinObs$DatesR,
                            Precip = Precip,
                            PotEvap = PotEvap,
                            Qobs = Qobs)
str(InputsModels)

## 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 GriwmRunOptions object
RunOptions <- CreateRunOptions(InputsModels,
                               IndPeriod_Run = Ind_Run)
str(RunOptions)

# Parameters of the SD models should be encapsulated in a named list
ParamGR4J <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
Param <- list(`GaugingDown` = c(Velocity, ParamGR4J))

# RunModel for the whole network
OutputsModels <- RunModel(InputsModels,
                          RunOptions = RunOptions,
                          Param = Param)
str(OutputsModels)

# Compare Simulation with reservoir and observation of natural flow
plot(OutputsModels, data.frame(GaugingDown = BasinObs$Qmm[Ind_Run]))
```

`CreateRunOptions.GRiwrmInputsModel`

Creation of the RunOptions object

Description

This function can be used either for a catchment (with an *InputsModel* object) or for a network (with a *GRiwrmInputsModel* object)

Usage

```
## S3 method for class 'GRiwrmInputsModel'
```

```
CreateRunOptions(x, IniStates = NULL, ...)

CreateRunOptions(x, ...)

## S3 method for class 'InputsModel'
CreateRunOptions(x, ...)

## S3 method for class 'character'
CreateRunOptions(x, ...)

## S3 method for class '`function`'
CreateRunOptions(x, ...)
```

Arguments

x	For a single catchment, it can be an object of class *InputsModel* or a function or a character corresponding to FUN_MOD (compliant with **airGR** call). For a network, it should be an object of class *GRiwrmInputsModel*. See CreateInputsModel for details
IniStates	(optional) numeric object or list of numeric object of class *IniStates*, see airGR::CreateIniStates for details
...	arguments passed to airGR::CreateRunOptions, see details

Details

See airGR::CreateRunOptions documentation for a complete list of arguments.

If InputsModel argument is a *GRiwrmInputsModel* object, IniStates must be a list of numeric object of class *IniStates* with one item per modeled sub-catchment.

With a *GRiwrmInputsModel* object, all arguments are applied on each sub-catchments of the network.

Value

Depending on the class of InputsModel argument (respectively *InputsModel* and *GRiwrmInputsModel* object), the returned value is respectively:

- a RunOptions object (See airGR::CreateRunOptions)
- a GRiwrmRunOptions object which is a list of RunOptions objects with one item per modeled sub-catchment

Examples

```
########################################################################
# Run the `airGR::RunModel_Lag` example in the GRiwrm fashion way      #
# Simulation of a reservoir with a purpose of low-flow mitigation      #
########################################################################

## ---- preparation of the InputsModel object
```

```
## loading package and catchment data
library(airGRiwrm)
data(L0123001)

## ---- 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
## 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

# This example is a network of 2 nodes which can be describe like this:
db <- data.frame(id = c("Reservoir", "GaugingDown"),
                 length = c(LengthHydro, NA),
                 down = c("GaugingDown", NA),
                 area = c(NA, BasinInfo$BasinArea),
                 model = c(NA, "RunModel_GR4J"),
                 stringsAsFactors = FALSE)

# Create GRiwrm object from the data.frame
griwrm <- CreateGRiwrm(db)
str(griwrm)

# Formatting observations for the hydrological models
# Each input data should be a matrix or a data.frame with the good id in the name of the column
Precip <- matrix(BasinObs$P, ncol = 1)
colnames(Precip) <- "GaugingDown"
PotEvap <- matrix(BasinObs$E, ncol = 1)
colnames(PotEvap) <- "GaugingDown"

# Observed flows contain flows that are directly injected in the model
Qobs = matrix(Qupstream, ncol = 1)
colnames(Qobs) <- "Reservoir"

# Creation of the GRiwrmInputsModel object (= a named list of InputsModel objects)
InputsModels <- CreateInputsModel(griwrm,
                            DatesR = BasinObs$DatesR,
                            Precip = Precip,
                            PotEvap = PotEvap,
                            Qobs = Qobs)
```

```
str(InputsModels)

## 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 GriwmRunOptions object
RunOptions <- CreateRunOptions(InputsModels,
                               IndPeriod_Run = Ind_Run)
str(RunOptions)

# Parameters of the SD models should be encapsulated in a named list
ParamGR4J <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
Param <- list(`GaugingDown` = c(Velocity, ParamGR4J))

# RunModel for the whole network
OutputsModels <- RunModel(InputsModels,
                          RunOptions = RunOptions,
                          Param = Param)
str(OutputsModels)

# Compare Simulation with reservoir and observation of natural flow
plot(OutputsModels, data.frame(GaugingDown = BasinObs$Qmm[Ind_Run]))
```

`CreateSupervisor` *Creation of a Supervisor for handling regulation in a model*

Description

Creation of a Supervisor for handling regulation in a model

Usage

```
CreateSupervisor(InputsModel, TimeStep = 1L)
```

Arguments

`InputsModel`	[object of type `GRiwrmInputsModel`] inputs of the model
`TimeStep`	numeric number of time steps between each supervision

Value

A `Supervisor` object which is an environment containing all the necessary variables to run a supervised simulation, such as:

- `DatesR` POSIXct: vector of date from `InputsModel`
- `InputsModel`: a copy of `InputsModel` provided by CreateInputsModel.GRiwrm
- `griwrm`: a copy of `griwrm` provided by CreateGRiwrm

- `Controllers` list: list of the controllers used in the supervised simulation (See CreateController)
- some internal state variables updated during simulation (`ts.index`, `ts.previous`, `ts.date`, `ts.index0`, `controller.id`)

Examples

```
data(Severn)
nodes <- Severn$BasinsInfo[, c("gauge_id", "downstream_id", "distance_downstream", "area")]
nodes$model <- "RunModel_GR4J"
griwrm <- CreateGRiwrm(nodes,
                  list(id = "gauge_id",
                       down = "downstream_id",
                       length = "distance_downstream"))
BasinsObs <- Severn$BasinsObs
DatesR <- BasinsObs[[1]]$DatesR
PrecipTot <- cbind(sapply(BasinsObs, function(x) {x$precipitation}))
PotEvapTot <- cbind(sapply(BasinsObs, function(x) {x$peti}))
Qobs <- cbind(sapply(BasinsObs, function(x) {x$discharge_spec}))
Precip <- ConvertMeteoSD(griwrm, PrecipTot)
PotEvap <- ConvertMeteoSD(griwrm, PotEvapTot)
InputsModel <- CreateInputsModel(griwrm, DatesR, Precip, PotEvap, Qobs)
sv <- CreateSupervisor(InputsModel)
```

`getNoSD_Ids` *Function to obtain the ID of sub-basins not using SD model*

Description

Function to obtain the ID of sub-basins not using SD model

Usage

```
getNoSD_Ids(InputsModel)
```

Arguments

`InputsModel`	[`GRiwrmInputsModel` object]

Value

character IDs of the sub-basins not using the SD model

getSD_Ids *Function to obtain the ID of sub-basins using SD model*

Description

Function to obtain the ID of sub-basins using SD model

Usage

```
getSD_Ids(InputsModel)
```

Arguments

InputsModel	[GRiwrmInputsModel object]

Value

character IDs of the sub-basins using SD model

isNodeDownstream *Check if a node is downstream another one*

Description

Check if a node is downstream another one

Usage

```
isNodeDownstream(InputsModel, current_node, down_node)
```

Arguments

InputsModel	[GRiwrmInputsModel object] see CreateInputsModel.GRiwrm for details
current_node	character with the id of the current node
down_node	character with the id of the node for which we want to know if it is downstream current_node

Value

logical TRUE if the node with the id down_node is downstream the node with the id current_node

`plot.GRiwrm` *Display of a diagram representing the network structure of a GRiwrm object*

Description

Display of a diagram representing the network structure of a GRiwrm object

Usage

```
## S3 method for class 'GRiwrm'
plot(
  x,
  display = TRUE,
  orientation = "LR",
  width = "100%",
  height = "100%",
  ...
)
```

Arguments

`x`	[GRiwrm object] data to display. See CreateGRiwrm for details
`display`	logical if `TRUE` displays the diagram with DiagrammeR::mermaid, returns the mermaid code otherwise
`orientation`	character orientation of the graph. Possible values are "LR" (left-right), "RL" (right-left), "TB" (top-bottom), or "BT" (bottom-top). "LR" by default
`width`	numeric width of the resulting graphic in pixels (See DiagrammeR::mermaid)
`height`	numeric height of the resulting graphic in pixels (See DiagrammeR::mermaid)
`...`	Other arguments and parameters you would like to send to JavaScript (See DiagrammeR::mermaid)

Details

This function only works inside RStudio because the HTMLwidget produced by DiagrammeR is not handled on some platforms

Value

Mermaid code of the diagram if display is `FALSE`, otherwise the function returns the diagram itself.

Examples

```
## Not run:
# Display diagram
plot.GRiwrm(griwrm)
# Is the same as
DiagrammeR::mermaid(plot.GRiwrm(griwrm, display = FALSE), width = "100%", height = "100%")

## End(Not run)
```

`plot.GRiwrmOutputsModel`

Function which creates screen plots giving an overview of the model outputs in the GRiwrm network

---

Description

Function which creates screen plots giving an overview of the model outputs in the GRiwrm network

Usage

```
## S3 method for class 'GRiwrmOutputsModel'
plot(x, Qobs = NULL, ...)
```

Arguments

`x`	[object of class *GRiwrmOutputsModel*] see RunModel.GRiwrmInputsModel for details
`Qobs`	(optional) matrix time series of observed flows (for the same time steps than simulated) (mm/time step) with one column by hydrological model output named with the node ID (See CreateGRiwrm for details)
`...`	Further arguments for airGR::plot.OutputsModel and plot

Value

list of plots.

Examples

```
######################################################################
# Run the `airGR::RunModel_Lag` example in the GRiwrm fashion way #
# Simulation of a reservoir with a purpose of low-flow mitigation #
######################################################################

## ---- preparation of the InputsModel object

## loading package and catchment data
```

```
library(airGRiwrm)
data(L0123001)

## ---- 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
## 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

# This example is a network of 2 nodes which can be describe like this:
db <- data.frame(id = c("Reservoir", "GaugingDown"),
                 length = c(LengthHydro, NA),
                 down = c("GaugingDown", NA),
                 area = c(NA, BasinInfo$BasinArea),
                 model = c(NA, "RunModel_GR4J"),
                 stringsAsFactors = FALSE)

# Create GRiwrm object from the data.frame
griwrm <- CreateGRiwrm(db)
str(griwrm)

# Formatting observations for the hydrological models
# Each input data should be a matrix or a data.frame with the good id in the name of the column
Precip <- matrix(BasinObs$P, ncol = 1)
colnames(Precip) <- "GaugingDown"
PotEvap <- matrix(BasinObs$E, ncol = 1)
colnames(PotEvap) <- "GaugingDown"

# Observed flows contain flows that are directly injected in the model
Qobs = matrix(Qupstream, ncol = 1)
colnames(Qobs) <- "Reservoir"

# Creation of the GRiwrmInputsModel object (= a named list of InputsModel objects)
InputsModels <- CreateInputsModel(griwrm,
                            DatesR = BasinObs$DatesR,
                            Precip = Precip,
                            PotEvap = PotEvap,
                            Qobs = Qobs)
str(InputsModels)
```

```
## 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 GriwmRunOptions object
RunOptions <- CreateRunOptions(InputsModels,
                               IndPeriod_Run = Ind_Run)
str(RunOptions)

# Parameters of the SD models should be encapsulated in a named list
ParamGR4J <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
Param <- list(`GaugingDown` = c(Velocity, ParamGR4J))

# RunModel for the whole network
OutputsModels <- RunModel(InputsModels,
                          RunOptions = RunOptions,
                          Param = Param)
str(OutputsModels)

# Compare Simulation with reservoir and observation of natural flow
plot(OutputsModels, data.frame(GaugingDown = BasinObs$Qmm[Ind_Run]))
```

`plot.Qm3s` *Plot of a* `Qm3s` *object (time series of simulated flows)*

Description

Plot of a `Qm3s` object (time series of simulated flows)

Usage

```
## S3 method for class 'Qm3s'
plot(
  x,
  type = "l",
  xlab = "Date",
  ylab = expression("Flow (m"^"3" * "/s)"),
  main = "Simulated flows",
  col = rainbow(ncol(x) - 1),
  legend = colnames(x)[-1],
  legend.cex = 0.7,
  legend.x = "topright",
  legend.y = NULL,
  lty = 1,
  ...
)
```

Arguments

x	data.frame with a first column with POSIXt dates and followings columns with flows at each node of the network
type	character plot type (See plot.default), default "l"
xlab	character label for the x axis, default to "Date"
ylab	character label for the y axis, default to "Flow (m3/s)"
main	character main title for the plot, default to "Simulated flows"
col	character plotting color (See par), default to rainbow colors
legend	character see parameter legend of legend. Set to NULL if display of the legend is not wanted
legend.cex	character cex parameter for the text of the legend (See par)
legend.x, legend.y	Legend position, see x and y parameters in graphics::legend
lty	character or numeric The line type (See par)
...	Further arguments to pass to the matplot functions

Value

Screen plot window.

RunModel — *RunModel function for both* **airGR** *InputsModel and GRiwrmInputsModel object*

Description

RunModel function for both **airGR** InputsModel and GRiwrmInputsModel object

Usage

```
RunModel(x, ...)
```

Arguments

x	[object of class *InputsModel* or *GRiwrmInputsModel*] see CreateInputsModel for details
...	further arguments passed to or from other methods

Value

Either a list of OutputsModel object (for GRiwrmInputsModel) or an OutputsModel object (for InputsModel)

RunModel.GR *Run of a rainfall-runoff model on a sub-basin*

Description

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

Usage

```
## S3 method for class 'GR'
RunModel(x, RunOptions, Param, ...)
```

Arguments

x	[object of class InputsModel] InputsModel for airGR::RunModel
RunOptions	[object of class *RunOptions*] see airGR::CreateRunOptions for details
Param	numeric vector of model parameters (See details for SD lag model)
...	further arguments passed to or from other methods

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.

RunModel.GRiwrmInputsModel

RunModel function for GRiwrmInputsModel *object*

Description

RunModel function for *GRiwrmInputsModel* object

Usage

```
## S3 method for class 'GRiwrmInputsModel'
RunModel(x, RunOptions, Param, ...)
```

Arguments

x	[object of class *GRiwrmInputsModel*] see CreateInputsModel.GRiwrm for details
RunOptions	[object of class *GRiwrmRunOptions*] see CreateRunOptions.GRiwrmInputsModel for details
Param	list parameter values. The list item names are the IDs of the sub-basins. Each item is a numeric vector
...	Further arguments for compatibility with S3 methods

Value

An object of class *GRiwrmOutputsModel*. This object is a list of *OutputsModel* objects produced by RunModel.InputsModel for each node of the semi-distributed model.

Examples

```
######################################################################
# Run the `airGR::RunModel_Lag` example in the GRiwrm fashion way #
# Simulation of a reservoir with a purpose of low-flow mitigation #
######################################################################

## ---- preparation of the InputsModel object

## loading package and catchment data
library(airGRiwrm)
data(L0123001)

## ---- 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
## 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

# This example is a network of 2 nodes which can be describe like this:
db <- data.frame(id = c("Reservoir", "GaugingDown"),
                 length = c(LengthHydro, NA),
                 down = c("GaugingDown", NA),
                 area = c(NA, BasinInfo$BasinArea),
                 model = c(NA, "RunModel_GR4J"),
                 stringsAsFactors = FALSE)

# Create GRiwrm object from the data.frame
griwrm <- CreateGRiwrm(db)
str(griwrm)

# Formatting observations for the hydrological models
# Each input data should be a matrix or a data.frame with the good id in the name of the column
Precip <- matrix(BasinObs$P, ncol = 1)
colnames(Precip) <- "GaugingDown"
PotEvap <- matrix(BasinObs$E, ncol = 1)
colnames(PotEvap) <- "GaugingDown"

# Observed flows contain flows that are directly injected in the model
Qobs = matrix(Qupstream, ncol = 1)
colnames(Qobs) <- "Reservoir"

# Creation of the GRiwrmInputsModel object (= a named list of InputsModel objects)
InputsModels <- CreateInputsModel(griwrm,
                            DatesR = BasinObs$DatesR,
                            Precip = Precip,
                            PotEvap = PotEvap,
                            Qobs = Qobs)
str(InputsModels)

## 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 GriwmRunOptions object
RunOptions <- CreateRunOptions(InputsModels,
                               IndPeriod_Run = Ind_Run)
str(RunOptions)

# Parameters of the SD models should be encapsulated in a named list
ParamGR4J <- c(X1 = 257.238, X2 = 1.012, X3 = 88.235, X4 = 2.208)
Param <- list(`GaugingDown` = c(Velocity, ParamGR4J))

# RunModel for the whole network
OutputsModels <- RunModel(InputsModels,
                          RunOptions = RunOptions,
                          Param = Param)
str(OutputsModels)

# Compare Simulation with reservoir and observation of natural flow
```

```
plot(OutputsModels, data.frame(GaugingDown = BasinObs$Qmm[Ind_Run]))
```

`RunModel.InputsModel` *Wrapper for airGR::RunModel for one sub-basin*

Description

Wrapper for airGR::RunModel for one sub-basin

Usage

```
## S3 method for class 'InputsModel'
RunModel(x, RunOptions, Param, FUN_MOD = NULL, ...)
```

Arguments

`x`	[object of class *InputsModel*] see airGR::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`)
`...`	Further arguments for compatibility with S3 methods

Details

This function calls airGR::RunModel (See airGR::RunModel for further details).

The list produced by the function (See Value section of airGR::RunModel_GR4J) is here completed by an item *$Qsim_m3* storing the simulated discharge series in m3/s.

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.

RunModel.SD *Run a semi-distributed model from rainfall-runoff model outputs*

Description

Run a semi-distributed model from rainfall-runoff model outputs

Usage

```
## S3 method for class 'SD'
RunModel(x, RunOptions, Param, QcontribDown, ...)
```

Arguments

x	[object of class InputsModel] used as InputsModel parameter for airGR::RunModel_Lag
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
...	further arguments passed to or from other methods

Value

OutputsModel object. See airGR::RunModel_Lag

RunModel.Supervisor *RunModel function for a GRiwrmInputsModel object*

Description

RunModel function for a GRiwrmInputsModel object

Usage

```
## S3 method for class 'Supervisor'
RunModel(x, RunOptions, Param, ...)
```

Arguments

x	[object of class Supervisor] see CreateSupervisor for details
RunOptions	[object of class *GRiwrmRunOptions*] see [CreateRunOptions.GRiwrm] for details
Param	list parameter values. The list item names are the IDs of the sub-basins. Each item is a vector of numerical parameters
...	Further arguments for compatibility with S3 methods

Value

GRiwrmOutputsModel object which is a list of *OutputsModel* objects (See airGR::RunModel) for each node of the semi-distributed model

---

Severn — *Catchment attributes and hydro-meteorological timeseries for some gauging stations on the Severn River*

Description

Catchment attributes and hydro-meteorological timeseries for some gauging stations on the Severn River

Usage

```
Severn
```

Format

a list with 2 items:

- "BasinsInfo" which contains a data.frame with Gauging station identifier, name, coordinates (GPS), area (km2), mean elevation (m), station type, flow period start and end, the bank full flow (m3/s), the identifier of the following downstream station and the distance to the following downstream station
- "BasinObs" which contains a list with an item by gauging station which contains a data.frame with POSIXct dates, precipitations (mm/time step), potential evapotranspiration (mm/time step) and measured flows (mm/time step)

Source

These data are extracted from the CAMEL-GB dataset.

Coxon, G.; Addor, N.; Bloomfield, J.P.; Freer, J.; Fry, M.; Hannaford, J.; Howden, N.J.K.; Lane, R.; Lewis, M.; Robinson, E.L.; Wagener, T.; Woods, R. (2020). Catchment attributes and hydro-meteorological timeseries for 671 catchments across Great Britain (CAMELS-GB). NERC Environmental Information Data Centre. (Dataset). doi: 10.5285/8344E4F3D2EA44F58AFA86D2987543A9

Index

∗ **datasets**
    Severn, 31
-, *13*

airGR::Calibration, *3*
airGR::CreateCalibOptions, *5*
airGR::CreateIniStates, *16*
airGR::CreateInputsCrit, *3*, *11*
airGR::CreateInputsCrit_Lavenne, *11*
airGR::CreateInputsModel, *12–14*, *29*
airGR::CreateRunOptions, *16*, *26*
airGR::ErrorCrit_NSE, *10*
airGR::ErrorCrit_RMSE, *10*
airGR::plot.OutputsModel, *22*
airGR::RunModel, *26*, *29*, *31*
airGR::RunModel_GR4J, *29*
airGR::RunModel_Lag, *30*

Calibration
        *(*Calibration.GRiwrmInputsModel*)*, 2
Calibration.GRiwrmInputsModel, 2
character, *4–6*, *8*, *11*, *16*, *19–21*, *25*
colnames, *4*
ConvertMeteoSD, 4
CreateCalibOptions, *3*
CreateCalibOptions
        *(*CreateCalibOptions.GRiwrmInputsModel*)*, 5
CreateCalibOptions.GRiwrmInputsModel, 5
CreateController, 6, *19*
CreateGRiwrm, *4*, 7, *13*, *18*, *21*, *22*
CreateInputsCrit, *3*
CreateInputsCrit
        *(*CreateInputsCrit.GRiwrmInputsModel*)*, 10
CreateInputsCrit.GRiwrmInputsModel, 10
CreateInputsModel, *3*, *5*, *10*, 12, *16*, *25*, *26*, *29*
CreateInputsModel.GRiwrm, *12*, 12, *18*, *20*, *27*
CreateRunOptions, *3*, *10*, *29*, *30*
CreateRunOptions
        *(*CreateRunOptions.GRiwrmInputsModel*)*, 15
CreateRunOptions.GRiwrmInputsModel, 15, *27*
CreateSupervisor, *6*, 18, *31*

data.frame, *4*, *7*, *8*, *11*, *13*, *25*, *31*
DiagrammeR::mermaid, *21*

environment, *18*

function, *5*, *6*, *16*

getNoSD_Ids, 19
getSD_Ids, 20
graphics::legend, *25*
GRiwrm *(*CreateGRiwrm*)*, 7

isNodeDownstream, 20

legend, *25*
list, *3*, *5*, *7*, *11*, *16*, *19*, *22*, *25*, *27*, *31*
logical, *4*, *8*, *13*, *20*, *21*

matplot, *25*
matrix, *4*, *6*, *11*, *13*, *22*

NA, *8*
NULL, *25*
numeric, *4*, *6*, *8*, *11*, *13*, *16*, *18*, *21*, *25–27*

par, *25*
plot, *22*
plot.default, *25*
plot.GRiwrm, 21
plot.GRiwrmOutputsModel, 22
plot.Qm3s, 24

POSIXct, *18*, *31*
POSIXt, *13*, *25*

RunModel, 25
RunModel.GR, 26
RunModel.GRiwrmInputsModel, *22*, 26
RunModel.InputsModel, *27*, 29
RunModel.SD, 30
RunModel.Supervisor, 30
RunModel_CemaNeigeGR4J, *26*, *29*
RunModel_GR4J, *26*, *29*

Severn, 31

vector, *7*, *11*, *13*, *27*